Machine Learning for Insurers and Actuaries

Arthur Charpentier

2025

Learning, with an algorithmic perspective

(lecture 2)

Discrimination and Insurance

"What is unique about insurance is that even statistical discrimination which by definition is absent of any malicious intentions, poses significant moral and legal challenges. Why? Because on the one hand, policy makers would like insurers to treat their insureds equally, without discriminating based on race, gender, age, or other characteristics, even if it makes statistical sense to discriminate (...) On the other hand, at the core of insurance business lies discrimination between risky and non-risky insureds. But riskiness often statistically correlates with the same characteristics policy makers would like to prohibit insurers from taking into account." Avraham (2017)

Glenn (2003) claimed that there are many ways to rate accurately. Insurers can rate risks in many different ways depending on the stories they tell on which characteristics are important and which are not. "The fact that the selection of risk factors is subjective and contingent upon narratives of risk and responsibility has in the past played a far larger role than whether or not someone with a wood stove is charged

higher premiums." Going further, "virtually every aspect of the insurance industry is predicated on stories first and then numbers."

"all models are wrong but some models are useful," Box et al. (2011) (in other words, any model is at best a useful fable).

Definition 2.1: Pure premium (homogeneous risks)

Let Y be the non-negative random variable corresponding to the total annual loss associated with a given policy, then the pure premium is $\mathbb{E}[Y]$.

Proposition 2.1: Law of Large Numbers (2)

Consider an infinite collection of i.i.d. random variables $Y, Y_1, Y_2, \cdots, Y_n, \cdots$ in a probabilistic space $(\Omega, \mathcal{F}, \mathbb{P})$, with finite expected value, then

$$\underbrace{\frac{1}{n}\sum_{i=1}^{n} Y_i}_{\text{(empirical) average}} \xrightarrow{\text{a.s.}} \underbrace{\mathbb{E}(Y)}_{\text{expected value}}, \text{ as } n \to \infty.$$

More realistically, population is heterogeneous (with respect to risks), with some covariates $\boldsymbol{x}$ (legitimate, or not).

Definition 2.2: Pure premium (heterogeneous risks)

Let Y be the non-negative random variable corresponding to the total annual loss associated with a given policy, with covariates $\boldsymbol{x}$, then the pure premium is $\mu(\boldsymbol{x}) = \mathbb{E}[Y|\boldsymbol{X} = \boldsymbol{x}]$.

In this general setting, $\boldsymbol{x}$ consist in numeric or categorical variables.

Definition 2.3: Balance Property

A pricing function m satisfies the balance property if $\mathbb{E}_{\boldsymbol{X}}[m(\boldsymbol{X})] = \mathbb{E}_Y[Y]$.

Proposition 2.2: Law of total expectations

$$\mathbb{E}_{\boldsymbol{X}}[\mu(\boldsymbol{X})] = \mathbb{E}_{Y}[Y] \text{ where } \mu(\boldsymbol{X}) = \mathbb{E}_{Y|\boldsymbol{X}}[Y|\boldsymbol{X}].$$

Proof Since $\mathbb{E}(Y) = \int y f_Y(y)\mathrm{d}y$ and $\mathbb{E}(Y|\boldsymbol{X}=\boldsymbol{x}) = \int y f_{Y|\boldsymbol{x}}(y|\boldsymbol{x})\mathrm{d}y$,

$$\mathbb{E}(\mathbb{E}(X|Y)) = \int \left(\int x \mathbb{P}[X=x|Y=y]\mathrm{d}x \right) \mathbb{P}[Y=y]\mathrm{d}y = \int \int x \mathbb{P}[X=x, Y=y]\mathrm{d}x\mathrm{d}y$$
$$= \int x \left(\int \mathbb{P}[X=x, Y=y]\mathrm{d}y \right) \mathrm{d}x = \int x \mathbb{P}[X=x]\mathrm{d}x = \mathbb{E}(X).$$

Discrimination and Insurance

Homogeneous risk sharing

	Policyholder	Insurer
Loss	$\mathbb{E}[Y]$	$Y - \mathbb{E}[Y]$
Average loss	$\mathbb{E}[Y]$	0
Variance	0	$\mathrm{Var}[Y]$

$\mathbb{E}[Y]$ is the premium paid, and Y the total loss,
from De Wit and Van Eeghen (1984) and Denuit and Charpentier (2004)

Discrimination and Insurance

Heterogeneous risk sharing, with perfect information

	Policyholder	Insurer
Loss	$\mathbb{E}[Y\|\Theta]$	$Y-\mathbb{E}[Y\|\Theta]$
Average loss	$\mathbb{E}[Y]$	0
Variance	$\mathrm{Var}[\mathbb{E}[Y\|\Theta]]$	$\mathrm{Var}[Y-\mathbb{E}[Y\|\Theta]]$

where Θ denotes the heterogeneous risk factor.

The term on the bottom right is $\mathbb{E}[\mathrm{Var}[Y|\Theta]]$, corresponding to the standard variance decomposition (or Pythagoras theorem)

$$\mathrm{Var}[Y]=\mathrm{Var}[\mathbb{E}[Y|\Theta]]+\mathbb{E}[\mathrm{Var}[Y|\Theta]].$$

Proposition 2.3: Variance decomposition (1)

For any measurable random variable Y with finite variance

$$\mathrm{Var}[Y] = \underbrace{\mathbb{E}[\mathrm{Var}[Y|\Theta]]}_{\to \text{ insurer}} + \underbrace{\mathrm{Var}[\mathbb{E}[Y|\Theta]]}_{\to \text{ policyholder}}.$$

Proof:

$$\begin{aligned}\mathrm{Var}[Y] &= \mathbb{E}\left[Y^2\right] - \mathrm{E}[Y]^2 = \mathbb{E}\left[\mathrm{Var}[Y|\Theta] + \mathbb{E}[Y|\Theta]^2\right] - \mathbb{E}[\mathbb{E}[Y|\Theta]]^2 \\ &= (\mathbb{E}[\mathrm{Var}[Y|\Theta]]) + \left(\mathbb{E}\left[\mathbb{E}[Y|\Theta]^2\right] - \mathbb{E}[\mathbb{E}[Y|\Theta]]^2\right) = \mathbb{E}[\mathrm{Var}[Y|\Theta]] + \mathrm{Var}[\mathbb{E}[Y|\Theta]].\end{aligned}$$

Discrimination and Insurance

Heterogeneous risk sharing, with imperfect information

	Policyholder	Insurer
Loss	$\mathbb{E}[Y\|\boldsymbol{X}]$	$Y-\mathbb{E}[Y\|\boldsymbol{X}]$
Average loss	$\mathbb{E}[Y]$	0
Variance	$\mathrm{Var}[\mathbb{E}[Y\|\boldsymbol{X}]]$	$\mathbb{E}[\mathrm{Var}[Y\|\boldsymbol{X}]]$

$$\mathbb{E}[\mathrm{Var}[Y|\boldsymbol{X}]] = \underbrace{\mathbb{E}[\mathrm{Var}[Y|\Theta]]}_{\text{perfect ratemaking}} + \underbrace{\mathbb{E}\{\mathrm{Var}[\mathbb{E}[Y|\Theta]|\boldsymbol{X}]\}}_{\text{misclassification}}$$

This "misclassification" term (on the right) is called "subsidierende solidariteit" in De Pril and Dhaene (1996), or "subsidiary solidarity", as opposed to "kanssolidariteit" or "random solidarity" term (on the left).

Discrimination and Insurance

Proposition 2.4: Variance decomposition (2)

For any measurable random variable Y with finite variance

$$\mathrm{Var}[Y] = \underbrace{\mathbb{E}[\mathrm{Var}[Y|\boldsymbol{X}]]}_{\to \text{ insurer}} + \underbrace{\mathrm{Var}[\mathbb{E}[Y|\boldsymbol{X}]]}_{\to \text{ policyholder}},$$

where

$$\begin{aligned}\mathbb{E}[\mathrm{Var}[Y|\boldsymbol{X}]] &= \mathbb{E}[\mathbb{E}[\mathrm{Var}[Y|\Theta]|\boldsymbol{X}]] + \mathbb{E}[\mathrm{Var}[\mathbb{E}[Y|\Theta]|\boldsymbol{X}]] \\ &= \underbrace{\mathbb{E}[\mathrm{Var}[Y|\Theta]]}_{\text{perfect ratemaking}} + \underbrace{\mathbb{E}\{\mathrm{Var}[\mathbb{E}[Y|\Theta]|\boldsymbol{X}]\}}_{\text{misclassification}}.\end{aligned}$$

The aim of risk classification, as explained in Wortham (1986), is to identify the specific characteristics that are supposed to determine an individual's propensity to

Discrimination and Insurance

suffer an adverse event, forming groups within which the risk is (approximately) equally shared. The problem, of course, is that the characteristics associated with various types of risk are almost infinite; as they cannot all be identified and priced in every risk classification system, there will necessarily be unpriced sources of heterogeneity between individuals in a given risk class.

[Most] "actuaries cannot think of individuals except as members of groups" claimed Brilmayer et al. (1979). Each individual is assigned the same value as all other members of the group to which it is assigned.

Simon (1987, 1988), and then Feeley and Simon (1992), defined "actuarialism," that designate the use of statistics to guide "*class-based decision-making*," used to price pensions and insurance. As explained in Harcourt (2015), this "actuarial classification" is the constitution of groups with no experienced social significance for the participants. A person classified as a particular risk by an insurance company shares nothing with the other people so classified, apart from a series of formal characteristics (e.g. age, sex, marital status, etc.).

Price Optimization

Decision theory under uncertainty (see Charpentier (2014)),

$$X \preceq Y \iff \mathcal{R}(X) \leq \mathcal{R}(Y),$$

A classical representation is $\mathcal{R}(Y) = \mathbb{E}[u(\omega - Y)]$, as in Neumann and Morgenstern (1947), where ω is the initial wealth.

u denotes the utility of the agent

Let π denote the premium asked to transfer risk (loss) Y,

$$\begin{cases} u(\omega - \pi) > \mathbb{E}\big[u(\omega - Y)\big] : & \text{purchases insurance} \\ u(\omega - \pi) < \mathbb{E}\big[u(\omega - Y)\big] : & \text{does not purchase insurance} \end{cases}$$

Find π such that $u(\omega - \pi) = \mathbb{E}\big[u(\omega - Y)\big]$.

Price Optimization

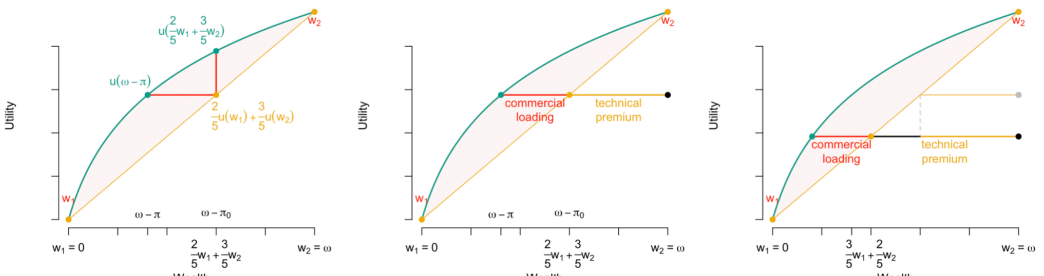
π such that $u(\omega - \pi) = \mathbb{E}\big[u(\omega - Y)\big]$ could be seem as the willingness to pay to transfer the risk.

> **Willingness to Pay**
> In behavioral economics, willingness to pay (WTP) is the maximum price at or below which a consumer will definitely buy one unit of a product.[1] This corresponds to the standard economic view of a consumer reservation price. W

Price Optimization

Definition 2.4: Indifference utility principle

Let Y be the non-negative random variable corresponding to the total annual loss associated with a given policy, for a policyholder with utility u and wealth w, the indifference premium is $\pi = \omega - u^{-1}\left(\mathbb{E}\left[u(\omega - Y)\right]\right)$.



Price Optimization

Price Walking

Price walking, or the loyalty penalty, is a form of price discrimination whereby longstanding, loyal customers of a service provider are charged higher prices for the same services compared to customers that have just switched to that provider. The pricing strategy is common in the insurance and telecommunications industries. It is used to acquire new customers with artificially low rates or other incentives not available to existing clients, effectively using existing customers to subsidize the prices offered to new clients. W

Ratemaking, Life Insurance

Excerpt from the Men and Women life tables in 1720 (source: Struyck (1912)).
Mortality, as a function of the age and the gender of the individual.

Table des Hommes.

Années	Personnes	Années	Personnes	Années	Personnes	Années	Personnes	Années	Personnes	Années	Personnes
5	710	20	607	35	474	50	313	65	142	80	33
6	697	21	599	36	464	51	301	66	132	81	29
7	688	22	591	37	454	52	289	67	123	82	25
8	681	23	583	38	444	53	277	68	114	83	22
9	675	24	575	39	434	54	265	69	105	84	19
10	670	25	567	40	424	55	253	70	97	85	16
11	665	26	558	41	414	56	241	71	89	86	13
12	660	27	549	42	404	57	229	72	82	87	10
13	654	28	540	43	393	58	217	73	75	88	8
14	648	29	531	44	382	59	206	74	68	89	6
15	642	30	522	45	371	60	195	75	61	90	4
16	635	31	513	46	360	61	184	76	54	91	3
17	628	32	504	47	349	62	173	77	48	92	2
18	621	33	494	48	337	63	162	78	43	93	1
19	614	34	484	49	325	64	152	79	38	94	

Table des femmes.

Années	Personnes	Années	Personnes	Années	Personnes	Années	Personnes	Années	Personnes	Années	Personnes
5	711	20	624	35	508	50	373	65	205	80	55
6	700	21	617	36	500	51	362	66	194	81	47
7	692	22	610	37	492	52	351	67	183	82	40
8	685	23	603	38	484	53	340	68	172	83	34
9	679	24	596	39	476	54	329	69	161	84	29
10	674	25	588	40	468	55	318	70	150	85	24
11	669	26	580	41	459	56	306	71	140	86	20
12	664	27	572	42	450	57	294	72	130	87	17
13	660	28	564	43	441	58	282	73	120	88	14
14	656	29	556	44	432	59	271	74	110	89	11
15	652	30	548	45	423	60	260	75	100	90	8
16	647	31	540	46	414	61	249	76	90	91	6
17	642	32	532	47	404	62	238	77	81	92	4
18	636	33	524	48	394	63	227	78	72	93	2
19	630	34	516	49	384	64	216	79	63	94	1

Ratemaking, Life Insurance

Excerpt from the Men and Women life tables in 1720 (source: Struyck (1912))
Mortality, as a function of the age and the gender of the individual.

men						women					
x	L_x	${}_5p_x$	x	L_x	${}_5p_x$	x	L_x	${}_5p_x$	x	L_x	${}_5p_x$
0	1000	29.0%	45	371	16.6%	0	1000	28.9%	45	423	11.8%
5	710	5.6%	50	313	19.2%	5	711	5.2%	50	373	14.7%
10	670	4.2%	55	253	22.9%	10	674	3.3%	55	318	18.2%
15	642	5.5%	60	195	27.2%	15	652	4.3%	60	260	21.2%
20	607	6.6%	65	142	31.7%	20	624	5.8%	65	205	26.8%
25	567	7.9%	70	97	37.1%	25	588	6.8%	70	150	33.3%
30	522	9.2%	75	61	45.9%	30	548	7.3%	75	100	45.0%
35	474	10.5%	80	33	51.5%	35	508	7.9%	80	55	56.4%
40	424	12.5%	85	16		40	468	9.6%	85	24	

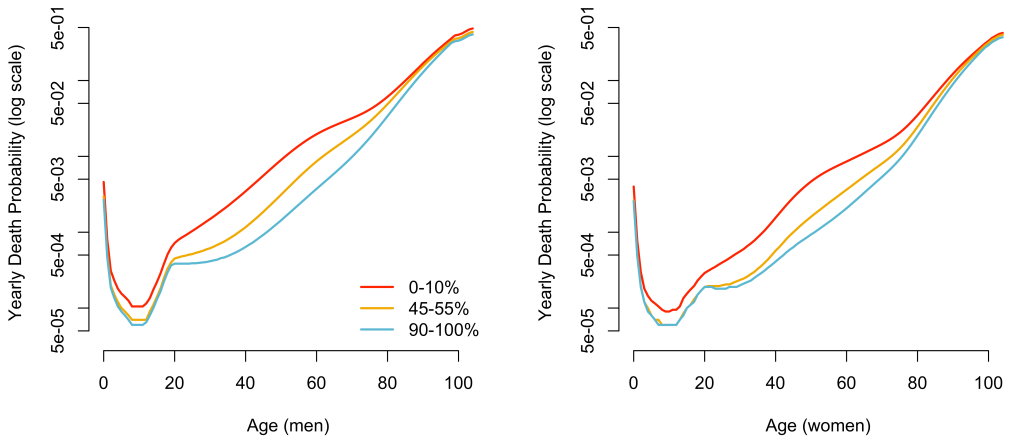
Ratemaking, Life Insurance

Excerpt from the Men and Women life tables in 2016 (source: Blanpain (2018))
Mortality, as a function of the age, the gender and the wealth of the individual.

men			
x	0-5%	45-50%	95-100%
0	100000	100000	100000
10	99299	99566	99619
20	99024	99396	99469
30	97930	98878	99094
40	95595	98058	98627
50	90031	96172	97757
60	77943	91050	95649
70	59824	79805	90399
80	38548	59103	76115
90	13337	23526	38837
100	530	1308	3231

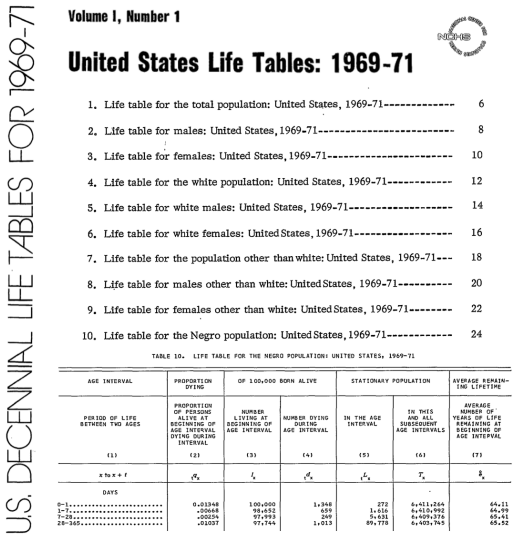
women			
x	0-5%	45-50%	95-100%
0	100000	100000	100000
10	99385	99608	99623
20	99227	99506	99526
30	98814	99302	99340
40	97893	98960	99074
50	95021	97959	98472
60	88786	95543	97192
70	79037	90408	94146
80	63224	79117	85825
90	31190	45750	55918
100	2935	5433	8717

Ratemaking, Life Insurance



Force of mortality (log scale) for various income quantile, in France, Blanpain (2018).

Ratemaking, Life Insurance



U.S. DECENNIAL LIFE TABLES FOR 1969-71

Volume 1, Number 1

United States Life Tables: 1969-71

TABLE 10. LIFE TABLE FOR THE NEGRO POPULATION: UNITED STATES, 1969-71

AGE INTERVAL	PROPORTION DYING	OF 100,000 BORN ALIVE		STATIONARY POPULATION		AVERAGE REMAINING LIFETIME
PERIOD OF LIFE BETWEEN TWO AGES	PROPORTION OF PERSONS BEGINNING OF AGE INTERVAL DYING DURING INTERVAL	NUMBER LIVING AT BEGINNING OF AGE INTERVAL	NUMBER DYING DURING AGE INTERVAL	IN THE AGE INTERVAL	IN THIS AND ALL SUBSEQUENT AGE INTERVALS	AVERAGE NUMBER OF YEARS OF LIFE REMAINING AT BEGINNING OF AGE INTERVAL
(1)	(2)	(3)	(4)	(5)	(6)	(7)
x to x + f	$_nq_x$	l_x	$_nd_x$	$_nL_x$	T_x	$\mathring{e}_x$
DAYS						
0-1.....	0.03948	100,000	1,348	272	6,411,264	64.11
1-7.....	.00468	98,652	659	1,616	6,410,992	64.99
7-28.....	.00254	97,993	249	5,631	6,409,376	65.41
28-365.....	.01007	97,744	1,013	89,778	6,403,745	65.52

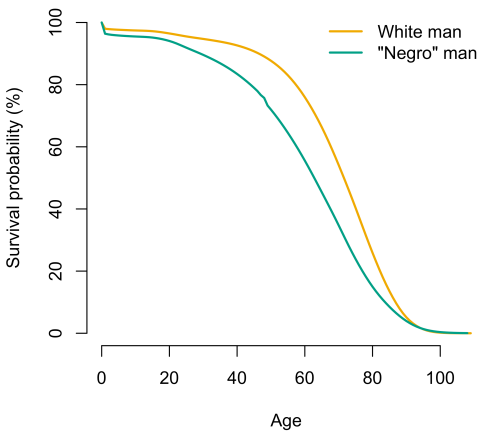
Mortality, gender and "race"
Frederick L. Hoffman
Hoffman (1896, 1918, 1931)

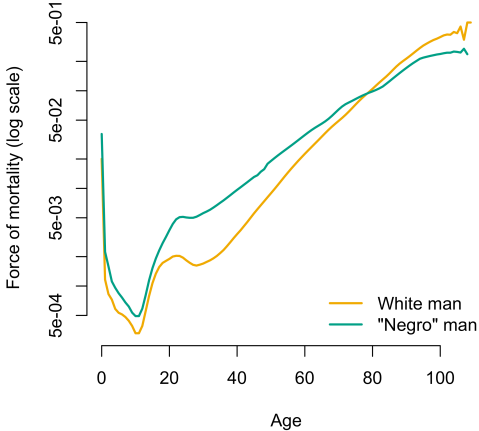
Ratemaking, Life Insurance

White, men					
x	L_x	${}_5p_x$	x	L_x	${}_5p_x$
0	100000	2.3%	55	83001	8.5%
5	97671	0.2%	60	75969	12.7%
10	97441	0.2%	65	66343	18.4%
15	97208	0.7%	70	54138	25.5%
20	96480	1.0%	75	40324	35.8%
25	95524	0.8%	80	25885	47.7%
30	94716	0.9%	85	13527	62.1%
35	93843	1.3%	90	5125	75.1%
40	92631	2.1%	95	1274	85.2%
45	90725	3.3%	100	189	90.5%
50	87690	5.3%	105	18	100.0%

"Negro", men					
x	L_x	${}_5p_x$	x	L_x	${}_5p_x$
0	100000	4.2%	55	66101	13.1%
5	95826	0.3%	60	57457	17.4%
10	95497	0.4%	65	47485	22.2%
15	95161	1.2%	70	36925	29.8%
20	94053	2.3%	75	25921	36.1%
25	91904	2.5%	80	16560	41.7%
30	89584	3.0%	85	9648	51.3%
35	86885	4.0%	90	4696	63.4%
40	83441	5.4%	95	1721	71.6%
45	78976	7.2%	100	489	74.8%
50	73282	9.8%	105	123	100.0%

Ratemaking, Life Insurance





Force of mortality (log scale) white men and "Negro" men, 1968-71, U.S.

Ratemaking, Motor Insurance

Groups, or risk classes, are built on the basis of available data, and exist primarily as the product of actuarial models.

For example, as mentioned in Bailey and Simon (1960), in motor insurance five risk classes can be considered,

- "*pleasure, no male operator under 25*," (reference),
- "*pleasure, non-principal male operator under 25*," +65%,
- "*business use*," +65%,
- "*married owner or principal operator under 25*," +65%,
- "*unmarried owner or principal operator under 25*," +140%.

Comparative Effectiveness of Merit Rating and Class Rating

Table 1 at the end of this section shows the Canadian automobile experience[1] arranged to show what it would have looked like if there had been (1) merit rating without class rating and (2) class rating without merit rating. The premiums have been adjusted to what they would have been if all the cars had been written at 1 B rates, by use of the approximate relativities:

Merit Rating Definition	Relativity
A-licensed and accident free three or more years	65
X-licensed and accident free two years	80
Y-licensed and accident free one year	90
B-all others	100
Class Definitions	
1-pleasure, no male operator under 25	100
2-pleasure, non-principal male operator under 25	165
3-business use	165
4-unmarried owner or principal operator under 25	240
5-married owner or principal operator under 25	165

There is no "physical basis" for group members to identify other members of *their* group, in the sense that they usually don't share anything, except some common characteristics, Gandy (2016).

Ratemaking, Motor Insurance

"1. The classification should bring together risks which have inherent in their operation the same causes of loss.
2. The variation from risk to risk in the strength of each cause or at least of the more important should not be greater than can be handled by the formula by which the classification is subdivided, i.e., the Schedule and / or Experience Rating Plan used.
3. The classification should not cover risks which include, as important elements of their hazard, causes which are not common to all.
4. The classification system and the formula for its extension (Schedule and / or Experience Rating Plans) should be harmonious.
5. The basis throughout should be the outward, recognizable indicia of the presence and potency of the several inherent causes of loss including extent as well as occurrence of loss," Mowbray (1921).

Ratemaking, Motor Insurance

[Most] "actuaries cannot think of individuals except as members of groups" claimed Brilmayer et al. (1979). Each individual is assigned the same value as all other members of the group to which it is assigned.

Ratemaking, Motor Insurance

	CA	HI	GA	NC	NY	MA	PA	FL	TX	AL	ON	NB	NL	QC
Gender	☐	☐	☑	☐	☑	☐	☐	☑	☑	☑	☑	☐	☐	☑
Age	☐	☐	☑	☐*	☑	☐	☑	☑	☑	☑*	☑	☐	☐	☑
Driving experience	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
Credit history	☐	☐	☑	☑	☑	☐	☑*	☑	☑	☐*	☐	☑*	☐	☑
Education	☐	☐	☐	☐	☐	☐	☑	☑	☑	☑	☑	☑	☑	☑
Occupation	☐	☐	☐	☑	☐	☐	☑	☑	☑	☑	☑	☑	☑	☑
Employment status	☐	☐	☐	☑	☐	☐	☑	☑	☑	☑	☑	☑	☑	☑
Marital status	☑	☐	☑	☑	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑
Housing situation	☐	☐	☑	☑	☑	☐	☑	☑	☑	☐	☐	☑	☑	☑
Address/ZIP code	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☐	☑	☑	☑
Insurance history	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑

CA: California, HI: Hawaii, GA: Georgia, NC: North Carolina, NY: New York, MA: Massachusetts, PA: Pennsylvania, FL: Florida, TX: Texas, AL: Alberta, ON: Ontario, NB: New-Brunswick, NL: Newfoundland-Labrador, QC: Québec.

Ordinary Least Squares

> **Least Squares**
> In regression analysis, least squares is a parameter estimation method based on minimizing the sum of the squares of the residuals (a residual being the difference between an observed value and the fitted value provided by a model) made in the results of each individual equation. W

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik} + \varepsilon_i,$$

Write with a matrix form, where $p = k + 1$,

$$\underbrace{\boldsymbol{Y}}_{(n\times 1)} = \underbrace{\boldsymbol{X}}_{(n\times p)} \underbrace{\beta}_{(p\times 1)} + \underbrace{\varepsilon}_{(n\times 1)}$$

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \boldsymbol{X} = \begin{pmatrix} x_{10} & x_{11} & \cdots & x_{1k} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n0} & x_{n1} & \cdots & x_{nk} \end{pmatrix}, \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_k \end{pmatrix}, \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

Ordinary Least Squares

$\mathcal{A}_1$: Design matrix $\boldsymbol{X}$ is a full rank matrix.

$$\widehat{\beta} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}.$$

$\mathcal{A}_2$: Les erreurs sont centrées, de même variance et non corrélées
$\Leftrightarrow \mathbb{E}(\varepsilon) = \mathbf{0}$ et $\text{Var}(\varepsilon) = \sigma^2 \mathbb{I}_n$.

$$\mathbb{E}(\widehat{\beta}) = \underbrace{(\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{X}}_{=\mathbb{I}} \beta + (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \underbrace{\mathbb{E}(\varepsilon)}_{=0} = \beta.$$

$$\text{Var}(\widehat{\beta}) = \sigma^2 (\boldsymbol{X}^\top \boldsymbol{X})^{-1}$$

$\mathcal{A}_2^{\text{gauss}}$: ε est un vecteur gaussien centré de matrice de covariance $\sigma^2 \mathbb{I}_n$, i.e.
$\varepsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbb{I}_n)$

Ordinary Least Squares

$\mathcal{A}_3$: La matrice de design $\boldsymbol{X}$ est telle que lorsque $n \to \infty$, $\frac{1}{n}(\boldsymbol{X}^\top \boldsymbol{X}) \to \boldsymbol{Q}$ où Q est une matrice définie positive. De plus, $h_n = \max_{1 \leq i,j \leq n} (\Pi_{\boldsymbol{X}})_{ij} \to 0$ lorsque $n \to \infty$ Sous les hypothèses $\mathcal{A}_1$, $\mathcal{A}_2$ et $\mathcal{A}_3^{\text{clt}}$, alors lorsque $n \to \infty$

$$\sqrt{n}\left(\widehat{\beta} - \beta\right) \xrightarrow{\mathcal{L}} \mathcal{N}(0, \sigma^2 \boldsymbol{Q}^{-1})$$

Since

$$\widehat{\sigma}^{-1} \boldsymbol{Q}^{1/2} \sqrt{n}\left(\widehat{\beta} - \beta\right) \xrightarrow{\mathcal{L}} \mathcal{N}(0, \mathbb{I}_p)$$

ou encore

$$\widehat{\sigma}^{-1} (\boldsymbol{X}^\top \boldsymbol{X})^{1/2} \left(\widehat{\beta} - \beta\right) \xrightarrow{\mathcal{L}} \mathcal{N}(0, \mathbb{I}_p)$$

soit, lorsque $n \to \infty$

$$\widehat{\beta} - \beta \approx \mathcal{N}\left(0, \widehat{\sigma}^2 (\boldsymbol{X}^\top \boldsymbol{X})^{-1}\right).$$

Bootstrap & Tests

Consider the test of $H_0 : \beta_j = 0$,

1. compute $t_n = \dfrac{(\widehat{\beta}_j - \beta_j)^2}{\widehat{\sigma}_j^2}$
2. generate B boostrap samples, under the null assumption H_0
3. for each boostrap sample, compute $t_n^{(b)} = \dfrac{(\widehat{\beta}_j^{(b)} - \widehat{\beta}_j)^2}{\widehat{\sigma}_j^{2(b)}}$
4. reject H_0 if $\dfrac{1}{B}\sum_{i=1}^{B} \mathbf{1}(t_n > t_n^{(b)}) < \alpha$.

Bootstrap & Tests

What does "generate B boostrap samples, under the null assumption H_0" mean ?

Example : $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$ with $H_0 : \beta_1 = 0$.

2.1. Estimate the model under H_0, i.e. $y_i = \beta_0 + \eta_i$, and save $\{\widehat{\eta}_1, \cdots, \widehat{\eta}_n\}$

2.2. Define $\widetilde{\boldsymbol{\eta}} = \{\widetilde{\eta}_1, \cdots, \widetilde{\eta}_n\}$ with $\widetilde{\eta} = \sqrt{\dfrac{n}{n-1}}\widehat{\eta}$

2.3. Draw (with replacement) residuals $\widetilde{\boldsymbol{\eta}}^{(b)} = \{\widetilde{\eta}_1^{(b)}, \cdots, \widetilde{\eta}_n^{(b)}\}$

2.4. Set $y_i^{(b)} = \widehat{\beta}_0 + \widetilde{\eta}_i^{(b)}$

2.5. Estimate the regression model $y_i^{(b)} = \beta_0^{(b)} + \beta_1^{(b)} x_i + \varepsilon_i^{(b)}$

3. for each boostrap sample, compute $t_n^{(b)} = \dfrac{(\widehat{\beta}_j^{(b)} - \widehat{\beta}_j)^2}{\widehat{\sigma}_j^{2(b)}}$

4. reject H_0 if $\dfrac{1}{B}\displaystyle\sum_{i=1}^{B} \mathbf{1}(t_n > t_n^{(b)}) < \alpha$.

Regression with Weights

Suppose that instead of $\{(y_1, \boldsymbol{x}_1), (y_2, \boldsymbol{x}_2), \cdots, (y_n, \boldsymbol{x}_n)\}$ we have $\{(y_1, \boldsymbol{x}_1, \omega_1), (y_2, \boldsymbol{x}_2, \omega_2), \cdots, (y_n, \boldsymbol{x}_n, \omega_n)\}$. To solve

$$\sum_{i=1}^{n} \omega_i (y_i - \boldsymbol{x}_i^\top \beta)^2 = (\boldsymbol{y} - \boldsymbol{X}\beta)^\top \boldsymbol{W} (\boldsymbol{y} - \boldsymbol{X}\beta)$$

where $\boldsymbol{W} = \text{diag}(\boldsymbol{\omega})$.
First order condition is $\left(\boldsymbol{X}^\top \boldsymbol{W} \boldsymbol{X}\right) \widehat{\beta} = \boldsymbol{X}^\top \boldsymbol{W} \boldsymbol{X}$.

Local Regression

Nadaraya-Watson is a simple nonparametric regression method using kernel-weighted averages

It is based on the idea that the conditional mean $\mathbb{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}]$ is a "local average" of Y in the neighborhood of $\boldsymbol{x}$.

Following the "law of the unconscious statistician," in Schervish and DeGroot (2014), for some $\boldsymbol{x}$,

$$\mathbb{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}] = \lim_{h \to 0} \mathbb{E}[Y \mid \boldsymbol{X} \in \mathcal{B}_h(\boldsymbol{x})] \text{ where } \mathcal{B}_h(\boldsymbol{x}) = \{\boldsymbol{x}' \in \mathcal{X} : \|\boldsymbol{x} - \boldsymbol{x}'\| \leq h\}$$

Local Regression

Definition 2.5: Nadaraya-Watson Estimator

Given data $(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_n, y_n)$, the estimate at point $\boldsymbol{x}$ is:

$$\widehat{m}(x) = \frac{\sum_{i=1}^{n} K_h(\boldsymbol{x} - \boldsymbol{x}_i) y_i}{\sum_{i=1}^{n} K_h(\boldsymbol{x} - \boldsymbol{x}_i)}$$

where $K_h(u) = \frac{1}{h} K\left(\frac{u}{h}\right)$ for some kernel K (on $\mathcal{X}$) and with bandwidth h.

In the univariate case, common kernels are Gaussian, Epanechnikov, Uniform

Local Regression

It's a weighted average of the y_i's, where weights depend on distance from x.

$$\widehat{m}(x) = \underset{\beta_0 \in \mathbb{R}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} w_i(x)(y_i - \beta_0)^2 \right\} \text{ where } w_i(x) = \frac{K_h(\boldsymbol{x} - \boldsymbol{x}_i)}{\sum_{i=1}^{n} K_h(\boldsymbol{x} - \boldsymbol{x}_i)}$$

More generally, we can consider some local regression, also known as LOESS (Locally Estimated Scatterplot Smoothing) or LOWESS

Local Regression

Definition 2.6: LOESS Estimator

Given data $(x_1, y_1), \cdots, (x_n, y_n)$, the estimate at point x is:

$$\widehat{m}(x) = \widehat{\beta}_0 + \widehat{\beta}_1 x, \text{ where } (\widehat{\beta}_0, \widehat{\beta}_1) = \underset{(\beta_0,\beta_1)\in\mathbb{R}^2}{\text{argmin}} \left\{ \sum_{i=1}^{n} w_i(x)(y_i - \beta_0 - \beta_1 x)^2 \right\}$$

where

$$w_i(x) = \frac{K_h(\boldsymbol{x} - \boldsymbol{x}_i)}{\sum_{i=1}^{n} K_h(\boldsymbol{x} - \boldsymbol{x}_i)}$$

where $K_h(u) = \frac{1}{h} K\left(\frac{u}{h}\right)$ for some kernel K (on $\mathcal{X}$) and with bandwidth h.

Local regression is very sensitive in hyper-parameter selection.

Isotonic Regression

Isotonic Regression is a type of regression that fits a non-decreasing (or non-increasing) function to data. It is used when there is a known order or monotonicity constraint in the response variable.

Definition 2.7: Isotonic Regression

Given data points $(x_1, y_1), \cdots, (x_n, y_n)$ with $x_1 < x_2 < \cdots < x_n$, the isotonic regression is an interpolation between $(x_1, \hat{y}_1), \cdots, (x_n, \hat{y}_n)$ such that:

$$\min_{\hat{y}} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

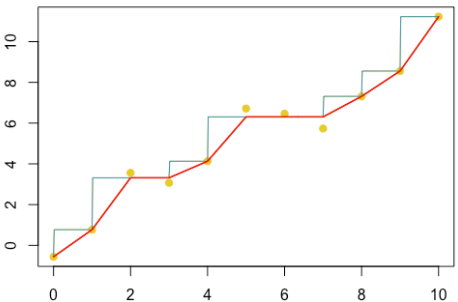
subject to:

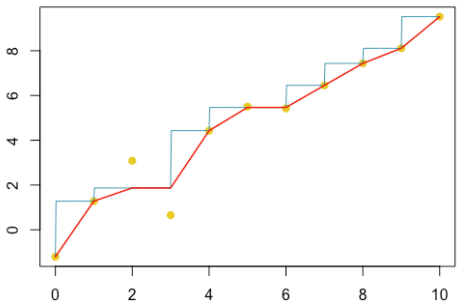
$$\hat{y}_1 \leq \hat{y}_2 \leq \cdots \leq \hat{y}_n$$

Unique solution via Pool Adjacent Violators Algorithm (PAVA), De Leeuw et al. (2010),

Isotonic Regression

```
> ir = isoreg(x=d$x, y=d$y)
> isofit1 = as.stepfun(ir)
> isofit2 = function(u) approx(ir$x, ir$yf, xout=u)$y
```





Correlated Features, Profile Likelihood and Frisch–Waugh–Lovell

Definition 2.8: Profile likelihood

Suppose that $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$, and define

$$\mathcal{L}_{\mathsf{p}}(\boldsymbol{\theta}_1) = \mathcal{L}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2^\star(\boldsymbol{\theta}_1)) = \sup_{\boldsymbol{\theta}_2} \{\mathcal{L}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)\}$$

$\widehat{\boldsymbol{\theta}}_1^\star \in \text{argmax}\{\mathcal{L}_{\mathsf{p}}(\boldsymbol{\theta}_1)\}$ is called a maximum profile likelihood estimator of $\boldsymbol{\theta}_1$.

Correlated Features, Profile Likelihood and Frisch–Waugh–Lovell

Suppose that $\beta = (\beta_1, \beta_2)$, associated to design matrix $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2)$.

$$\beta_2^\star(\beta_1) = \left(\boldsymbol{X}_2^\top \boldsymbol{X}_2\right)^{-1} \boldsymbol{X}_2^\top (\boldsymbol{y} - \boldsymbol{X}_1\beta_1).$$

and then, one can prove that

$$\widehat{\beta}_1^\star = \left(\boldsymbol{X}_1^\top (\mathbb{I} - \boldsymbol{\Pi}_{\boldsymbol{X}_2}) \boldsymbol{X}_1\right)^{-1} \boldsymbol{X}_1^\top (\mathbb{I} - \boldsymbol{\Pi}_{\boldsymbol{X}_2}) \boldsymbol{y}$$

where $\boldsymbol{\Pi}_{\boldsymbol{X}_2} = \boldsymbol{X}_2 \left(\boldsymbol{X}_2^\top \boldsymbol{X}_2\right)^{-1} \boldsymbol{X}_2^\top$ is the projection matrix on $\boldsymbol{X}_2$.

Correlated Features, Profile Likelihood and Frisch–Waugh–Lovell

Under-identification is obtained when the true model is $y = \beta_0 + \boldsymbol{x}_1^\top \boldsymbol{\beta}_1 + \boldsymbol{x}_2^\top \boldsymbol{\beta}_2 + \varepsilon$, but we estimate $y = b_0 + \boldsymbol{x}_1^\top \boldsymbol{b}_1 + \eta$.
Maximum likelihood estimator for $\boldsymbol{b}_1$ is

$$\begin{aligned}
\widehat{\boldsymbol{b}}_1 &= (\boldsymbol{X}_1^\top \boldsymbol{X}_1)^{-1} \boldsymbol{X}_1^\top \boldsymbol{y} \\
&= (\boldsymbol{X}_1^\top \boldsymbol{X}_1)^{-1} \boldsymbol{X}_1^\top [\boldsymbol{X}_{1,i}\boldsymbol{\beta}_1 + \boldsymbol{X}_{2,i}\boldsymbol{\beta}_2 + \varepsilon] \\
&= \beta_1 + \underbrace{(\boldsymbol{X}_1'\boldsymbol{X}_1)^{-1} \boldsymbol{X}_1^\top \boldsymbol{X}_2 \boldsymbol{\beta}_2}_{\beta_{12}} + \underbrace{(\boldsymbol{X}_1^\top \boldsymbol{X}_1)^{-1} \boldsymbol{X}_1^\top \varepsilon}_{\nu_i}
\end{aligned}$$

so that $\mathbb{E}[\widehat{\boldsymbol{b}}_1] = \beta_1 + \beta_{12}$,
and the bias is null when $\boldsymbol{X}_1^\top \boldsymbol{X}_2 = \boldsymbol{0}$ i.e. $\boldsymbol{X}_1 \perp \boldsymbol{X}_2$.

Correlated Features, Profile Likelihood and Frisch–Waugh–Lovell

Assume that the true model is

$$\boldsymbol{y} = \beta_0 + \boldsymbol{X}_1\beta_1 + \boldsymbol{X}_2\beta_2 + \varepsilon$$

If $\boldsymbol{X}_1^\top \boldsymbol{X}_2 = \boldsymbol{0}$, $\widehat{\beta}_2$ can be estimated using

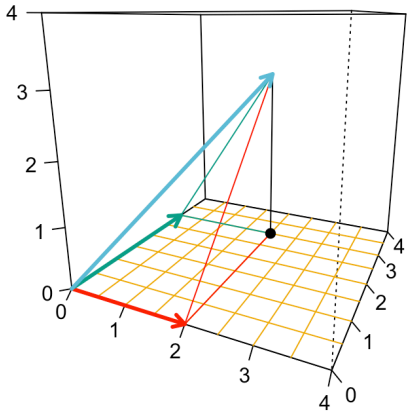
$$(b_0, \widehat{\beta}_2) = (\boldsymbol{X}_2'^\top \boldsymbol{X}_2')^{-1} \boldsymbol{X}_2'^\top \boldsymbol{y} \text{ where } \boldsymbol{X}_2' = [\boldsymbol{1}|\boldsymbol{X}_2]$$

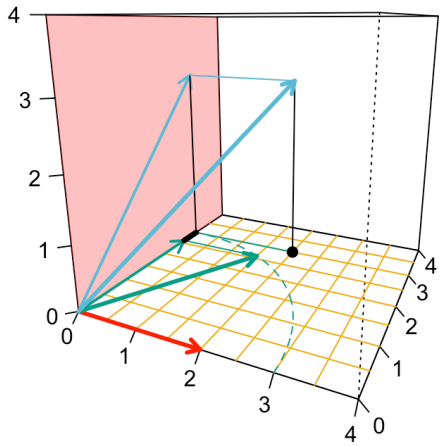
Otherwise, let $\boldsymbol{y}_2^\star = \Pi_{\mathcal{X}_1^\perp} \boldsymbol{y}$ and $\boldsymbol{X}_2^\star = \Pi_{\mathcal{X}_1^\perp} \boldsymbol{X}_2$, then

$$\widehat{\beta}_2 = [\boldsymbol{X}_2^{\star\top} \boldsymbol{X}_2^\star]^{-1} \boldsymbol{X}_2^{\star -1} \boldsymbol{y}_2^\star$$

$\boldsymbol{X}_2^\star = \boldsymbol{X}_2$ if $\boldsymbol{X}_1 \perp \boldsymbol{X}_2$,

Correlated Features, Profile Likelihood and Frisch–Waugh–Lovell





Binary Classifier, Logistic Regression

> In statistics, A logistic model (or logit model) is a statistical model that models the log-odds of an event as a linear combination of one or more independent variables. $\mathbb{W}$

$\{(y_i, \boldsymbol{x}_i)\}$ sample, realizations of i.i.d. vectors $(Y_i, \boldsymbol{X}_i)$ where $(Y|\boldsymbol{X}=\boldsymbol{x}) \sim \mathcal{B}(p(\boldsymbol{x}))$,

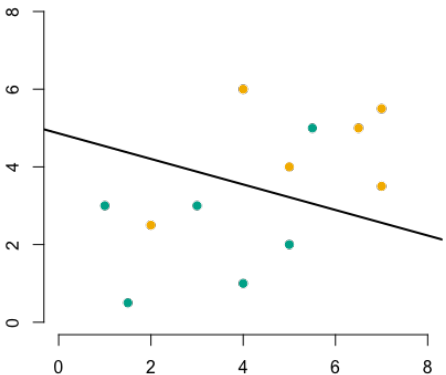
$$\text{where } \mathbb{P}[Y=1|\boldsymbol{X}=\boldsymbol{x}] = p(\boldsymbol{x}) = \frac{e^{\boldsymbol{x}^\top\beta}}{1+e^{\boldsymbol{x}^\top\beta}} = s_{\text{logistic}}(\boldsymbol{x})$$

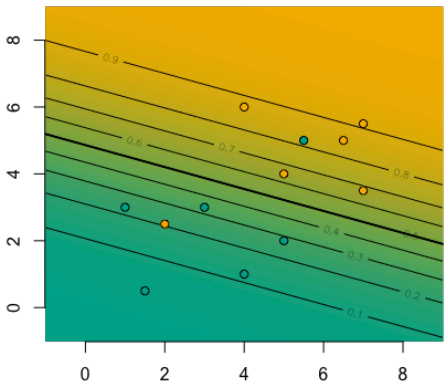
Inference: maximum likelihood

$$\log \mathcal{L}(\beta) = \sum_{i=1}^{n} y_i \log p_i + (1-y_i)\log(1-p_i), \text{ where } p_i = g^{-1}(\boldsymbol{x}^\top\beta) = \frac{\exp[\boldsymbol{x}^\top\beta]}{1+\exp[\boldsymbol{x}^\top\beta]}$$

Classical first order condition, $\nabla \log \mathcal{L}(\widehat{\beta}) = \boldsymbol{0}$ is $\boldsymbol{X}^\top(\boldsymbol{y}-\widehat{\boldsymbol{p}}) = \boldsymbol{0}$ where $\widehat{\boldsymbol{p}} = g^{-1}(\boldsymbol{X}\widehat{\beta})$.

Binary Classifier, Logistic Regression





Binary Classifier, Linear Discriminant Analysis

$\{(y_i, \boldsymbol{x}_i)\}$ sample, realizations of i.i.d. vectors $(Y_i, \boldsymbol{X}_i)$ where $(\boldsymbol{X}|Y=y) \sim \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma})$.

$$\mathbb{P}[Y_i = y|\boldsymbol{X}_i = \boldsymbol{x}] = \frac{f(\boldsymbol{x}|y)\cdot\pi(y)}{f(\boldsymbol{x})}.$$

$$\log\left[\frac{\mathbb{P}[Y_i=1|\boldsymbol{X}_i=\boldsymbol{x}]}{\mathbb{P}[Y_i=0|\boldsymbol{X}_i=\boldsymbol{x}]}\right] = \log\left[\frac{f(\boldsymbol{x}|1)\cdot\pi(1)}{f(\boldsymbol{x}|0)\cdot\pi(0)}\right]$$

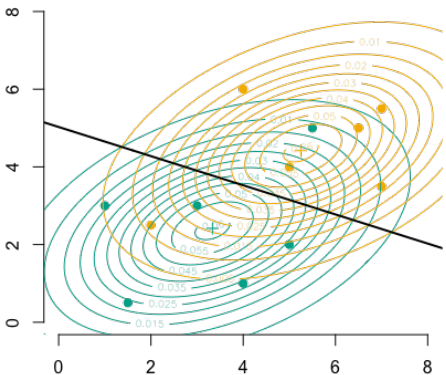
$$= \boldsymbol{x}^\top \underbrace{\boldsymbol{\Sigma}^{-1}[\boldsymbol{\mu}_1-\boldsymbol{\mu}_0]}_{\beta} + \underbrace{\left(-\frac{1}{2}[\boldsymbol{\mu}_1-\boldsymbol{\mu}_0]^\top\boldsymbol{\Sigma}^{-1}[\boldsymbol{\mu}_1-\boldsymbol{\mu}_0] + \log\frac{\pi(1)}{\pi(0)}\right)}_{\text{constant } \beta_0}$$

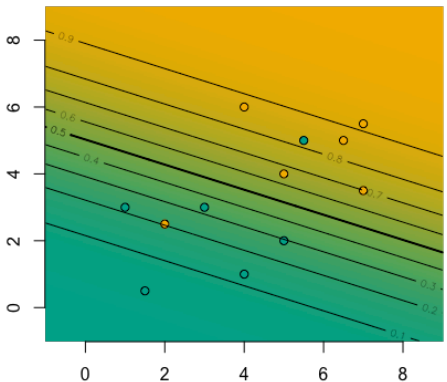
that is linear. And

$$\mathbb{P}[Y_i=1|\boldsymbol{X}_i=\boldsymbol{x}] = \frac{e^{\boldsymbol{x}^\top\boldsymbol{\beta}+\beta_0}}{1+e^{\boldsymbol{x}^\top\boldsymbol{\beta}+\beta_0}} = s_{\text{LDA}}(\boldsymbol{x})$$

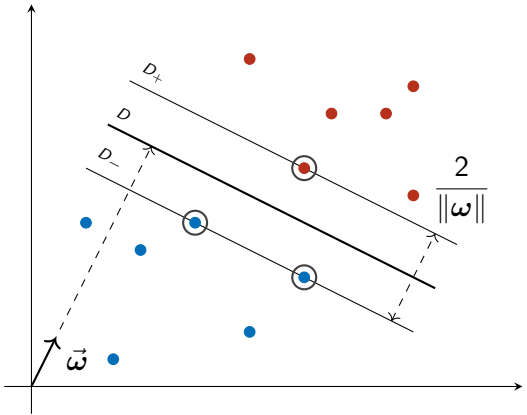
where $\boldsymbol{\mu}_0, \boldsymbol{\mu}_1$ and $\boldsymbol{\Sigma}$ are estimated from the data (and $\boldsymbol{\Sigma}$ should be the same in the two groups).

Binary Classifier, Linear Discriminant Analysis





Binary Classifier, SVM (Support Vector Machine)



$\{(y_i, \boldsymbol{x}_i)\}$ collection of observations, where $y_i \in \{-1, +1\}$.
Consider some separating hyperplane $D = \{\boldsymbol{x} : b + \boldsymbol{x}^\top \boldsymbol{\omega} = 0\}$.
Parameters b and $\boldsymbol{\omega}$ are solution of

$$\underset{b,\boldsymbol{\omega}}{\text{argmax}} \Big\{ \min_i \{||\boldsymbol{x} - \boldsymbol{x}_i|| : x \in \mathbb{R}^n, \text{ s.t. } b + \boldsymbol{x}^\top \boldsymbol{\omega} = 0\} \Big\}$$

Binary Classifier, SVM (Support Vector Machine)

$$\underset{b,\omega}{\text{argmax}}\left\{ \min_i \{||\boldsymbol{x}-\boldsymbol{x}_i|| : x \in \mathbb{R}^n, \text{ s.t. } b+\boldsymbol{x}^\top\boldsymbol{\omega}=0\}\right\}$$

If $m(\boldsymbol{x}) = b+\boldsymbol{x}^\top\boldsymbol{\omega}$, the classifier is $\widehat{y} = \text{sign}(m(\boldsymbol{x})) = \begin{cases} +1 \text{ if } m(\boldsymbol{x}) > 0 \\ -1 \text{ if } m(\boldsymbol{x}) < 0 \end{cases}$

All points are properly classified if $y_i \cdot m(\boldsymbol{x}) > 0$, $\forall i$

Binary Classifier, SVM (Support Vector Machine)

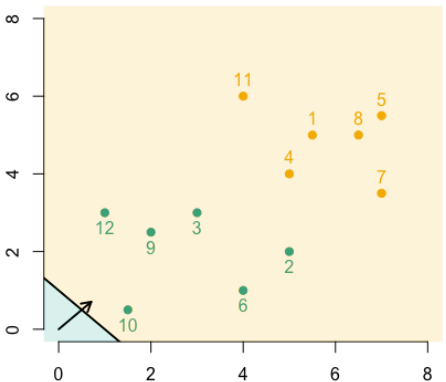
- **Perceptron problem**

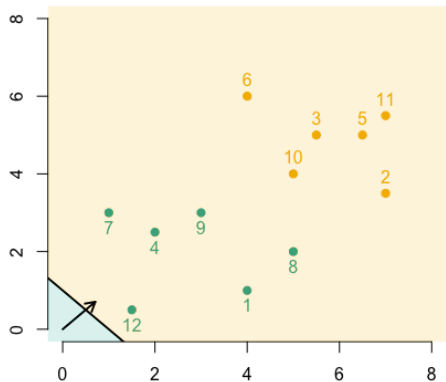
Algorithm 1: Perceptron

initialization : $\boldsymbol{\omega} \leftarrow \mathbf{0}$; $\gamma \leftarrow 0.1$ (learning rate);
for *t=1,2,...* **do**
  for $i = 1, 2, ..., n$ **do**
    if i missclassified, $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \gamma \cdot \text{sign}(m(\boldsymbol{x}_i)) \cdot \boldsymbol{x}_i$
until all the data is correctly classified

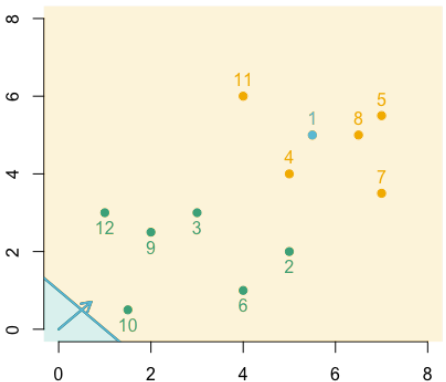
We cycle though the data points (each cycle is coined "**epoch**")
If the data is linearly separable, then the algorithm will converge (but convergence can be slow)

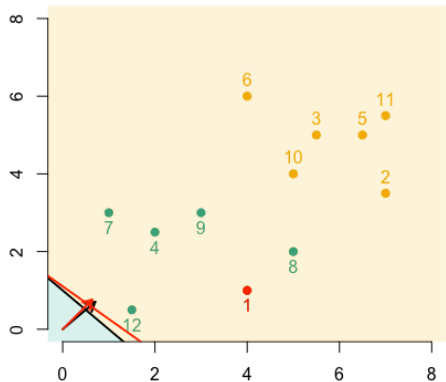
Binary Classifier, SVM (Support Vector Machine)



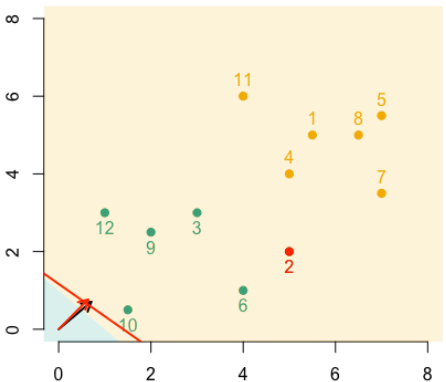


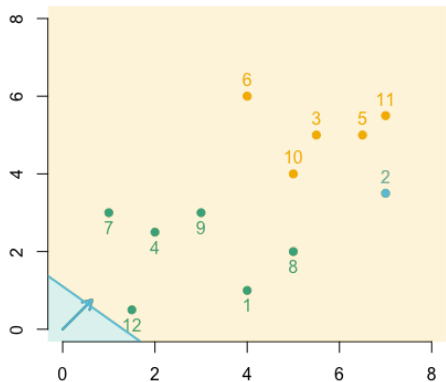
Binary Classifier, SVM (Support Vector Machine)



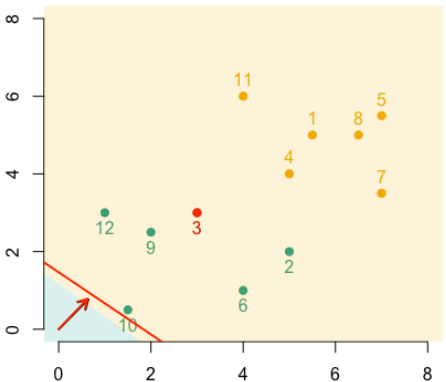


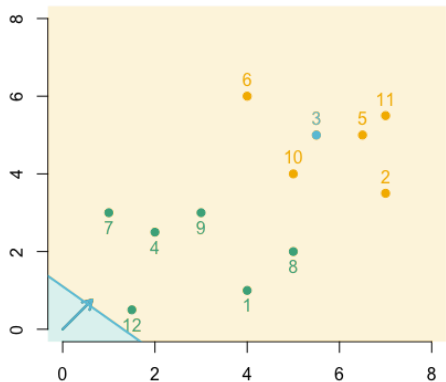
Binary Classifier, SVM (Support Vector Machine)



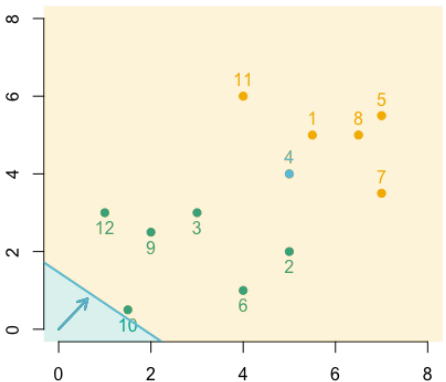


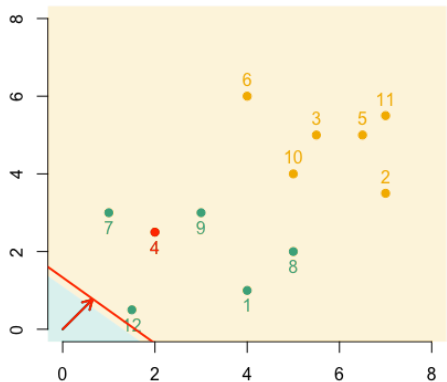
Binary Classifier, SVM (Support Vector Machine)



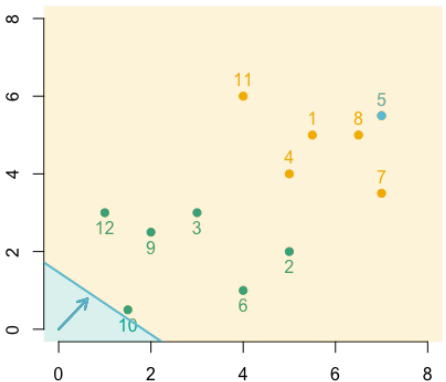


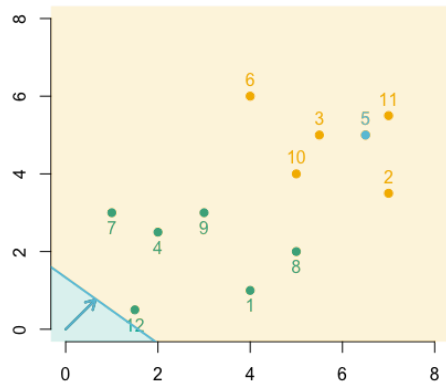
Binary Classifier, SVM (Support Vector Machine)



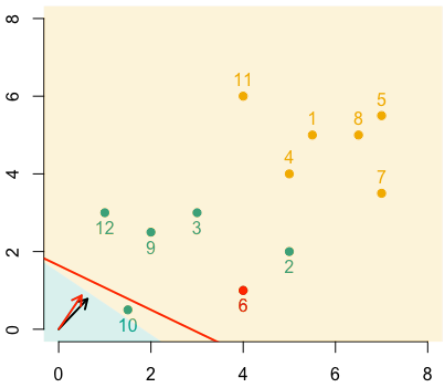


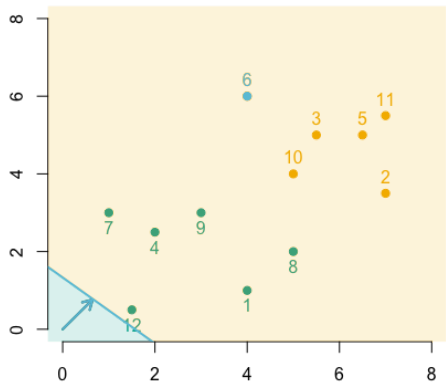
Binary Classifier, SVM (Support Vector Machine)



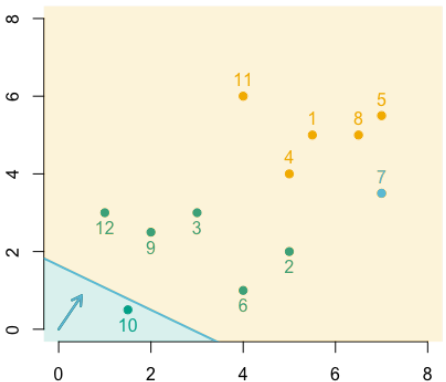


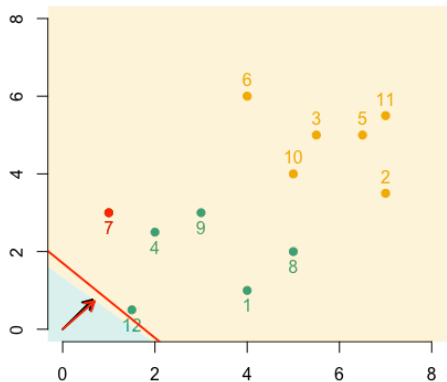
Binary Classifier, SVM (Support Vector Machine)



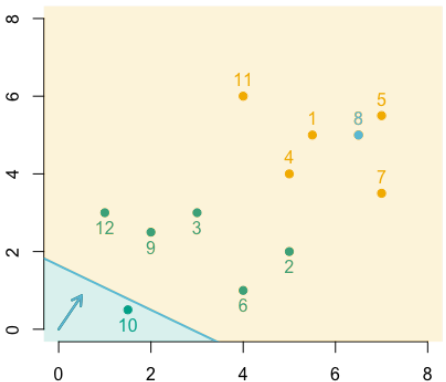


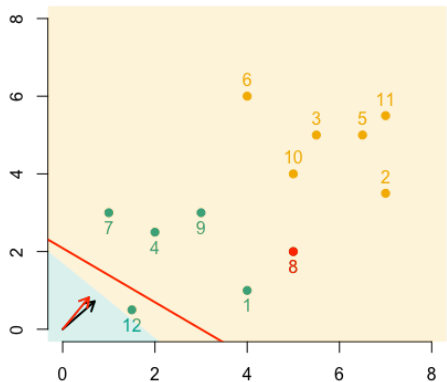
Binary Classifier, SVM (Support Vector Machine)



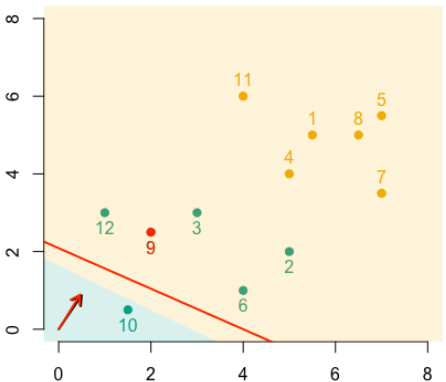


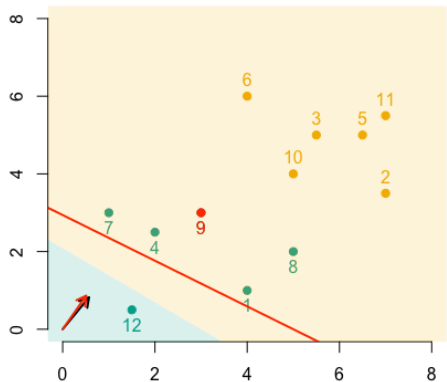
Binary Classifier, SVM (Support Vector Machine)



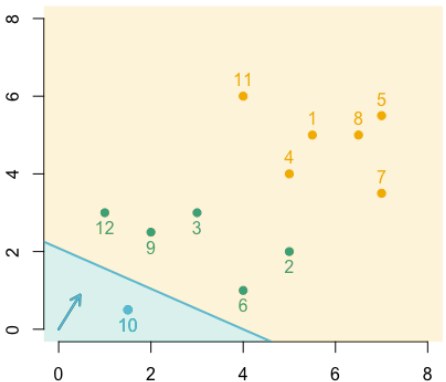


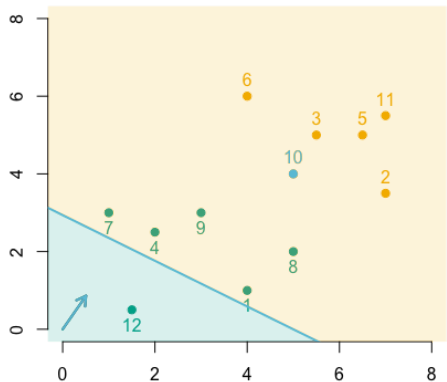
Binary Classifier, SVM (Support Vector Machine)



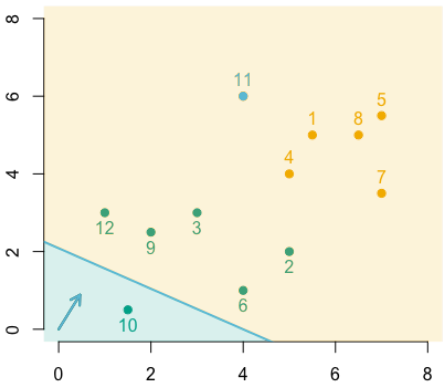


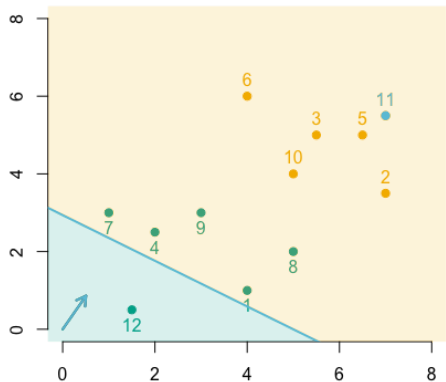
Binary Classifier, SVM (Support Vector Machine)



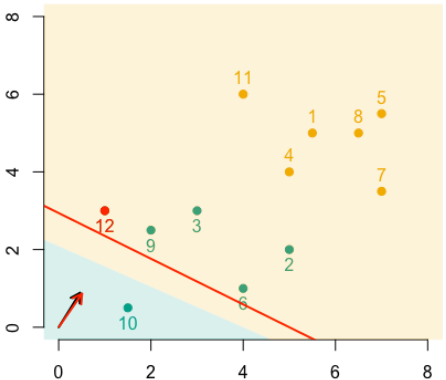


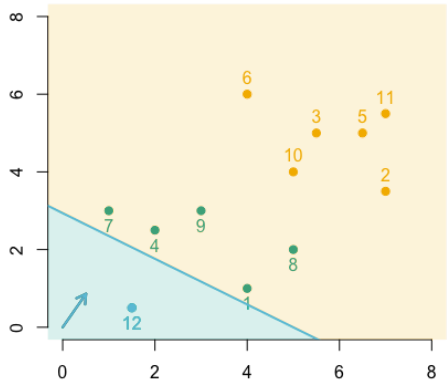
Binary Classifier, SVM (Support Vector Machine)



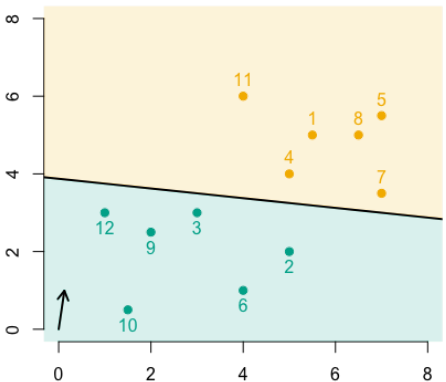


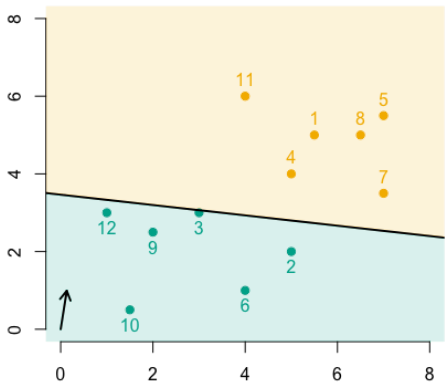
Binary Classifier, SVM (Support Vector Machine)



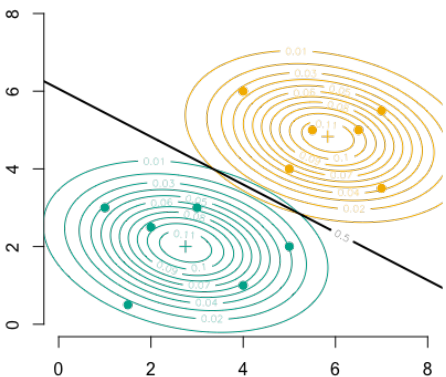


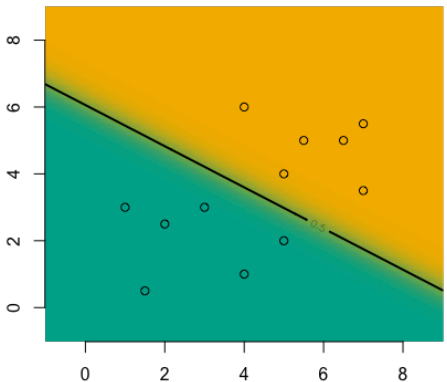
Binary Classifier, SVM (Support Vector Machine)





Binary Classifier, SVM (Support Vector Machine)





Binary Classifier, SVM (Support Vector Machine)

$$\underset{b,\omega}{\text{argmax}}\Big\{ \min_i \{||\boldsymbol{x}-\boldsymbol{x}_i|| : x \in \mathbb{R}^n, \text{ s.t. } b+\boldsymbol{x}^\top\boldsymbol{\omega}=0\}\Big\}$$

- **Primal problem**

$$\underset{b,\omega}{\text{argmax}}\Big\{ \frac{2}{\|\boldsymbol{\omega}\|}, \text{ s.t. } \boldsymbol{y}\cdot(\boldsymbol{X}\boldsymbol{\omega}+b)\geq \mathbf{1}\Big\}$$

This is an optimization problem subject to n linear constraints.
$b+\boldsymbol{x}^\top\boldsymbol{\omega}=\pm 1$ are the **support** lines

$$\underset{b,\omega}{\text{argmin}}\Big\{ \|\boldsymbol{\omega}\|^2, \text{ s.t. } \boldsymbol{y}\cdot(\boldsymbol{X}\boldsymbol{\omega}+b)\geq \mathbf{1}\Big\}$$

which is an quadratic optimization problem subject to n linear constraints

Binary Classifier, SVM (Support Vector Machine)

$$\underset{b,\omega}{\text{argmax}}\Big\{ \min_i \{||\boldsymbol{x}-\boldsymbol{x}_i|| : x \in \mathbb{R}^n, \text{ s.t. } b+\boldsymbol{x}^\top\boldsymbol{\omega}=0\}\Big\}$$

- **Dual problem**

Consider the following "linear classifier"

$$m(\boldsymbol{x}) = \sum_{i=1}^n \alpha_i y_i \cdot \boldsymbol{x}_i^\top \boldsymbol{x} + b = \left(\sum_{i=1}^n \alpha_i y_i \boldsymbol{x}_i^\top\right) \cdot \boldsymbol{x} + b$$

(see representer theorem)
Platt et al. (1999) (following Vapnik et al. (1998), Wahba et al. (1998) and Wahba (1999)) suggested "fitting a sigmoid after the SVM",

$$\frac{e^{\boldsymbol{x}^\top\omega+b}}{1+e^{\boldsymbol{x}^\top\omega+b}} = s_{\text{SVM}}(\boldsymbol{x})$$

Binary Classifier, SVM (Support Vector Machine)

$$\underset{b,\omega}{\text{argmin}}\left\{\|\omega\|^2,\ \text{s.t. } \boldsymbol{y}\cdot(\boldsymbol{X}\beta+b)\geq\mathbf{1}\right\}$$

- **Soft margin problem**

$$\underset{b,\omega,\boldsymbol{\xi}\in\mathbb{R}^n_+}{\text{argmin}}\left\{\|\omega\|^2+\gamma\cdot\mathbf{1}^\top\boldsymbol{\xi},\ \text{s.t. } \boldsymbol{y}\cdot(\boldsymbol{X}\beta+b)\geq\mathbf{1}-\boldsymbol{\xi}\right\}$$

ξ_i's are called "**slack variables**," used when points are not "**linearly separable**".

$$\begin{cases}\xi_i\in(0,1) : \text{ point is between margin and correct side of hyper- plan (margin violation)}\\ \xi_i>1 : \text{ point is misclassified}\end{cases}$$

γ is a regularization parameter:

$$\begin{cases}\text{small } \gamma \text{ allows constraints to be easily ignored, i.e., large margin}\\ \text{large } \gamma \text{ makes constraints hard to ignore, i.e., narrow margin}\end{cases}$$

Binary Classifier, SVM (Support Vector Machine)

The optimization problem was

$$\underset{b,\boldsymbol{\omega},\boldsymbol{\xi}\in\mathbb{R}^n_+}{\operatorname{argmin}} \left\{ \|\boldsymbol{\omega}\|^2 + \gamma \cdot \mathbf{1}^\top \boldsymbol{\xi},\ \text{s.t. } \boldsymbol{y} \cdot (\boldsymbol{X}\boldsymbol{\omega} + b) \geq \mathbf{1} - \boldsymbol{\xi} \right\}$$

with n (linear) constraints, $y_i \cdot m(\boldsymbol{x}_i) \geq 1 - \xi_i$. Since $\xi_i \in \mathbb{R}_+$, satuared constraints becomes $\xi_i = \max\{0, 1 - y_i \cdot m(\boldsymbol{x}_i)\}$, and the optimization problem

$$\underset{b,\boldsymbol{\omega}\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ \|\boldsymbol{\omega}\|^2 + \gamma \cdot \mathbf{1}^\top \max\{\mathbf{0}, 1 - \boldsymbol{y} \cdot (\boldsymbol{X}\boldsymbol{\omega} + b)\} \right\}$$

$$\text{regularization } \sum_{j=1}^{p} \omega_j^2 \qquad \text{loss } \sum_{i=1}^{n} \max\{0, 1 - y_i \cdot (\boldsymbol{x}_i^\top \boldsymbol{\omega} + b)\}$$

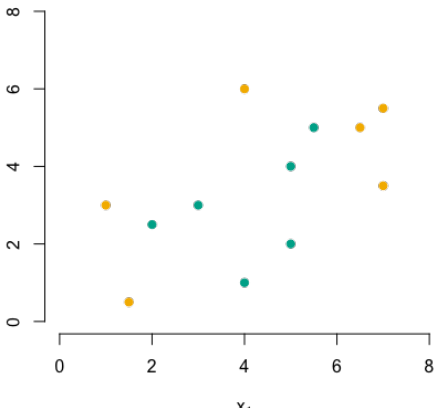
$$\underset{b,\boldsymbol{\omega}\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ \|\boldsymbol{\omega}\|^2 + \gamma \cdot \mathbf{1}^\top \max\{\mathbf{0}, 1 - \boldsymbol{y} \cdot (\boldsymbol{X}\boldsymbol{\omega} + b)\} \right\}$$

Binary Classifier, SVM (Support Vector Machine)

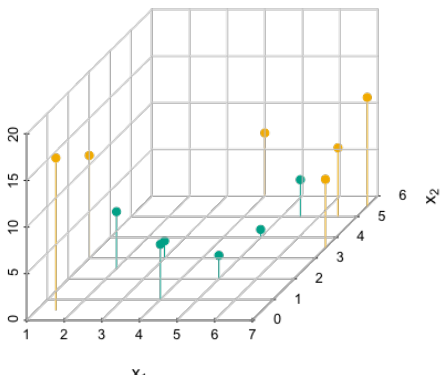
$\ell(y, \widehat{y}) = \max\{0, 1 - y \cdot \widehat{y}\}$ is called "**Hinge loss**", Vito et al. (2004).

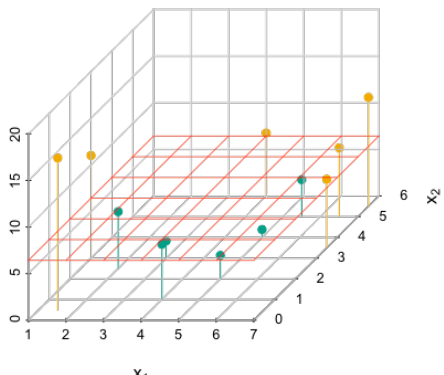
> **Hinge loss**
> In machine learning, the hinge loss is a loss function used for training classifiers. The hinge loss is used for 'maximum-margin' classification, most notably for support vector machines (SVMs). For an intended output $y = \pm 1$ and a classifier score s, the hinge loss of the prediction y is defined as $\ell(y, s) = \max\{0, 1 - s \cdot y\}$.
> W

Kernel trick



Kernel trick





Kernel trick

Heuristically, the dual problem was

$$m(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i \cdot \boldsymbol{x}_i^\top \boldsymbol{x} + b$$

linear kernel

$$m(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i \cdot K(\boldsymbol{x}_i, \boldsymbol{x}) + b$$

$$= \varphi(\boldsymbol{x}_i)^\top \varphi(\boldsymbol{x})$$

Kernel trick

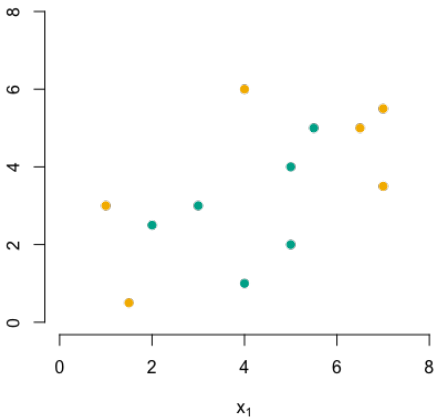
For all $\boldsymbol{x}$ and $\boldsymbol{x}'$ in the input space $\mathcal{X}$, certain functions $k(\boldsymbol{x}, \boldsymbol{x}')$ can be expressed as an inner product in another space $\mathcal{V}$. The function $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is often referred to as a **kernel** or a kernel function. W

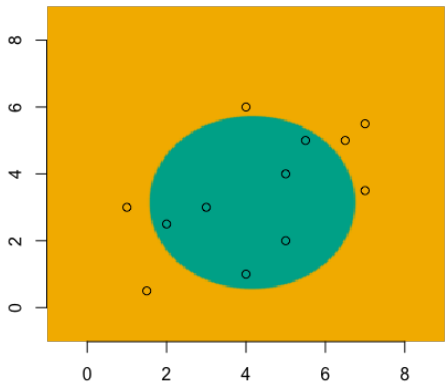
Kernel trick

E.g. for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$

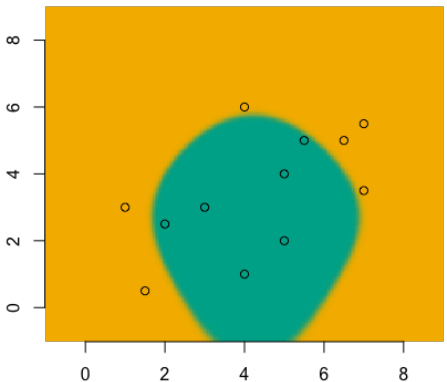
$$\begin{cases} \text{Linear kernel : } k(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^\top \boldsymbol{y}, \\ \text{Polynomial kernel : } k(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^\top \boldsymbol{y} + r)^p, r \geq 0, p \geq 1 \\ \text{Gaussian kernel (RBF) : } k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\dfrac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2\sigma^2}\right), \sigma > 0 \end{cases}$$

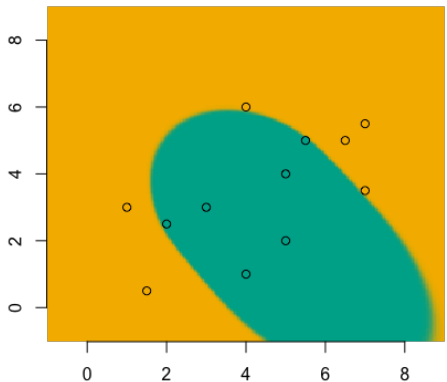
Kernel trick





Kernel trick





Performance Metrics

$\widehat{y}$ \ y	Positive	Negative
Positive	true positive	false positive
Negative	false negative	true negative

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

$$\text{sensitivity} \ / \ \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F1 score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$$

Accuracy

Consider the case where $y \in \{0, 1\}$, and a score $m(\boldsymbol{x})$ (classically in $[0, 1]$).

E.g., for a logistic regression, $m(\boldsymbol{x}) = \dfrac{\exp[\boldsymbol{x}^\top \beta]}{1 + \exp[\boldsymbol{x}^\top \beta]}$.

> **Receiver operating characteristic**
> A receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the performance of a binary classifier model (can be used for multi class classification as well) at varying threshold values. The true-positive rate is also known as sensitivity, recall or probability of detection. The false-positive rate is also known as the probability of false alarm and equals (1 - specificity). W

Accuracy

Definition 2.9: ROC curve

The ROC curve is the parametric curve

$$\{\mathbb{P}[m(\boldsymbol{X}) > t|Y = 0], \mathbb{P}[m(\boldsymbol{X}) > t|Y = 1]\} \text{ for } t \in [0, 1],$$

when the score $m(\boldsymbol{X})$ and Y evolve in the same direction (a high score indicates a high risk).

$$C(t) = \text{TPR} \circ \text{FPR}^{-1}(t),$$

where

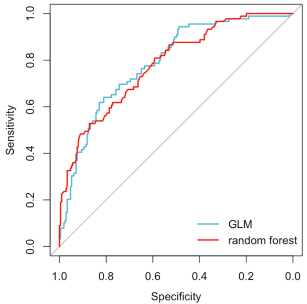
$$\begin{cases} \text{FRP}(t) = \mathbb{P}[m(\boldsymbol{X}) > t|Y = 0] = \mathbb{P}[m_0(\boldsymbol{X}) > t] \\ \text{TPR}(t) = \mathbb{P}[m(\boldsymbol{X}) > t|Y = 1] = \mathbb{P}[m_1(\boldsymbol{X}) > t]. \end{cases}$$

Accuracy

```
> library(ROCR)
> pred = prediction(df$yhat, df$y)
> roc = performance(pred,"tpr","fpr")
> plot(roc)
> auc = performance( pred,"auc")
```

see also

```
> library(pROC)
```

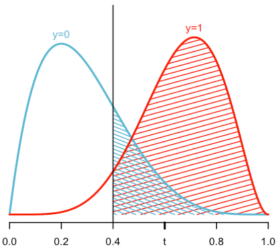


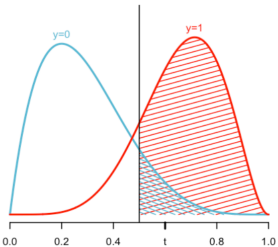
Definition 2.10: AUC, area under the ROC curve

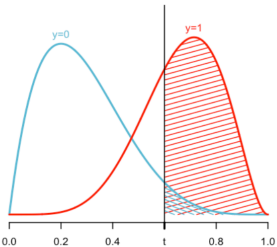
The area under the curve is defined as the area below the ROC curve,

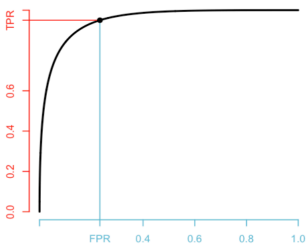
$$\text{AUC} = \int_0^1 C(t)\mathrm{d}t = \int_0^1 \text{TPR} \circ \text{FPR}^{-1}(t)\mathrm{d}t.$$

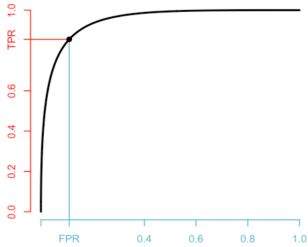
Accuracy

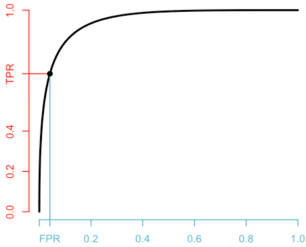

y=0
y=1
0.0
0.2
0.4
t
0.8
1.0


y=0
y=1
0.0
0.2
0.4
t
0.8
1.0


y=0
y=1
0.0
0.2
0.4
t
0.8
1.0

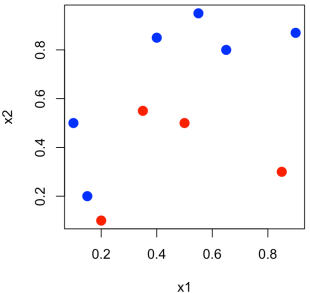

TPR
0.0
0.2
0.4
0.6
FPR
0.4
0.6
0.8
1.0


TPR
0.0
0.2
0.4
0.6
1.0
FPR
0.4
0.6
0.8
1.0


TPR
0.0
0.2
0.4
0.6
1.0
FPR
0.2
0.4
0.6
0.8
1.0

ROC Curve

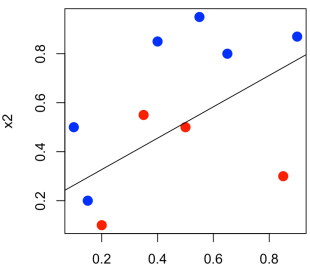
```
> x1 = c(.4,.55,.65,.9,.1,.35,.5,.15,.2,.85)
> x2 = c(.85,.95,.8,.87,.5,.55,.5,.2,.1,.3)
> y = c(1,1,1,1,1,0,0,1,0,0)
> df = data.frame(x1 = x1, x2 = x2,
                  y = as.factor(y))
> plot(x1,x2,col=1+y)
> reg = glm(y~x1+x2, data=df,
            family=binomial(link = "logit"))
> b = coefficients(reg)
> abline(a=-b[1]/b[3],b=-b[2]/b[3])
```



$\mathbb{P}[Y = 1|\boldsymbol{X} = \boldsymbol{x}] = \mathbb{P}[Y = 0|\boldsymbol{X} = \boldsymbol{x}]$ si

$$e^{\boldsymbol{x}^\top \boldsymbol{\beta}} = 1 \text{ ou } \boldsymbol{x}^\top \boldsymbol{\beta} = 0$$

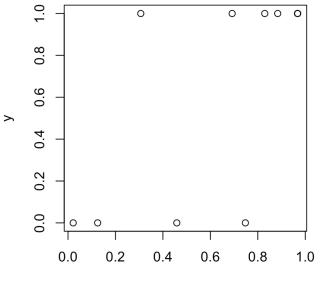
soit ici $\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$.



ROC Curve

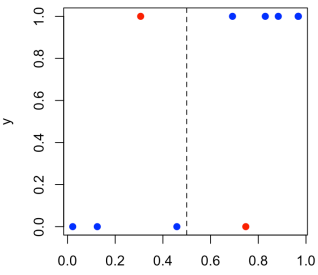
Let Sdenote the predicted score (i.e. $\widehat{m}(\boldsymbol{x}_i)$)

```
> Y = df$y
> S = predict(reg, type="response")
> plot(S,y)
> threshold = .5
> Yhat = (S>threshold)*1
> plot(S,y,col=1+(y==Yhat))
> abline(v=threshold,lty=2)
```



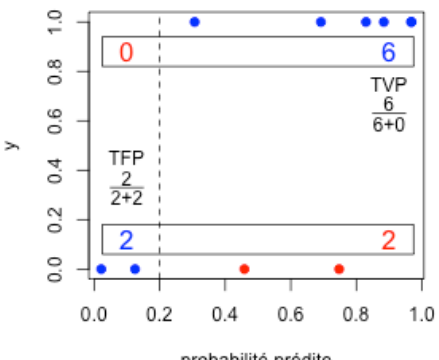
$$\widehat{y}_i = \begin{cases} 1 & \text{if } \widehat{m}(\boldsymbol{x}_i) > \text{threshold} \\ 0 & \text{if } \widehat{m}(\boldsymbol{x}_i) \leq \text{threshold} \end{cases}$$

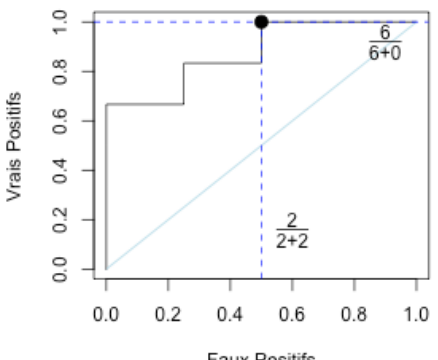
```
> table(Yhat,Y)
    Y
Yhat 0 1
   0 3 1
   1 1 5
```



ROC Curve

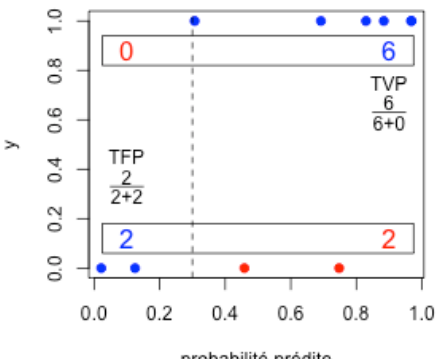
$$FPR = \frac{\mathbb{P}[y = 0, \widehat{y} = 1]}{\mathbb{P}[y = 0]} \text{ and } TPR = \frac{\mathbb{P}[y = 1, \widehat{y} = 1]}{\mathbb{P}[y = 1]}$$

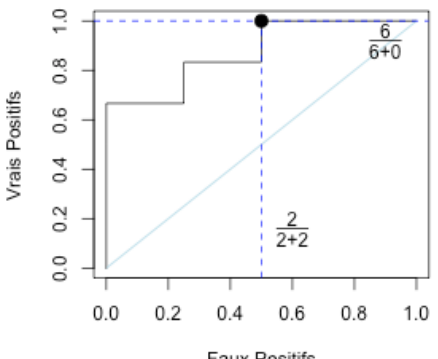




ROC Curve

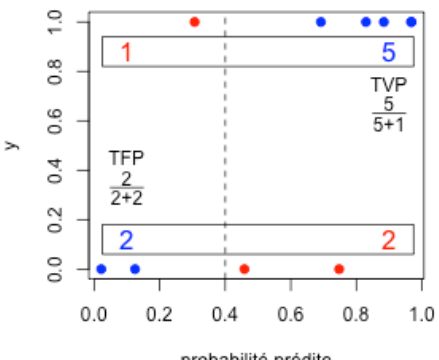
$$FPR = \frac{\mathbb{P}[y = 0, \widehat{y} = 1]}{\mathbb{P}[y = 0]} \text{ and } TPR = \frac{\mathbb{P}[y = 1, \widehat{y} = 1]}{\mathbb{P}[y = 1]}$$

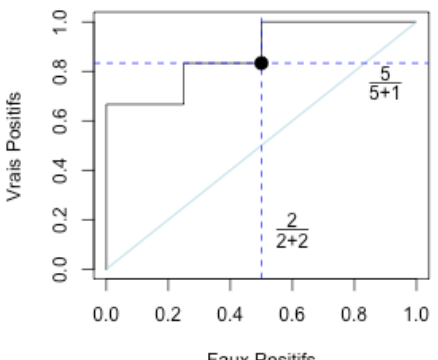




ROC Curve

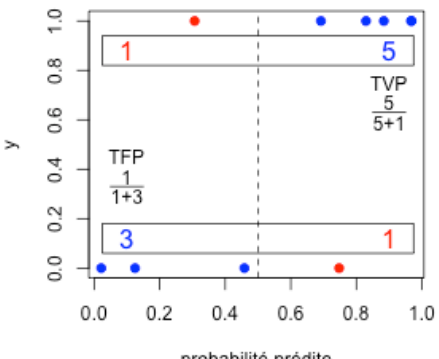
$$FPR = \frac{\mathbb{P}[y = 0, \widehat{y} = 1]}{\mathbb{P}[y = 0]} \text{ and } TPR = \frac{\mathbb{P}[y = 1, \widehat{y} = 1]}{\mathbb{P}[y = 1]}$$

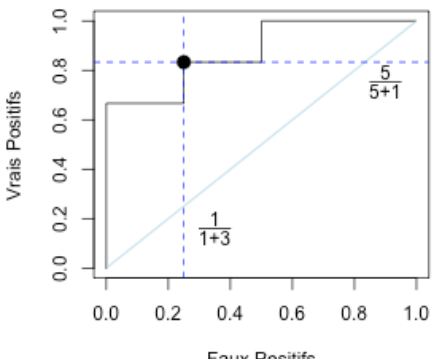




ROC Curve

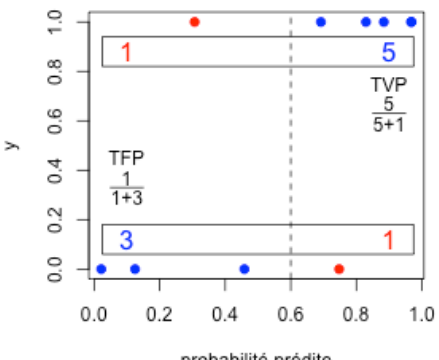
$$FPR = \frac{\mathbb{P}[y = 0, \widehat{y} = 1]}{\mathbb{P}[y = 0]} \text{ and } TPR = \frac{\mathbb{P}[y = 1, \widehat{y} = 1]}{\mathbb{P}[y = 1]}$$

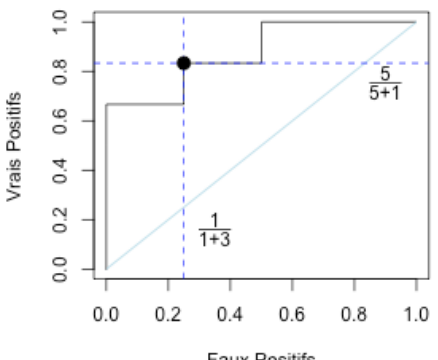




ROC Curve

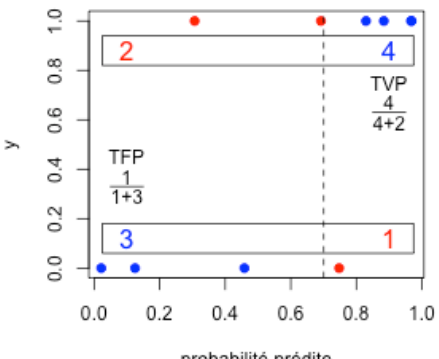
$$FPR = \frac{\mathbb{P}[y = 0, \widehat{y} = 1]}{\mathbb{P}[y = 0]} \text{ and } TPR = \frac{\mathbb{P}[y = 1, \widehat{y} = 1]}{\mathbb{P}[y = 1]}$$

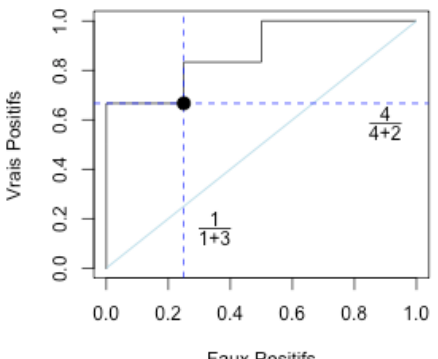




ROC Curve

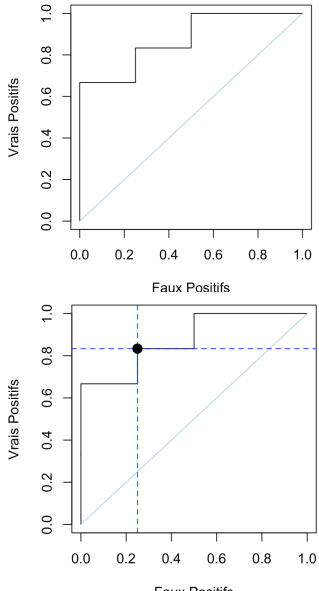
$$FPR = \frac{\mathbb{P}[y = 0, \widehat{y} = 1]}{\mathbb{P}[y = 0]} \text{ and } TPR = \frac{\mathbb{P}[y = 1, \widehat{y} = 1]}{\mathbb{P}[y = 1]}$$





ROC Curve

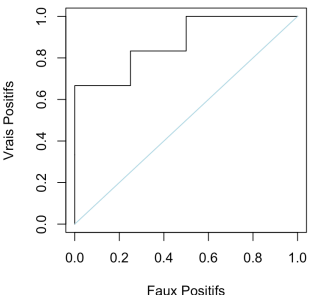
```
> roc.curve=function(s){
  Ps = (S>s)*1
  FP = sum((Ps==1)*(Y==0))/sum(Y==0)
  TP = sum((Ps==1)*(Y==1))/sum(Y==1)
  vect = c(FP,TP)
  names(vect) = c("FPR","TPR")
  return(vect) }
> u = seq(0,1,length=251)
> V = Vectorize(roc.curve)(u)
> plot(t(V), type="s")
> table(Yhat,Y)
     Y
Yhat 0 1
   0 3 1
   1 1 5
> sum((Yhat)*(Y==0))/sum(Y==0)
[1] 0.25
> sum((Yhat==1)*(Y==1))/sum(Y==1)
[1] 0.8333333
```



ROC Curve

Various R packages could be used `ROCR` (`pROC` or `plotROC`)

```
> library(ROCR)
> pred = prediction(S,Y)
> plot(performance(pred,"tpr","fpr"))
> auc.perf = performance(pred, measure = "auc")
> auc.perf@y.values[[1]]
[1] 0.875
```

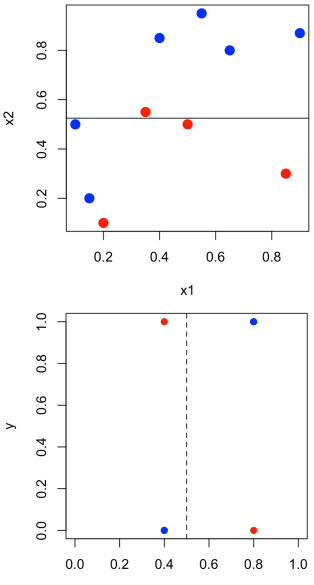


The AUC – Area Under the Curve – is a standard accuracy measure.

ROC Curve

If we consider some tree based model
E.g., regression of y on $\mathbf{1}_{[.525,\infty)}(x_2)$

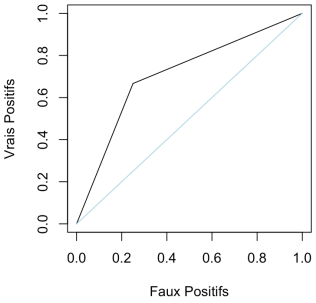
```
> reg = glm(y~I(x2> .525), data=df,
            family=binomial(link = "logit"))
> abline(h=.525)
> Y = df$y
> S = predict(reg,type="response")
> plot(S,y,xlim=0:1)
> threshold = .5
> Yhat  = (S>threshold)*1
> table(Yhat,Y)
     Y
Yhat 0 1
   0 3 2
   1 4 5
```

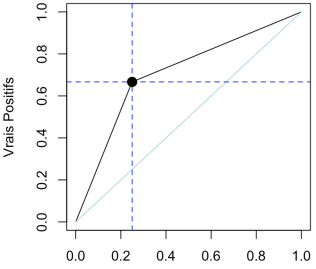


ROC Curve

With classes, the ROC curve is piecewise linear

```
> pred = prediction(S,Y)
> plot(performance(pred,"tpr","fpr"))
> table(Yhat,Y)
     Y
Yhat 0 1
   0 3 2
   1 1 4
> auc.perf = performance(pred, measure = "auc")
> auc.perf@y.values[[1]]
[1] 0.7083333
```





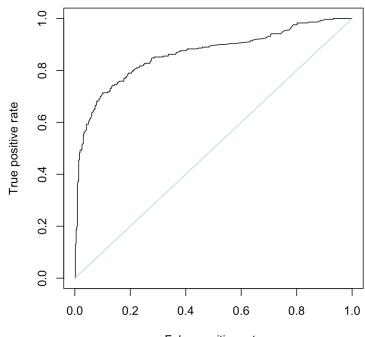
Survie des Passagers du Titanic

y survival for a Titanic passenger,

```
> loc = "http://freakonometrics.free.fr/titanic.RData"
> download.file(loc, "titanic.RData")
> load("titanic.RData")
> base = base[!is.na(base$Age),1:7]
> reg = glm(Survived ~ Sex+poly(Age,3)+Pclass+SibSp,
            family = "binomial", data = base)
```

Consider a logistic regression

```
> library(ROCR)
> Y = base$Survived
> S = predict(reg,type="response")
> pred = prediction(S,Y)
> plot(performance(pred,"tpr","fpr"))
> performance(pred, measure = "auc")
  @y.values[[1]]
[1] 0.8627358
```



Multi-class Classification

$$\mathbb{P}(Y=k \mid \boldsymbol{X}=\boldsymbol{x}) = \frac{e^{\beta_{k,0}+\beta_{k,1}x_1+\beta_{k,2}x_2+\cdots+\beta_{k,p}x_p}}{1+\sum_{l=1}^{K-1} e^{\beta_{k,0}+\beta_{k,1}x_1+\beta_{k,2}x_2+\cdots+\beta_{k,p}x_p}} = \frac{e^{\boldsymbol{x}^\top\boldsymbol{\beta}_k}}{1+\sum_{l=1}^{K-1} e^{\boldsymbol{x}^\top\boldsymbol{\beta}_l}}$$

$\forall k \in \{1,\cdots,K-1\}$, while

$$\mathbb{P}(Y=K \mid \boldsymbol{X}=\boldsymbol{x}) = \frac{1}{1+\sum_{l=1}^{K-1} e^{\beta_{k,0}+\beta_{k,1}x_1+\beta_{k,2}x_2+\cdots+\beta_{k,p}x_p}} = \frac{1}{1+\sum_{l=1}^{K-1} e^{\boldsymbol{x}^\top\boldsymbol{\beta}_l}}$$

And in that case, the predicted class is $\widehat{y} = \max_{k\in\{1,\cdots,K\}} \mathbb{P}(Y=k \mid \boldsymbol{x})$ (majority rule).

Voronoi diagram

A Voronoi diagram is a partition of a plane into regions close to each of a given set of objects. It can be classified also as a tessellation. In the simplest case, these objects are just finitely many points in the plane. For each seed there is a corresponding region, called a Voronoi cell, consisting of all points of the plane closer to that seed than to any other. The Voronoi diagram of a set of points is dual to that set's Delaunay triangulation. W

Multi-class Classification

Simplex

A k-simplex is a k-dimensional polytope that is the convex hull of its $k+1$ vertices. More formally, suppose the $k+1$ points $u_0, \ldots, u_k$ are affinely independent. Then, the simplex determined by them is the set of points

$$C(\boldsymbol{u}) = \left\{ \theta_0 u_0 + \cdots + \theta_k u_k \;\middle|\; \sum_{i=0}^{k} \theta_i = 1 \text{ and } \theta_i \geq 0 \text{ for } i = 0, \ldots, k \right\}.$$

The standard simplex or probability simplex is the $(k-1)$-dimensional simplex whose vertices are the k standard unit vectors in $\mathbb{R}^k$, or in other words

$$\left\{ x \in \mathbb{R}^k : x_0 + \cdots + x_{k-1} = 1, x_i \geq 0 \text{ for } i = 0, \ldots, k-1 \right\}.$$

W

Multi-class Classification

Ternary plot

A ternary plot, ternary graph, triangle plot, simplex plot, or Gibbs triangle is a barycentric plot on three variables which sum to a constant.[1] It graphically depicts the ratios of the three variables as positions in an equilateral triangle. W

Viviani's theorem

Viviani's theorem, named after Vincenzo Viviani, states that the sum of the shortest distances from any interior point to the sides of an equilateral triangle equals the length of the triangle's altitude. W

Metrics related to Classifiers

Inputs, $\{y_i, \widehat{y}_i\}$, where $y_i, \widehat{y}_i \in \{0, 1\}$.

	True Value y	**Predicted Value** $\hat{y}$
True Positive (TP)	1	1
True Negative (TN)	0	0
False Positive (FP)	0	1
False Negative (FN)	1	0

	True $y = 0$	**True** $y = 1$
$\hat{y} = 0$	True Negative (TN)	False Negative (FN)
$\hat{y} = 1$	False Positive (FP)	True Positive (TP)

Metrics related to Classifiers

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Heuristics:

- Works well for balanced datasets.
- Can be misleading for imbalanced datasets.

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}$$

Heuristics:

- Precision: Useful when false positives are costly.
- Recall: Useful when false negatives are costly.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{1}$$

Heuristics:

Metrics related to Classifiers

- Harmonic mean of precision and recall.
- Useful for imbalanced datasets.

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right)$$

```
> library(caret)
> Y    = as.factor(base$Survived)
> Yhat = as.factor((predict(reg,
    type="response") > .5)*1)
> confusionMatrix(Yhat, Y, mode =
    "everything", positive="1")

          Reference
Prediction   0   1
         0 367  75
         1  57 215

               Accuracy : 0.8151
                 95% CI : (0.7847,
      0.843)
```

```
                  Kappa : 0.613

 Mcnemar's Test P-Value : 0.139

            Sensitivity : 0.7414
            Specificity : 0.8656
         Pos Pred Value : 0.7904
         Neg Pred Value : 0.8303
              Precision : 0.7904
                 Recall : 0.7414
                     F1 : 0.7651
             Prevalence : 0.4062
         Detection Rate : 0.3011
   Detection Prevalence : 0.3810
      Balanced Accuracy : 0.8035
```

Metrics related to Classifiers

Note, for the confidence interval,

$$\text{CI} = \left[\text{Accuracy} \pm \Phi^{-1}(1-\alpha/2) \times \sqrt{\frac{\text{Accuracy}(1-\text{Accuracy})}{n}}\right]$$

Inputs, $\{y_i, \widehat{s}(\boldsymbol{x})_i\}$, for some model $\widehat{s}$,

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Heuristics:

- Expressed as a percentage.
- Sensitive to small values.

Exponential Family

Definition 2.11: Exponential family, McCullagh and Nelder (1989)

The distribution of Y is in the exponential family if its density (with respect to some appropriate measure) is

$$f_{\theta,\varphi}(y) = \exp\left(\frac{y\theta - b(\theta)}{\varphi} + c(y,\varphi)\right),$$

where θ is the canonical parameter, φ is a nuisance parameter, and $b : \mathbb{R} \to \mathbb{R}$ is some $\mathbb{R} \to \mathbb{R}$ function.

Such as the binomial, Poisson, Gaussian, gamma distributions, etc.

Also compound Poisson / Tweedie (from Tweedie (1984)).

Exponential Family

Pour une variable aléatoire Y de la famille exponentielle, alors

$$\mathbb{E}(Y) = b'(\theta) \text{ et } \mathrm{Var}(Y) = b''(\theta)\varphi,$$

On définie la fonction variance V par

$$V(\mu) = b''([b']^{-1}(\mu)) \text{ et } \mathrm{Var}(Y) = \varphi \mathrm{V}(\mu)$$

Lien canonique $g = b'^{-1}$

Exponential Family

$$f(y;\theta,\varphi)=\exp\left(\frac{y\,\boldsymbol{\theta}-b(\theta)}{\varphi}+c(y,\varphi)\right),$$

The **Gaussian** distribution $\mathcal{N}(\mu,\sigma^2)$

$$f(y;\mu,\sigma^2)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2\right)=\exp\left(\frac{-y^2+2y\mu-\mu^2}{2\sigma^2}+\log\frac{1}{\sigma\sqrt{2\pi}}\right)$$

canonical parameter θ

$$f(y|\mu,\sigma^2)=\exp\left(\frac{y\ \mu\ -\mu^2/2}{\sigma^2}+-\frac{y^2}{2\sigma^2}\log\frac{1}{\sigma\sqrt{2\pi}}\right)$$

i.e., $\boldsymbol{\theta}=\mu$, $\varphi=\sigma^2$, $b(\theta)=\theta^2/2$, canonical link $g(\mu)=\mu$ ("identity")

Famille Exponentielle

$$f(y|\theta,\varphi)=\exp\left(\frac{y\,\boldsymbol{\theta}-b(\theta)}{\varphi}+c(y,\varphi)\right),$$

Exemple La loi de **Poisson** $\mathcal{P}(\lambda)$

$$f(y|\lambda)=e^{-\lambda}\frac{\lambda^y}{y!}=\exp\left(-\lambda+y\log\lambda-log(y!)\right)$$

$$f(y|\lambda)=\exp\left(\frac{y\log\lambda-\lambda}{1}-\log(y!)\right)$$

soit $\boldsymbol{\theta}=\log\lambda$, $\varphi=1$, $b(\theta)=\lambda=e^{\theta}$, canonical link $g(\mu)=\log(\mu)$ ("log")

Famille Exponentielle

$$f(y|\theta,\varphi) = \exp\left(\frac{y\,\boldsymbol{\theta} - b(\theta)}{\varphi} + c(y,\varphi)\right),$$

Exemple La loi de **Bernoulli** $\mathcal{B}(p)$

$$f(y|\lambda) = p^y(1-p)^{1-y} = \exp\left(y\log p + (1-y)\log(1-p)\right)$$

$$f(y|\lambda) = \exp\left(\frac{y[\log p - \log(1-p)] + \log(1-p)}{1}\right)$$

soit $\boldsymbol{\theta} = \log\dfrac{p}{1-p}$, $\varphi = 1$, $b(\theta) = \log\left(1+e^{\theta}\right)$

Ici $p = \dfrac{e^{\theta}}{1+e^{\theta}}$, canonical link $g(\mu) = \log\dfrac{\mu}{1-\mu}$ ("logit")

Famille Exponentielle

Tweedie (1984) a suggéré la famille suivante

$$f(y|\mu,\varphi) = A(y,\varphi)\cdot\exp\left\{\frac{1}{\varphi}\Big[y\theta(\mu)-\kappa(\theta(\mu))\Big]\right\},$$

où

$$\theta(\mu)=\begin{cases}\dfrac{\mu^{1-\gamma}}{1-\gamma} & \text{si } \gamma\neq 1\\ \log\mu & \text{si } \gamma=1\end{cases} \text{ et } \kappa(\theta(\mu))=\begin{cases}\dfrac{\mu^{2-\gamma}}{2-\gamma} & \text{si } \gamma\neq 2\\ \log\mu & \text{si } \gamma=2\end{cases}$$

La loi de Y est alors une loi Poisson composée à sauts Gamma,

$$Y\sim\mathcal{CP}oi\left(\mu^{2-\gamma}\varphi(2-\gamma),\mathcal{G}\left(-\frac{2-\gamma}{\varphi(1-\gamma)},\varphi(2-\gamma)\mu^{\gamma-1}\right)\right),$$

lorsque $\gamma\in[1,2]$, et $V(\mu)=\mu^{\gamma}$.

Régression GLM

$$f(y_i|\theta_i,\varphi)=\exp\left\{\frac{y_i\theta_i-b(\theta_i)}{a(\varphi)}+c(y_i,\varphi)\right\}$$

La log-vraisemblance est alors

$$\log\mathcal{L}(\boldsymbol{\theta},\varphi|\boldsymbol{y})=\frac{\sum_{i=1}^n y_i\theta_i-\sum_{i=1}^n b(\theta_i)}{a(\varphi)}+\sum_{i=1}^n c(y_i,\varphi).$$

On peut remplacer $a(\varphi)$ par φ/w_i, w_i est un poids individuel.

Hypothèses

- g est suffisemment régulière (au moins $\mathcal{C}^2$
- φ est supposé connu (pour l'instant)
- $n>p=k+1$ et $\boldsymbol{X}$ est de plein rang (et donc $\boldsymbol{X}^\top\boldsymbol{X}$ est définie positive)

Losses

In GLM, the scaled deviance ($-2\times$ the log-likelihood) of the exponential model is

$$D^\star = \sum_{i=1}^{n} d^\star\left(y_i, \widehat{y}_i\right), \text{ where } d^\star\left(y_i, \widehat{y}_i\right) = 2\big(\log \mathcal{L}_i(y_i) - \log \mathcal{L}_i(\widehat{y}_i)\big).$$

that can be related to in-sample empirical risk

$$\widehat{\mathcal{R}}_n(\widehat{m}) = \sum_{i=1}^{n} \ell\left(y_i, \widehat{m}(\boldsymbol{x}_i)\right),$$

For the Poisson distribution (with a log-link), the loss would be

$$\ell(y_i, \widehat{y}_i) = \begin{cases} 2\left(y_i \log y_i - y_i \log \widehat{y}_i - y_i + \widehat{y}_i\right) & y_i > 0 \\ 2\widehat{y}_i & y_i = 0, \end{cases}$$

while for a logistic regression, we have the standard binary cross-entropy loss

$$\ell(y_i, \widehat{y}_i) = -\big(y_i \log[\widehat{y}_i] + (1 - y_i) \log[1 - \widehat{y}_i]\big).$$

Régression GLM

$$\log \mathcal{L}(\boldsymbol{\theta}, \varphi, \boldsymbol{y}) = \sum_{i=1}^{n} \Big[\underbrace{\frac{y_i\theta_i - b(\theta_i)}{\varphi} + c(y_i, \varphi)}_{\log \mathcal{L}_i} \Big].$$

$$\nabla \log \mathcal{L} = \boldsymbol{X}^\top (\boldsymbol{y} - \widehat{\boldsymbol{y}}) = \boldsymbol{X}^\top (\boldsymbol{y} - g^{-1}(\boldsymbol{X}\widehat{\boldsymbol{\beta}})) = \boldsymbol{0}.$$

$$\frac{\partial \log \mathcal{L}_i}{\partial \beta_j} = \frac{\partial \log \mathcal{L}_i}{\partial \theta_i} \cdot \frac{\partial \theta_i}{\partial \mu_i} \cdot \frac{\partial \mu_i}{\partial \eta_i} \cdot \frac{\partial \eta_i}{\partial \beta_j}$$

$$\frac{\partial \log \mathcal{L}_i}{\partial \theta_i} = \frac{y_i - b'(\theta_i)}{\varphi} = \frac{y_i - \mu_i}{\varphi}$$

$$\frac{\partial \theta_i}{\partial \mu_i} = \left(\frac{\partial \mu_i}{\partial \theta_i} \right)^{-1} = \frac{1}{b''(\theta_i)} = \frac{1}{V(\mu_i)} \text{ and } \frac{\partial \eta_i}{\partial \beta_j} = \frac{\partial \boldsymbol{x}_i^\top \boldsymbol{\beta}}{\partial \beta_j} = x_{i,j}$$

Régression GLM

Les conditions du premier ordre s'écrivent

$$\sum_{i=1}^{n} \frac{\partial \log \mathcal{L}_i}{\partial \beta_j} = \sum_{i} \frac{\partial \mu_i}{\partial \eta_i} \times \frac{y_i - \mu_i}{\mathrm{V}(\mu_i)} x_{ij} = 0, \ \forall j$$

Pour simplifier, notons que

$$\frac{\partial \mu_i}{\partial \eta_i} = (\frac{\partial \eta_i}{\partial \mu_i})^{-1} = (g'(\mu_i))^{-1}$$

Les conditions du premier ordre s'écrivent

$$\boldsymbol{X}^\top \boldsymbol{\Omega} (\boldsymbol{y} - \boldsymbol{\mu}) = \boldsymbol{0}$$

$$\boldsymbol{\Omega} = \mathrm{diag}((V(\mu_i) g'(\mu_i))^{-1})$$

mais on utilisera une autre écriture...

Régression GLM

Les conditions du premier ordre s'écrivent

$$\boldsymbol{X}^\top \boldsymbol{W}\boldsymbol{\Delta}(\boldsymbol{y}-\boldsymbol{\mu})=\boldsymbol{0}$$

$$\boldsymbol{W}=\text{diag}((V(\mu_i)g'(\mu_i)^2)^{-1}) \text{ et } \boldsymbol{\Delta}=\text{diag}(g'(\mu_i))$$

Remarque: la matrice d'information de Fisher est $\boldsymbol{X}^\top \boldsymbol{W}\boldsymbol{X}$

Remarque: avec la fonction de lien canonique $\boldsymbol{X}^\top(\boldsymbol{y}-\boldsymbol{\mu})=\boldsymbol{0}$

Régression GLM

$$f(y_i|\theta_i,\varphi)=\exp\left\{\frac{y_i\theta_i-b(\theta_i)}{a(\varphi)}+c(y_i,\varphi)\right\}$$

La log-vraisemblance est alors

$$\log\mathcal{L}(\boldsymbol{\theta},\varphi|\boldsymbol{y})=\frac{\sum_{i=1}^n y_i\theta_i-\sum_{i=1}^n b(\theta_i)}{a(\varphi)}+\sum_{i=1}^n c(y_i,\varphi).$$

Paramètre naturel pour y_i : θ_i
Prédiction pour y_i : $\mu_i=\mathbb{E}(Y_i)=b'(\theta_i)$
Score associé à y_i : $\eta_i=\boldsymbol{x}_i^\top\boldsymbol{\beta}$

Fonction de lien : g telle que $\eta_i=g(\mu_i)=g(b'(\theta_i))$
(g doit être bijective, et croissante...)

Régression GLM

Proposition 2.5: Statistical Properties

sous les hypothèses mentionnées auparavant

- l'estimateur du maximum de vraisemblance $\widehat{\boldsymbol{\beta}}$ existe et est unique,
- $\widehat{\boldsymbol{\beta}} \overset{p.s.}{\to} \boldsymbol{\beta}$ (fortement consistant)
- lorsque $n \to \infty$, $\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta} \overset{\mathcal{L}}{\to} \mathcal{N}(\mathbf{0}, \varphi\boldsymbol{\Sigma})$

où $\boldsymbol{\Sigma} = (\boldsymbol{X}^\top \boldsymbol{W}\boldsymbol{X})^{-1}$ et $\boldsymbol{W} = \text{diag}((V(\mu_i)g'(\mu_i)^2)^{-1})$.

Régression GLM

Proposition 2.6: Statistical Properties

sous les hypothèses mentionnées auparavant

- lorsque $n \to \infty$, $\widehat{\boldsymbol{\eta}} - \boldsymbol{\eta} \xrightarrow{\mathcal{L}} \mathcal{N}(\mathbf{0}, \varphi \boldsymbol{X}\boldsymbol{\Sigma}\boldsymbol{X}^\top)$
- lorsque $n \to \infty$, $\widehat{\boldsymbol{\mu}} - \boldsymbol{\mu} \xrightarrow{\mathcal{L}} \mathcal{N}(\mathbf{0}, \varphi \boldsymbol{\Delta}^{-2}\boldsymbol{X}\boldsymbol{\Sigma}\boldsymbol{X}^\top)$

Déviance

La déviance est l'écart entre la log-vraisemblance obtenue en β, et celle obtenue avec un modèle parfait (dit saturé),

$$D = 2\varphi \times [\log \mathcal{L}(\boldsymbol{y}) - \log \mathcal{L}(\widehat{\boldsymbol{\mu}})]$$

où $\widehat{\boldsymbol{\mu}} = g^{-1}(\boldsymbol{X}\widehat{\boldsymbol{\beta}})$. On peut aussi définir la scaled deviance,

$$D^\star = \frac{D}{\varphi} = 2 \times [\log \mathcal{L}(\boldsymbol{y}) - \log \mathcal{L}(\widehat{\boldsymbol{\mu}})]$$

et la null deviance,

$$D_0^\star = 2 \times [\log \mathcal{L}(\boldsymbol{y}) - \log \mathcal{L}(\bar{y})]$$

Normale : $D = \sum_{i=1}^{n}(y_i - \mu_i)^2$, Poisson : $D = 2\sum_{i=1}^{n}\left\{y_i \ln \frac{y_i}{\mu_i} - (y_i - \mu_i)\right\}$

Gamma : $D = 2\sum_{i=1}^{n}\left\{-\ln \frac{y_i}{\mu_i} + \frac{y_i - \mu_i}{\mu_i}\right\}$

Out-of-sample Risk

Consider training sample $\{(y_i, \boldsymbol{x}_i); i = 1, \cdots, n'\}$ and validation sample $\{(y_i, \boldsymbol{x}_i); i = n' + 1, \cdots, n\}$.
Let $\widehat{m}_{\text{is}}$ denote the model estimated on the training sample.
The empirical out-of-sample risk is

$$\widehat{\mathcal{R}}_{\text{os}} = \frac{1}{n - n'} \sum_{i=n'+1}^{n} \ell(y_i, \widehat{m}_{\text{is}}(\boldsymbol{x}_i))$$

Out-of-sample Risk

Consider a partition $\bigcup_{j=1}^{K} \mathcal{I}_j = \{1, 2, \cdots, n\}$.

Let $\widehat{m}_{-j}$ denote the model estimated on the sample without set $\mathcal{I}_j$,

$$\widehat{m}_{-j} \in \underset{m \in \mathcal{M}}{\operatorname{argmin}} \left\{ \sum_{i \in \{1,\cdots,n\} \setminus \mathcal{I}_j} \ell(y_i, m(\boldsymbol{x}_i)) \right\}$$

The empirical cross-validation risk is

$$\widehat{\mathcal{R}}_{\text{cv}} = \frac{1}{K} \sum_{j=1}^{K} \frac{1}{n_j} \sum_{i \in \mathcal{I}_j} \ell(y_i, \widehat{m}_{-j}(\boldsymbol{x}_i))$$

Out-of-sample Risk

Consider a bootstrap set $\mathcal{I}_b$ of indices $\mathcal{I} = \{1, 2, \cdots, n\}$.
Let $\widehat{m}_b$ denote the model estimated on the sample of points with index $\mathcal{I}_b$,

$$\widehat{m}_b \in \underset{m \in \mathcal{M}}{\operatorname{argmin}} \left\{ \sum_{i \in \mathcal{I}_b} \ell(y_i, m(\boldsymbol{x}_i)) \right\}$$

The empirical out-of-bag cross-validation risk is

$$\widehat{\mathcal{R}}_{\text{oob}} = \frac{1}{B} \sum_{b=1}^{B} \frac{1}{\text{card}(\mathcal{I} \backslash \mathcal{I}_b)} \sum_{i \in \mathcal{I} \backslash \mathcal{I}_b} \ell(y_i, \widehat{m}_b(\boldsymbol{x}_i))$$

Out-of-sample Risk

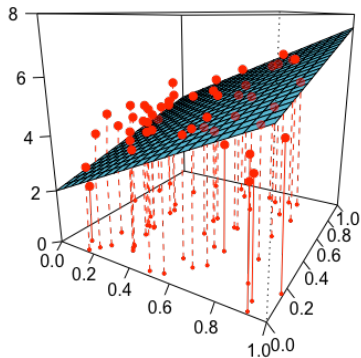
Other classical tools are Akaike's information criterion (AIC), and Schwarz's Bayesian information criterion (BIC),

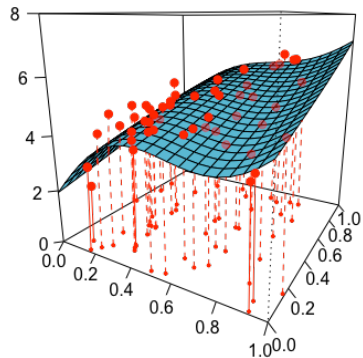
$$\text{AIC} = -2\log\mathcal{L}(\widehat{\boldsymbol{\beta}}) + 2\text{dim}(\widehat{\boldsymbol{\beta}})$$

Preference should be given to the model that has the smallest AIC value.

$$\text{BIC} = -2\log\mathcal{L}(\widehat{\boldsymbol{\beta}}) + \log(n)\cdot\text{dim}(\widehat{\boldsymbol{\beta}})$$

Smoothing Splines





"Natura non facit saltus," Gottfried Leibniz, 1704.

Smoothing Splines

Generalized Additive Models (GAMs) extend Generalized Linear Models (GLMs) by allowing non-linear relationships through smooth functions:

$$g(\mathbb{E}[Y]) = \beta_0 + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p)$$

- $g(\cdot)$ is the (known) link function,
- $f_j(X_j)$ are smooth functions modeled using splines.
- GLM: $g(\mathbb{E}[Y]) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$
- GAM: $g(\mathbb{E}[Y]) = \beta_0 + f_1(X_1) + \cdots + f_p(X_p)$

  GAM allows capturing non-linearity via smooth functions.

Splines are piecewise polynomial functions ensuring smoothness at knots.

- Basis splines (B-splines)
- Natural splines
- Thin-plate splines

Smoothing Splines

A B-spline is defined as:

$$f(X) = \sum_{j=1}^{k} \beta_j B_j(X)$$

where:

- $B_j(X)$ are basis functions,
- β_j are coefficients.

A cubic spline consists of piecewise cubic polynomials ensuring smoothness at knots:

$$f(x) = \sum_{j=1}^{k} \beta_j B_j(x)$$

subject to:

- Continuity at knots

Smoothing Splines

- Continuous first and second derivatives

A natural spline is a cubic spline with additional constraints to enforce linearity at boundaries.

$$f(x) = \sum_{j=1}^{k} \beta_j B_j(x)$$

Ensures:

- Reduced degrees of freedom
- Better extrapolation properties

GAMs estimate smooth functions by minimizing:

$$\sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int f''(x)^2 dx$$

where λ controls smoothness.

The penalty parameter λ is chosen via:

Smoothing Splines

- Cross-validation
- Generalized Cross-Validation (GCV)
- Restricted Maximum Likelihood (REML)

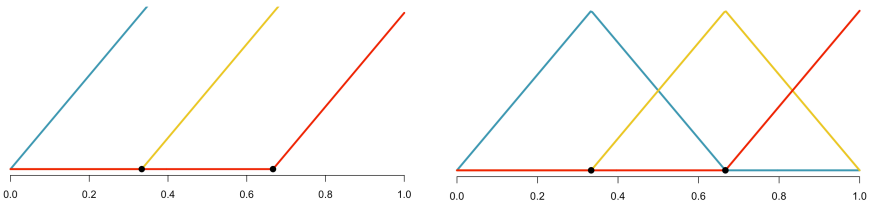
Methods for knot selection:

- Fixed knots (equally spaced or quantiles)
- Adaptive knot selection (using optimization techniques)

Controlling smoothness with penalization

Linear splines (piecewise linear continuous models) are

$$L_1(x) = 1,\ L_2(x) = x,\ L_3(x) = (x - k_1)_+,\ L_4(x) = (x - k_2)_+,\ \ldots$$

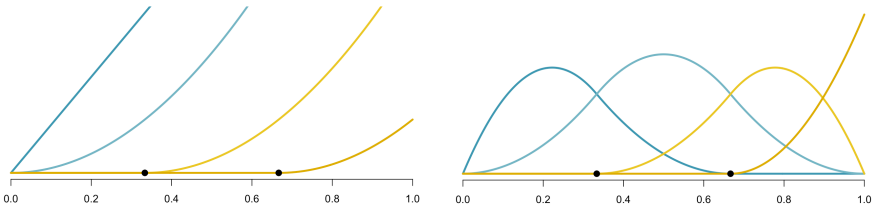



```
> x = sort(runif(n))
> X = bs(x,knots=quantile(x,p=c(1/3,2/3)),degree = 1)
attr(,"degree")
[1] 1
attr(,"knots")
33.33333% 66.66667%
0.3542930 0.7091861
attr(,"Boundary.knots")
[1] 0.003697588 0.989722282
```

Controlling smoothness with penalization

Quadratic splines (piecewise linear continuous models) are

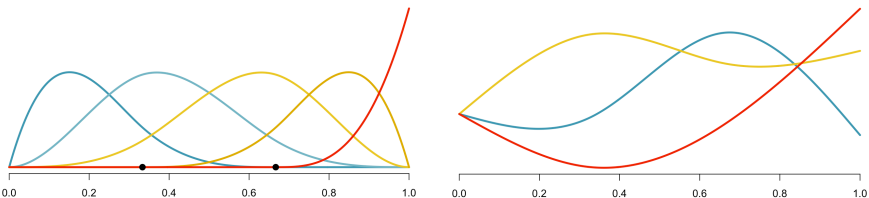
$$L_1(x) = 1,\ L_2(x) = x,\ L_3(x) = x^2,\ L_4(x) = (x - k_1)_+^2,\ \ldots$$




```
> x = sort(runif(n))
> X = bs(x,knots=quantile(x,p=c(1/3,2/3)),degree = 2)
attr(,"degree")
[1] 2
attr(,"knots")
33.33333% 66.66667%
0.3542930 0.7091861
attr(,"Boundary.knots")
[1] 0.003697588 0.989722282
```

Controlling smoothness with penalization

Cubic splines, vs. Natural Splines



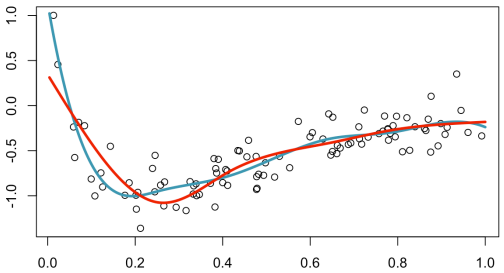

```
> Xb = bs(x,knots=quantile(x,p=c(1/3,2/3)),degree = 3)
> Xn = ns(x,knots=quantile(x,p=c(1/3,2/3)),degree = 3)
```

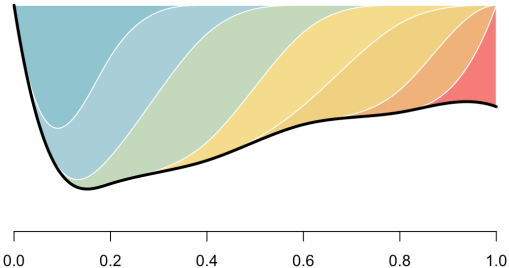
Polynomial models tend to be volatile at the boundaries

So are cubic splines

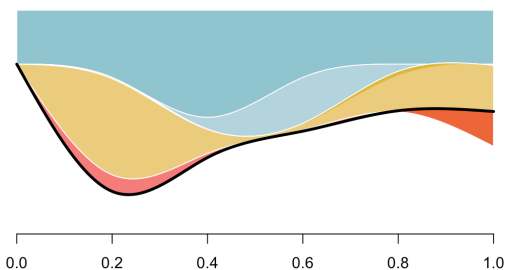
Natural cubic splines adding constraints that the function is linear beyond the boundaries of the data

Controlling smoothness with penalization



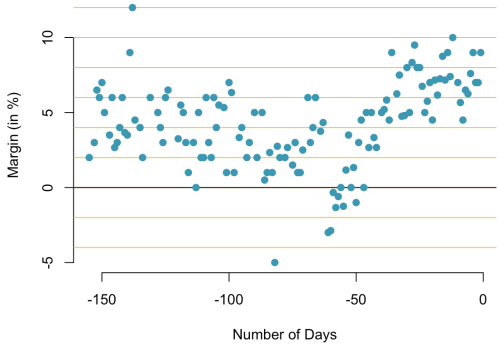


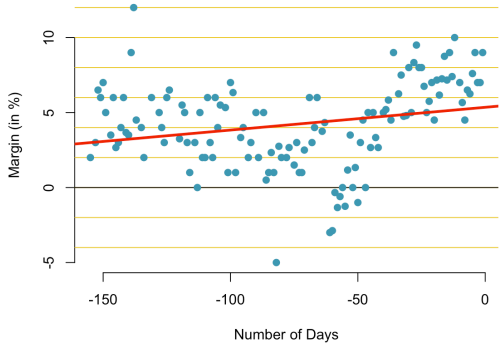

```
> set.seed(1)
> x = sort(runif(100))
> y = sin(log(x))+rnorm(100)/5
> plot(x,y)
> base = data.frame(x,y)
> q = quantile(x,p=c
    (1/5,2/5,3/5,4/5))
> regb = lm(y~bs(x,knots=q))
> regn = lm(y~ns(x,knots=q))
```



Natura non facit saltus

Data source: http://www.pollster.com/08USPresGEMvO-2.html
pollsters for the popular vote between Obama and McCain (2008 US presidential election), last 150 days.



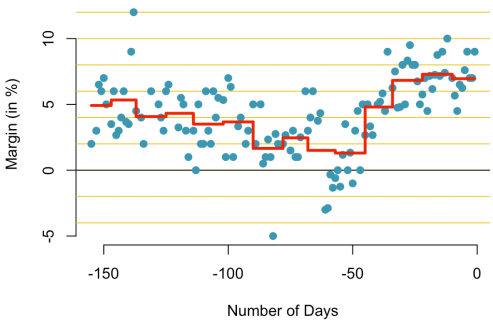


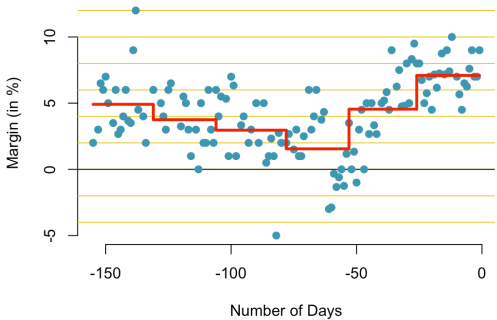

```
> library(dslabs)
> data("polls_2008")
> plot(polls_2008$day, polls_2008$margin*100)
```

Regressogram

From Tukey (1961), the regressogram is defined as

$$\hat{m}_{\boldsymbol{a}}(x) = \frac{\sum_{i=1}^{n} \mathbf{1}(x_i \in [a_j, a_{j+1}))y_i}{\sum_{i=1}^{n} \mathbf{1}(x_i \in [a_j, a_{j+1}))}$$



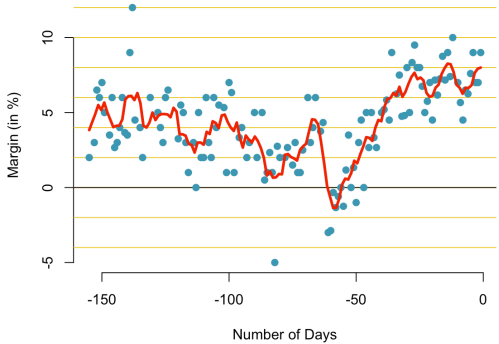


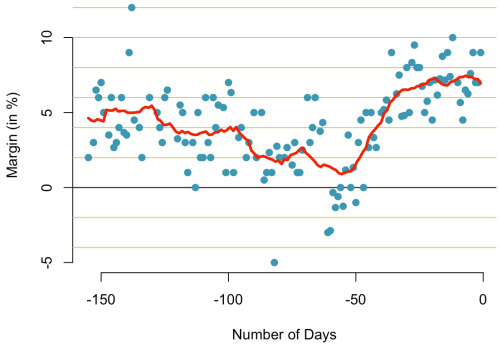

```
> reg=lm(margin~cut(day,seq(-160,0,length=15)),data=polls_2008)
```

Moving Regressogram

and the moving regressogram is

$$\hat{m}(x) = \frac{\sum_{i=1}^{n} \mathbf{1}(x_i \in [x \pm h_n]) y_i}{\sum_{i=1}^{n} \mathbf{1}(x_i \in [x \pm h_n])}$$






```
> with(polls_2008, ksmooth(day, margin, kernel = "box", bandwidth = 7)
```

with bandwidth h_n (size of the neighborhood around x)

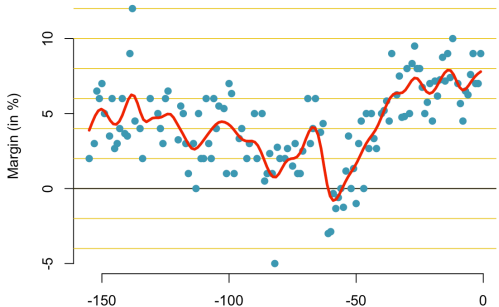
Local Regression

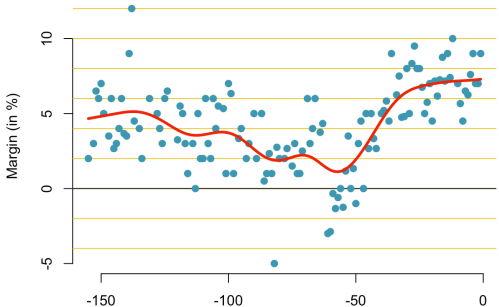
More generally, as moving from the histogram to kernel estimate

$$\tilde{m}(x) = \frac{\sum_{i=1}^{n} y_i \kappa_h(x - x_i)}{\sum_{i=1}^{n} \kappa_h(x - x_i)}$$

Observe that this regression estimator is a weighted average

$$\tilde{m}(x) = \sum_{i=1}^{n} \omega_i(x) y_i \text{ with } \omega_i(x) = \frac{\kappa_h(x - x_i)}{\sum_{i=1}^{n} \kappa_h(x - x_i)}$$





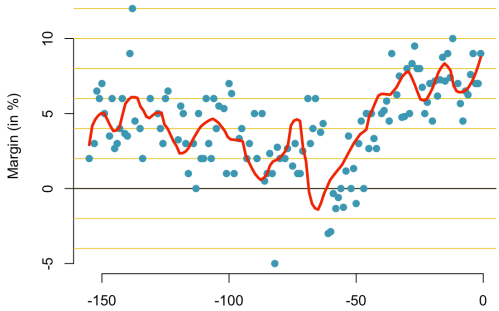
k-Nearest Neighbors

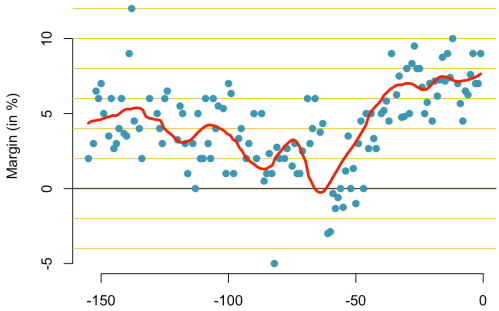
An alternative is to consider

$$\tilde{m}_k(x) = \frac{1}{n}\sum_{i=1}^{n} \omega_{i,k}(x) y_i$$

where $\omega_{i,k}(x) = \dfrac{n}{k}$ if $i \in \mathcal{I}_x^k$ with

$$\mathcal{I}_x^k = \{i : x_i \text{ one of the } k \text{ nearest observations to } x\}$$





Local Regression & k-NN

```
> fit = with(polls_2008, ksmooth(day, margin, kernel = "normal",
    bandwidth = span))
> lines(fit$x, fit$y)
```

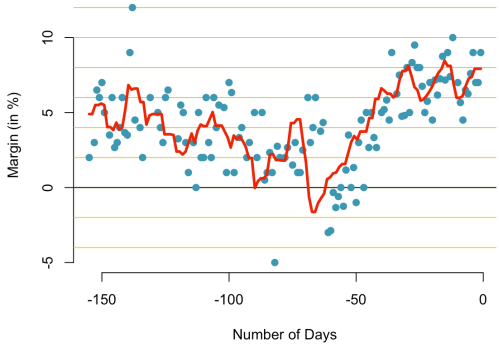
or

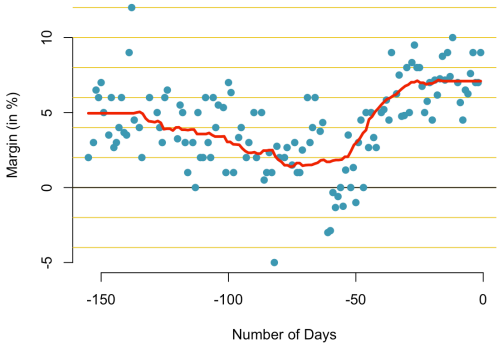
```
> library(FNN)
> p2=knn.reg(train = polls_2008, test = polls_2008, y = polls_2008$margin
    , k = 25)
> lines(polls_2008$day, p2$pred)
```

LOESS (locally weighted polynomial)

Solve

$$\tilde{m}(x) = \text{argmin} \left\{ \sum_{i=1}^{n} \omega_i(x) (y_i - \alpha - \beta x_i)^2 \right\}, \ \omega_i(x) = \frac{\kappa_h (x - x_i)}{\sum_{i=1}^{n} \kappa_h (x - x_i)}$$





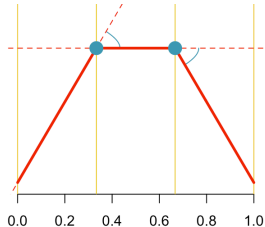

```
1 > fitL = loess(margin ~ day, degree=1, span = 7, data=polls_2008)
```

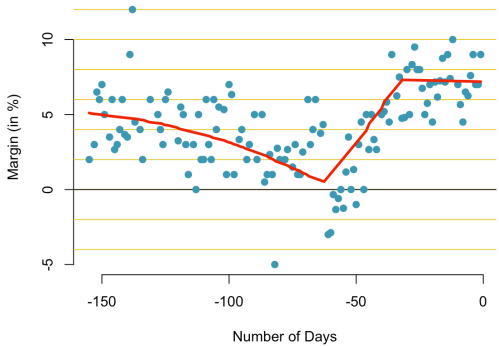
(Linear) Spline Regression

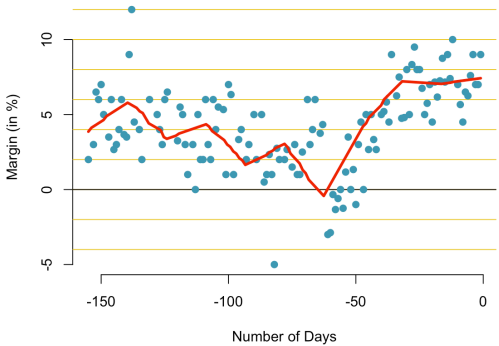
Select some knots $\{s_1, \cdots, s_k\}$, then with $s_0 = 0$

$$\tilde{m}(x) = \alpha + \sum_{j=0}^{k} \beta_j (x - s_k)_+$$

where $(x - s)_+ = (x - s)$ if $x > s$, 0 otherwise





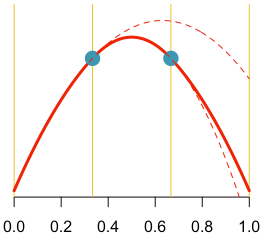


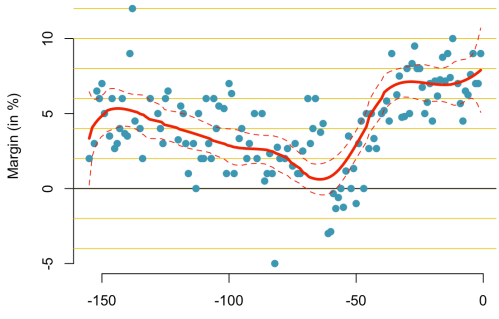
(Quadratic) Spline Regression

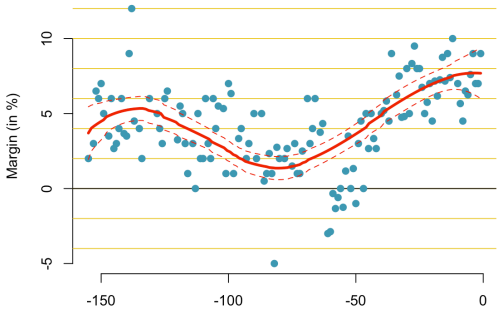
Select some knots $\{s_1, \cdots, s_k\}$, then with $s_0 = 0$

$$\tilde{m}(x) = \alpha + \gamma x + \sum_{j=0}^{k} \beta_j (x - s_k)_+^2$$

where $(x-s)_+^2 = (x-s)^2$ if $x > s$, 0 otherwise







Accuracy

Likelihood-based metrics assess how well a model fits the data. These metrics are based on the likelihood function, which measures the probability of the observed data given the model parameters.

- Log-Likelihood: The natural logarithm of the likelihood function.
- Deviance: A measure of how well the model fits the data relative to a saturated model.
- AIC and BIC: Information criteria used for model selection, balancing fit and complexity.

The Log-Likelihood function is the natural logarithm of the likelihood function. It is used to estimate model parameters by maximizing the likelihood.

$$\text{Log-Likelihood} = \log \mathcal{L}(\theta|y) = \sum_{i=1}^{n} \log(f(y_i|\theta))$$

where

Accuracy

- y_i: Observed data points.
- θ: Model parameters.
- $f(y_i|\theta)$: "Probability" of the data given the model.

Higher log-likelihood values indicate better fitting models.
Deviance is a measure of model fit, representing the difference between the likelihood of a saturated model and the likelihood of the fitted model.

$$\text{Deviance} = -2 \log \left(\frac{\mathcal{L}(\hat{\theta})}{\mathcal{L}(\hat{\theta}_{\text{sat}})} \right)$$

where:

- $\mathcal{L}(\hat{\theta})$: Likelihood of the fitted model.
- $\mathcal{L}(\hat{\theta}_{\text{sat}})$: Likelihood of the saturated model (perfect fit).

Accuracy

Smaller deviance values indicate better fit.
AIC is used for model selection and penalizes models for their complexity. It balances goodness-of-fit with the number of parameters.

$$AIC = -2\ell(\hat{\theta}) + 2k$$

Where:

- $\ell(\hat{\theta})$: Log-Likelihood of the fitted model.
- k: Number of model parameters.

Lower AIC values indicate a better model with a good trade-off between fit and complexity.

Accuracy

BIC is similar to AIC but includes a stronger penalty for complexity. It is derived from a Bayesian perspective.

$$BIC = -2\ell(\hat{\theta}) + k\log(n)$$

Where:

- $\ell(\hat{\theta})$: Log-Likelihood of the fitted model.
- k: Number of model parameters.
- n: Sample size.

Lower BIC values suggest a better model, penalizing larger models more heavily than AIC.

Accuracy

Both AIC and BIC are used to compare models, but they differ in the severity of penalizing model complexity:

- AIC: Tends to select more complex models.
- BIC: More conservative, preferring simpler models with fewer parameters.

When comparing models, lower AIC or BIC indicates a better model. However, BIC typically selects models with fewer parameters compared to AIC.

Regularization

Regularization

In machine learning, a key challenge is enabling models to accurately predict outcomes on unseen data, not just on training data. Regularization is crucial for addressing overfitting—where a model memorizes training data details but cannot generalize to new data. The goal of regularization is to encourage models to learn the broader patterns within the data rather than memorizing it. Techniques like early stopping, ℓ_1 and ℓ_2 regularization, and dropout are designed to prevent overfitting and underfitting, thereby enhancing the model's ability to adapt to and perform well with new data, thus improving model generalization. W

With very general notations,

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^p}\left\{\underbrace{\sum_{i=1}^{n}\ell(y_i,m(\boldsymbol{x}_i;\boldsymbol{\theta}))}_{\text{regularization}}+\lambda\cdot\underbrace{\text{Reg}(\boldsymbol{\theta})}_{\text{loss}}\right\}$$

$\text{Reg}(\boldsymbol{\theta})$ is a penalty function, while $\lambda\geq 0$ is the pregularization parameter

Regularization

- classical LASSO "Least Absolute Shrinkage and Selection", with squared loss (quadratic) and ℓ_1 regularization

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^p}\left\{\sum_{i=1}^{n}\left(y_i - m(\boldsymbol{x}_i;\boldsymbol{\theta})\right)^2 + \lambda\cdot\sum_{j=1}^{p}|\boldsymbol{\theta}_j|\right\}$$

- sparse solution (some weights are zero $\leftrightarrow$ feature selection)
- "Best-subset selection regularization", with squared loss (quadratic) and so-called "ℓ_0" regularization

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^p}\left\{\sum_{i=1}^{n}\left(y_i - m(\boldsymbol{x}_i;\boldsymbol{\theta})\right)^2 + \lambda\cdot\sum_{j=1}^{p}\mathbf{1}_{\mathbb{R}\setminus\{0\}}(\boldsymbol{\theta}_j)\right\}$$

Regularization

- squared loss (quadratic) and ℓ_q regularization

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^p}\left\{\sum_{i=1}^{n}\left(y_i - m(\boldsymbol{x}_i;\boldsymbol{\theta})\right)^2 + \lambda\cdot\sum_{j=1}^{p}|\boldsymbol{\theta}_j|^q\right\}$$

- $q \geq 1$ cost function is convex (strictly convex if $q > 1$)
- $q < 1$ cost function is not convex
- squared loss (quadratic) and elastic net regularization

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^p}\left\{\sum_{i=1}^{n}\left(y_i - m(\boldsymbol{x}_i;\boldsymbol{\theta})\right)^2 + \lambda\cdot\left[\alpha\sum_{j=1}^{p}|\boldsymbol{\theta}_j| + (1-\alpha)\sum_{j=1}^{p}\boldsymbol{\theta}_j^2\right]\right\},$$

with $\alpha \in [0, 1]$.

Regularization

- ε-insensitive loss ℓ_2 regularization

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^p}\left\{\sum_{i=1}^{n}\left(|y_i - m(\boldsymbol{x}_i;\boldsymbol{\theta})| - \varepsilon\right)_+ + \frac{\lambda}{2}\cdot\sum_{j=1}^{p}|\boldsymbol{\theta}_j|^2\right\}$$

- quadratic programming (dual can be solved)

Suppose that variables are centered, $\overline{y} = \overline{x} = 0$

$$\min_{\beta}\left\{\frac{1}{n}\sum_{i=1}^{n}(y_i - \beta x_i)^2\right\} \text{ : (classical) ordinary least squares (OLS)}$$

$$\min_{\beta}\left\{\frac{1}{n}\sum_{i=1}^{n}(y_i - \beta x_i)^2 + \lambda\beta^2\right\} \text{ : Ridge, with OLS loss,}$$

$$\min_{\beta}\left\{\frac{1}{n}\sum_{i=1}^{n}(y_i - \beta x_i)^2 + \lambda|\beta|\right\} \text{ : Lasso, with OLS loss,}$$

Regularization

Least squares , $\text{LS} = \sum_{i=1}^{n}(y_i - \beta x_i)^2 = \boldsymbol{y}^\top \boldsymbol{y} - 2\boldsymbol{x}^\top \boldsymbol{y}\beta + \beta \boldsymbol{x}^\top \boldsymbol{x}\beta$

$$\frac{\partial \text{LS}}{\partial \beta} = -2\sum_{i=1}^{n} x_i(y_i - \beta x_i) = -2\boldsymbol{x}^\top(\boldsymbol{y} - \beta \boldsymbol{x}) \text{ and } \widehat{\beta}^{\text{ols}} = \frac{\boldsymbol{x}^\top \boldsymbol{y}}{\boldsymbol{x}^\top \boldsymbol{x}} = (\boldsymbol{x}^\top \boldsymbol{x})^{-1}\boldsymbol{x}^\top \boldsymbol{y}$$

Ridge, $\text{PLS} = \boldsymbol{y}^\top \boldsymbol{y} - 2\boldsymbol{x}^\top \boldsymbol{y}\beta + \beta \boldsymbol{x}^\top \boldsymbol{x}\beta + \lambda\beta^2$

$$\frac{\partial \text{PLS}}{\partial \beta} = -2\boldsymbol{y}^\top \boldsymbol{x} + 2\boldsymbol{x}^\top \boldsymbol{x}\beta + 2\beta\lambda \text{ and } \widehat{\beta}_\lambda^{\text{ridge}} = \frac{\boldsymbol{x}^\top \boldsymbol{y}}{\boldsymbol{x}^\top \boldsymbol{x}+\lambda} = (\boldsymbol{x}^\top \boldsymbol{x}+\lambda)^{-1}\boldsymbol{x}^\top \boldsymbol{y}$$

Lasso, $\text{PLS} = \boldsymbol{y}^\top \boldsymbol{y} - 2\boldsymbol{x}^\top \boldsymbol{y}\beta + \beta \boldsymbol{x}^\top \boldsymbol{x}\beta + \lambda|\beta|$

$$\frac{\partial \text{PLS}}{\partial \beta} = -2\boldsymbol{y}^\top \boldsymbol{x} + 2\boldsymbol{x}^\top \boldsymbol{x}\widehat{\beta} \pm \lambda$$

where the sign of $\pm$ is the sign of $\widehat{\beta}$.

Regularization

Let's assume that the least-squares estimator (obtained when $\lambda = 0$) is (strictly) positive, i.e. $\boldsymbol{x}^\top \boldsymbol{y} > 0$.

If λ is not too large $\widehat{\beta}_\lambda$ and $\widehat{\beta}^{\text{ols}}$ are of the same sign, and

$$-2\boldsymbol{y}^\top \boldsymbol{x} + 2\boldsymbol{x}^\top \boldsymbol{x}\widehat{\beta} + \lambda = 0, \text{ i.e. } \widehat{\beta}_\lambda^{\text{lasso}} = \frac{\boldsymbol{y}^\top \boldsymbol{x} - \lambda/2}{\boldsymbol{x}^\top \boldsymbol{x}}$$

We then increase λ until we have $\widehat{\beta}_\lambda = 0$.

By increasing it a little, *widehat*β_λ cannot become negative, and therefore

$$-2\boldsymbol{y}^\top \boldsymbol{x} + 2\boldsymbol{x}^\top \boldsymbol{x}\widehat{\beta} - \lambda = 0, \text{ i.e. } \widehat{\beta}_\lambda^{\text{lasso}} = \frac{\boldsymbol{y}^\top \boldsymbol{x} + \lambda/2}{\boldsymbol{x}^\top \boldsymbol{x}}$$

But this solution is positive ($\boldsymbol{y}^\top \boldsymbol{x} > 0$) so we must have $\widehat{\beta}_\lambda < 0$.
The only possible solution is then $\widehat{\beta}_\lambda = 0$.

Regularization

If x is centered ($\bar{x} = 0$) and reduced ($\boldsymbol{x}^\top \boldsymbol{x} = n$),

$$\widehat{\beta}^{\text{ols}} = \frac{1}{n} \boldsymbol{x}^\top \boldsymbol{y}$$

$$\widehat{\beta}_\lambda^{\text{ridge}} = \frac{\boldsymbol{x}^\top \boldsymbol{y}}{n+\lambda} = \frac{n}{n+\lambda} \cdot \frac{1}{n} \boldsymbol{x}^\top \boldsymbol{y} = \frac{n}{n+\lambda} \cdot \widehat{\beta}^{\text{ols}}$$

$$\widehat{\beta}_\lambda^{\text{lasso}} = \frac{\boldsymbol{x}^\top \boldsymbol{y} \pm \lambda/2}{n} = \frac{1}{n} \boldsymbol{x}^\top \boldsymbol{y} \pm \frac{\lambda}{2n} = \widehat{\beta}^{\text{ols}} \pm \frac{\lambda}{2n}$$

Ridge Regression

- Idea: perturb the matrix $\boldsymbol{X}$ to move its eigenvalues away from 0 and thus stabilize the inversion of $\boldsymbol{X}^\top\boldsymbol{X}$.
- Notation: $0 \leq \mu_1 \leq \ldots \mu_p$ ordered eigenvalues of $\boldsymbol{X}^\top\boldsymbol{X}$ and let $\boldsymbol{P}$ be the orthogonal matrix such that $\boldsymbol{X}^\top\boldsymbol{X} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^\top$ with $\boldsymbol{D} = \mathrm{diag}(\mu_1, \ldots, \mu_p)$.
- Let $\lambda \geq 0$, $\boldsymbol{X}^\top\boldsymbol{X} + \lambda\mathbb{I}_p$ has the same eigenvectors as $\boldsymbol{X}^\top\boldsymbol{X}$ and for eigenvalues $\mu_j + \lambda$ for $j = 1, \ldots, p$.
- Hoerl and Kennard (1970a): replace $\boldsymbol{X}^\top\boldsymbol{X}$ by $\boldsymbol{X}^\top\boldsymbol{X} + \lambda\mathbb{I}_p$ in the OLS definition:

$$\widehat{\boldsymbol{X}^{\text{ridge}}}(\lambda) = (\boldsymbol{X}^\top\boldsymbol{X} + \lambda\mathbb{I}_p)^{-1}\boldsymbol{X}^\top\boldsymbol{y}$$

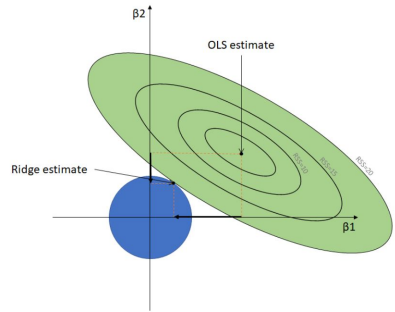
- If $\lambda = 0$, then $\widehat{\boldsymbol{\beta}}^{\text{ridge}} = \widehat{\boldsymbol{\beta}}^{\text{mco}}$; if $\lambda \to \infty$, then $\widehat{\boldsymbol{\beta}}^{\text{ridge}} \to \mathbf{0}$.
- Fixing λ is important. What are the consequences on the bias and variance of the estimators?

Ridge Regression

- Let $\lambda \geq 0$, we can show that $\widehat{\boldsymbol{\beta}}^{\text{ridge}}(\lambda)$ minimizes

$$\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|^2$$

$$= \sum_{i=1}^{n}(y_i - \boldsymbol{x}_i\boldsymbol{\beta})^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$



https://rstatisticsblog.com/

- Let $\widetilde{\boldsymbol{\beta}}$ be the estimator minimizing

$$\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 \text{ s.t. } \|\boldsymbol{\beta}\|^2 \leq \delta.$$

$\forall \lambda > 0$, $\exists \delta$ so that the two solutions coincide.

Ridge Regression

- $\widehat{\boldsymbol{\beta}}^{\text{ridge}} = \left(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \mathbb{I}_p\right)^{-1} \boldsymbol{X}^\top \boldsymbol{X} \widehat{\boldsymbol{\beta}}$.
- $\mathbb{E}\left(\widehat{\boldsymbol{\beta}}^{\text{ridge}}\right) = \boldsymbol{\beta} - \lambda \left(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \mathbb{I}_p\right)^{-1} \boldsymbol{\beta}$.
- $\text{Var}(\widehat{\boldsymbol{\beta}}^{\text{ridge}}) = \sigma^2 \left(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \mathbb{I}_p\right)^{-1} \boldsymbol{X}^\top \boldsymbol{X} \left(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \mathbb{I}_p\right)^{-1}$.
- and the mean squared error is

$$\text{MSE}(\widehat{\boldsymbol{\beta}}^{\text{ridge}}) \left(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \mathbb{I}_p\right)^{-1} \left(\sigma^2 (\boldsymbol{X}^\top \boldsymbol{X}) + \lambda^2 \boldsymbol{\beta}\boldsymbol{\beta}^\top\right) \left(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \mathbb{I}_p\right)^{-1}$$

while $\text{MSE}(\widehat{\boldsymbol{\beta}}) = \text{Var}(\widehat{\boldsymbol{\beta}}) = \sigma^2 (\boldsymbol{X}^\top \boldsymbol{X})^{-1}$.

Régression Ridge

If μ_j is eigenvalue of $\boldsymbol{X}^\top\boldsymbol{X}$

- Under $\mathcal{A}_1$, $\text{trace}(\text{MSE}(\widehat{\beta})) = \sum_{j=1}^{p} \frac{\sigma^2}{\mu_j}$;
- while $\text{trace}(\text{MSE}(\widehat{\beta}^{\text{ridge}})) = \sum_{j=1}^{r} \frac{\sigma^2\mu_j + \lambda^2\left(\boldsymbol{P}^\top\beta\right)_j^2}{(\mu_j+\lambda)^2}$
- Thus $\text{trace}(\text{MSE}(\widehat{\beta}^{\text{ridge}})) \leq \text{trace}(\text{MSE}(\widehat{\beta})) \quad \Leftrightarrow \quad \lambda \leq \frac{2\sigma^2}{\beta^\top\beta}$.

Shrinkage

- Let $\widehat{\theta}$ be an unbiased estimator of θ, with variance σ^2, so $\text{MSE}(\widehat{\theta}) = \sigma^2$.
- Let $\lambda > 0$ et $\widetilde{\theta} = \dfrac{\widehat{\theta}}{1+\lambda}$ alors

$$\mathbb{E}(\widetilde{\theta}) = \frac{\theta}{1+\lambda}, \quad \text{Var}(\widetilde{\theta}) = \frac{\sigma^2}{(1+\lambda)^2} \text{ and } \text{MSE}(\widetilde{\theta}) = \frac{\lambda^2\theta^2 + \sigma^2}{(1+\lambda)^2}.$$

  It is possible to find $\lambda^\star$ such that $\text{MSE}(\widetilde{\theta})$ is minimal
- $\widetilde{\theta}$ is biased but with a smaller variance than $\widehat{\theta}$.

Similarly, we can find $\lambda^\star$ such that $\lambda \mapsto \text{trace}(\text{MSE}(\widehat{\boldsymbol{\beta}}^{\text{ridge}}(\lambda)))$ is minimal

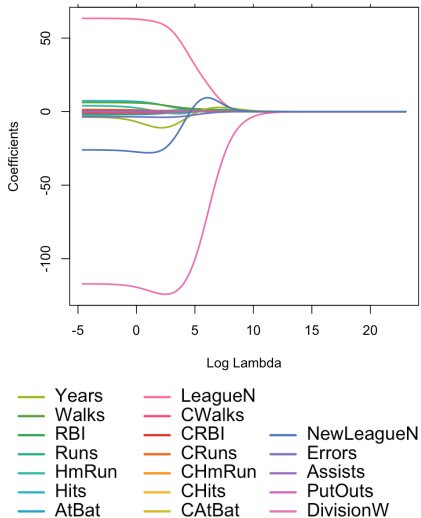
Ridge Regression

Use

```
> library(MASS)
> ?lm.ridge
```

or

```
> library(ISLR)
> library(glmnet)
> Hitters = na.omit(Hitters)
> x = model.matrix(Salary~., Hitters
    )[,-1]
> y = Hitters$Salary
> ridge_mod = glmnet(x, y, alpha =
    0)
> plot(ridge_mod, var="lambda")
```

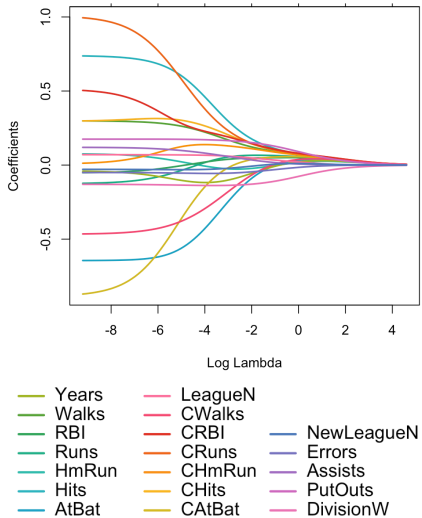


Ridge Regression

instead of the Euclidean norm $\|\beta\|_2$ use Mahalanobis norm
i.e. center and scale $\boldsymbol{x}_j$

$$\boldsymbol{x}_j \mapsto \frac{\boldsymbol{x}_j - \overline{x}_j}{s_{x_j}}$$

```
> ys = (y-mean(y))/sd(y)
> xs = x
> for(i in 1:ncol(x)) xs[,i] = (x[,i
    ]-mean(x[,i]))/sd(x[,i])
> ridge_mod_s = glmnet(xs, ys, alpha
    = 0)
> plot(ridge_mod_s, var="lambda")
```



LASSO Regression

- Here, we're not constraining the Euclidean $\|boldsymbol{\beta}\| = \|boldsymbol{\beta}\|_2$ norm (i.e. the ℓ_2 norm) but the ℓ_1 norm of the coefficients.
- The Lasso method consists in minimizing

$$\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 \text{ s.t. } \|\boldsymbol{\beta}\|_1 \leq \delta.$$

- There is no exact solution (several algorithms have been proposed - LARS, coordinate descent algorithm).
- It can be shown that the problem is equivalent to minimizing the regularized problem

$$\widehat{\boldsymbol{\beta}}^{\text{lasso}}(\lambda) = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|_1$$

($\forall \lambda > 0$, $\exists \delta > 0$ such that the solutions to these two problems coincide)

Ridge, LASSO and Elastic-Net

Definition 2.12: Best-subset selection (OLS), Hoerl and Kennard (1970b)

$$\widehat{\boldsymbol{\beta}}_\lambda^{\text{bss}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^k}{\text{argmin}} \left\{ \frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{k} \mathbf{1}_{\beta_j \neq 0} \right\}.$$

Definition 2.13: Best-subset selection (GLM)

$$\widehat{\boldsymbol{\beta}}_\lambda^{\text{bss}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^k}{\text{argmin}} \left\{ -\sum_{i=1}^{n} \log f(y_i|\mu_i = g^{-1}(\boldsymbol{x}_i^\top \boldsymbol{\beta})) + \lambda \sum_{j=1}^{k} \mathbf{1}_{\beta_j \neq 0} \right\}.$$

Ridge, LASSO and Elastic-Net

Definition 2.14: Ridge Estimator (OLS), Hoerl and Kennard (1970b)

$$\widehat{\boldsymbol{\beta}}_\lambda^{\text{ridge}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^k}{\text{argmin}} \left\{ \frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top\boldsymbol{\beta})^2 + \lambda\sum_{j=1}^{k}\beta_j^2 \right\}.$$

$$\widehat{\boldsymbol{\beta}}_\lambda^{\text{ridge}} = (\boldsymbol{X}^\top\boldsymbol{X} + \lambda\mathbb{I})^{-1}\boldsymbol{X}^\top\boldsymbol{y}$$

Definition 2.15: Ridge Estimator (GLM)

$$\widehat{\boldsymbol{\beta}}_\lambda^{\text{ridge}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^k}{\text{argmin}} \left\{ -\sum_{i=1}^{n}\log f(y_i|\mu_i = g^{-1}(\boldsymbol{x}_i^\top\boldsymbol{\beta})) + \lambda\sum_{j=1}^{k}\beta_j^2 \right\}.$$

Ridge, LASSO and Elastic-Net

Definition 2.16: LASSO Estimator (OLS), Tibshirani (1996)

$$\widehat{\boldsymbol{\beta}}_\lambda^{\text{lasso}} = \text{argmin}\left\{\frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top\boldsymbol{\beta})^2 + \lambda\sum_{j=1}^{k}|\beta_j|\right\}.$$

Definition 2.17: LASSO Estimator (GLM)

$$\widehat{\boldsymbol{\beta}}_\lambda^{\text{lasso}} = \text{argmin}\left\{-\sum_{i=1}^{n}\log f(y_i|\mu_i = g^{-1}(\boldsymbol{x}_i^\top\boldsymbol{\beta})) + \lambda\sum_{j=1}^{k}|\beta_j|\right\}.$$

Ridge, LASSO and Elastic-Net

Elastic net

$$\min\left\{\frac{1}{2}\sum_{i=1}^{n}(y_i-\boldsymbol{x}_i^\top\boldsymbol{\beta})^2+\lambda_1\sum_{j=1}^{k}|\beta_j|+\frac{\lambda_2}{2}\sum_{j=1}^{k}\beta_j^2\right\},$$

e.g. $\lambda_1=\alpha\lambda$ and $\lambda_2=(1-\alpha)\lambda$ (two parameters — one for the global regularization, one for the trade-off between Ridge (Tikhonov) vs. Lasso).

Ridge, LASSO and Elastic-Net

Suppose that we group the covariates in G groups, $\boldsymbol{\beta} = (\beta_0, \boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \cdots, \boldsymbol{\beta}_G)$

Definition 2.18: Group LASSO Estimator (OLS), Tibshirani (1996)

$$\widehat{\boldsymbol{\beta}}_\lambda^{G-\text{lasso}} = \text{argmin}\left\{ \frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2 + \sum_{g=1}^{G}\lambda_g \|\boldsymbol{\beta}_g\|_2 \right\}.$$

Definition 2.19: Group LASSO Estimator (GLM)

$$\widehat{\boldsymbol{\beta}}_\lambda^{G-\text{lasso}} = \text{argmin}\left\{ -\sum_{i=1}^{n}\log f(y_i|\mu_i = g^{-1}(\boldsymbol{x}_i^\top \boldsymbol{\beta})) + \sum_{g=1}^{G}\lambda_g \|\boldsymbol{\beta}_g\|_2 \right\}.$$

Ridge, LASSO and Elastic-Net

Suppose we have an adjacency relation on the covariate components, i.e., the covariate component X_j is naturally embedded between the components X_{j-1} and X_{j+1}

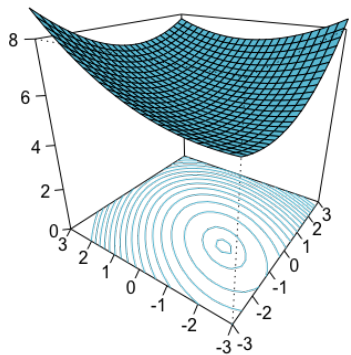
Definition 2.20: Fused LASSO Estimator (OLS), Tibshirani (1996)

$$\widehat{\beta}_\lambda^{F-\text{lasso}} = \text{argmin}\left\{\frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top\boldsymbol{\beta})^2 + \sum_{j=2}^{k}\lambda_j|\beta_j - \beta_{j-1}|\right\}.$$

Definition 2.21: Fused LASSO Estimator (GLM)

$$\widehat{\beta}_\lambda^{F-\text{lasso}} = \text{argmin}\left\{-\sum_{i=1}^{n}\log f(y_i|\mu_i = g^{-1}(\boldsymbol{x}_i^\top\boldsymbol{\beta})) + \sum_{j=2}^{k}\lambda_j|\beta_j - \beta_{j-1}|\right\}.$$

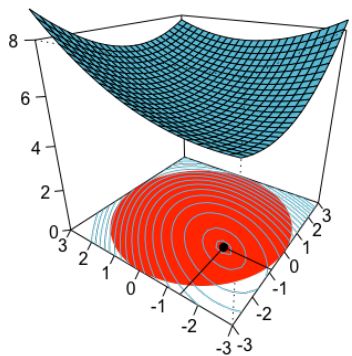
Least Square



Without any contraint, the least square problem is quadratic (and strictly convex)

$$\widehat{\boldsymbol{\beta}}^{\text{ols}} = \text{argmin}\left\{\frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2\right\}.$$

Ridge: Least Square with ℓ_2 Penalty



$$\widehat{\beta}_\lambda^{\text{ridge}} = \text{argmin}\left\{ \frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \beta)^2 + \lambda \sum_{j=1}^{k} \beta_j^2 \right\}$$

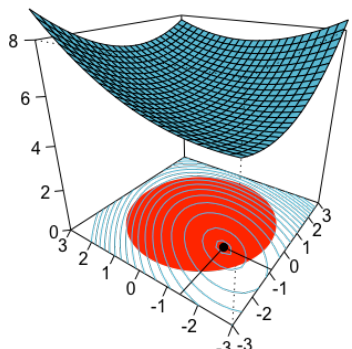
Lagrangian of constrained optimization problem

$$\min\left\{ \frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \beta)^2 \right\} \text{ s.t. } \sum_{j=1}^{k} \beta_j^2 \leq r_\lambda$$

or $\min_{\beta:\ \beta^\top\beta \leq r_\lambda} \left\{ \frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}\beta)^\top(\boldsymbol{y} - \boldsymbol{X}\beta) \right\}$,

with λ small enough, i.e. $\widehat{\beta}_\lambda^{\text{ridge}} = \widehat{\beta}^{\text{ols}}$

Ridge: Least Square with ℓ_2 Penalty



$$\widehat{\beta}_\lambda^{\text{ridge}} = \text{argmin}\left\{ \frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \beta)^2 + \lambda \sum_{j=1}^{k} \beta_j^2 \right\}$$

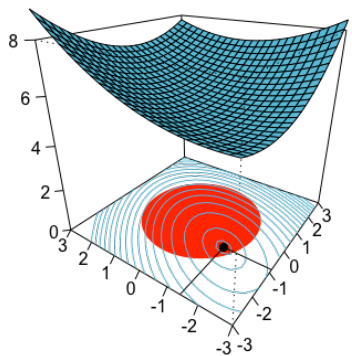
Lagrangian of constrained optimization problem

$$\min\left\{ \frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \beta)^2 \right\} \text{ s.t. } \sum_{j=1}^{k} \beta_j^2 \leq r_\lambda$$

or $\min_{\beta:\ \beta^\top\beta \leq r_\lambda} \left\{ \frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}\beta)^\top(\boldsymbol{y} - \boldsymbol{X}\beta) \right\}$,

with λ small enough, i.e. $\widehat{\beta}_\lambda^{\text{ridge}} = \widehat{\beta}^{\text{ols}}$

Ridge: Least Square with ℓ_2 Penalty



$$\widehat{\beta}_\lambda^{\text{ridge}} = \text{argmin}\left\{ \frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \beta)^2 + \lambda \sum_{j=1}^{k} \beta_j^2 \right\}$$

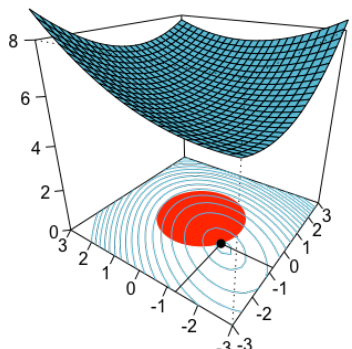
Lagrangian of constrained optimization problem

$$\min\left\{ \frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \beta)^2 \right\} \text{ s.t. } \sum_{j=1}^{k} \beta_j^2 \leq r_\lambda$$

or $\min_{\beta:\ \beta^\top\beta \leq r_\lambda} \left\{ \frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}\beta)^\top(\boldsymbol{y} - \boldsymbol{X}\beta) \right\}$,

with λ small enough, i.e. $\widehat{\beta}_\lambda^{\text{ridge}} = \widehat{\beta}^{\text{ols}}$

Ridge: Least Square with ℓ_2 Penalty



$$\widehat{\beta}_\lambda^{\text{ridge}} = \text{argmin}\left\{\frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top\beta)^2 + \lambda\sum_{j=1}^{k}\beta_j^2\right\}$$

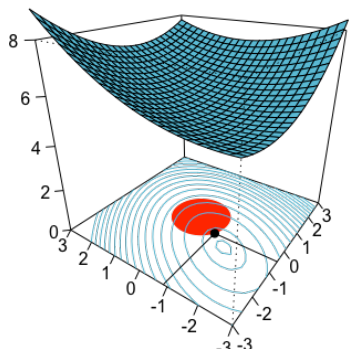
Lagrangian of constrained optimization problem

$$\min\left\{\frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top\beta)^2\right\} \text{ s.t. } \sum_{j=1}^{k}\beta_j^2 \leq r_\lambda$$

or $\min_{\beta:\ \beta^\top\beta\leq r_\lambda}\left\{\frac{1}{2}(\boldsymbol{y}-\boldsymbol{X}\beta)^\top(\boldsymbol{y}-\boldsymbol{X}\beta)\right\}$,

with λ large enough, i.e. $\widehat{\beta}_\lambda^{\text{ridge}} \neq \widehat{\beta}^{\text{ols}}$

Ridge: Least Square with ℓ_2 Penalty



$$\widehat{\beta}_\lambda^{\text{ridge}} = \text{argmin}\left\{ \frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \beta)^2 + \lambda \sum_{j=1}^{k} \beta_j^2 \right\}$$

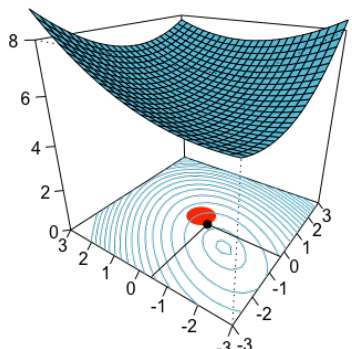
Lagrangian of constrained optimization problem

$$\min\left\{ \frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \beta)^2 \right\} \text{ s.t. } \sum_{j=1}^{k} \beta_j^2 \leq r_\lambda$$

or $\min_{\beta:\ \beta^\top\beta \leq r_\lambda} \left\{ \frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}\beta)^\top(\boldsymbol{y} - \boldsymbol{X}\beta) \right\}$,

with λ large enough, i.e. $\widehat{\beta}_\lambda^{\text{ridge}} \neq \widehat{\beta}^{\text{ols}}$

Ridge: Least Square with ℓ_2 Penalty



$$\widehat{\beta}_\lambda^{\text{ridge}} = \text{argmin} \left\{ \frac{1}{2} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^\top \beta)^2 + \lambda \sum_{j=1}^{k} \beta_j^2 \right\}$$

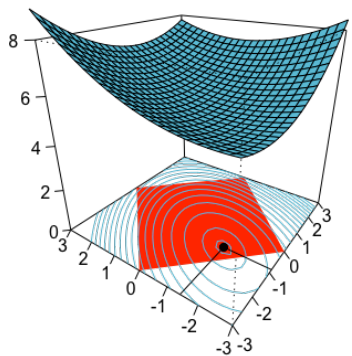
Lagrangian of constrained optimization problem

$$\min \left\{ \frac{1}{2} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^\top \beta)^2 \right\} \text{ s.t. } \sum_{j=1}^{k} \beta_j^2 \leq r_\lambda$$

or $\min_{\beta:\ \beta^\top \beta \leq r_\lambda} \left\{ \frac{1}{2} (\boldsymbol{y} - \boldsymbol{X}\beta)^\top (\boldsymbol{y} - \boldsymbol{X}\beta) \right\}$,

with λ large enough, i.e. $\widehat{\beta}_\lambda^{\text{ridge}} \to \mathbf{0}$

Lasso: Least Square with ℓ_1 Penalty



$$\widehat{\beta}_\lambda^{\text{lasso}} = \text{argmin}\left\{\frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top\beta)^2 + \lambda\sum_{j=1}^{k}|\beta_j|\right\}$$

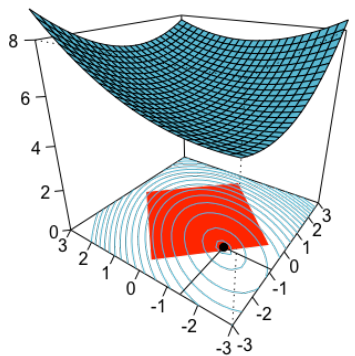
Lagrangian of constrained optimization problem

$$\min\left\{\frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top\beta)^2\right\} \text{ s.t. } \sum_{j=1}^{k}|\beta_j| \leq r_\lambda$$

or $$\min_{\beta:\ |\beta|^\top\mathbf{1}\leq r_\lambda}\left\{\frac{1}{2}(\boldsymbol{y}-\boldsymbol{X}\beta)^\top(\boldsymbol{y}-\boldsymbol{X}\beta)\right\},$$

with λ small enough, i.e. $\widehat{\beta}_\lambda^{\text{lasso}} = \widehat{\beta}^{\text{ols}}$

Lasso: Least Square with ℓ_1 Penalty



$$\widehat{\beta}_\lambda^{\text{lasso}} = \text{argmin}\left\{\frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top\beta)^2 + \lambda\sum_{j=1}^{k}|\beta_j|\right\}$$

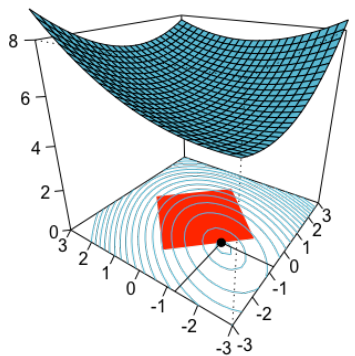
Lagrangian of constrained optimization problem

$$\min\left\{\frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top\beta)^2\right\} \text{ s.t. } \sum_{j=1}^{k}|\beta_j| \leq r_\lambda$$

or $$\min_{\beta:\ |\beta|^\top\mathbf{1}\leq r_\lambda}\left\{\frac{1}{2}(\boldsymbol{y}-\boldsymbol{X}\beta)^\top(\boldsymbol{y}-\boldsymbol{X}\beta)\right\},$$

with λ small enough, i.e. $\widehat{\beta}_\lambda^{\text{lasso}} = \widehat{\beta}^{\text{ols}}$

Lasso: Least Square with ℓ_1 Penalty



$$\widehat{\beta}_\lambda^{\text{lasso}} = \text{argmin}\left\{\frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top\beta)^2 + \lambda\sum_{j=1}^{k}|\beta_j|\right\}$$

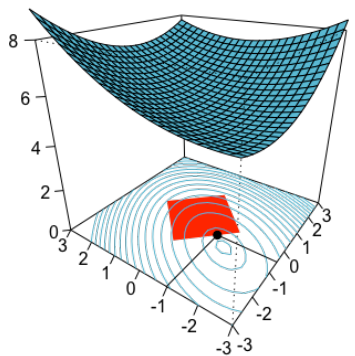
Lagrangian of constrained optimization problem

$$\min\left\{\frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top\beta)^2\right\} \text{ s.t. } \sum_{j=1}^{k}|\beta_j| \leq r_\lambda$$

or $$\min_{\beta:\ |\beta|^\top\mathbf{1}\leq r_\lambda}\left\{\frac{1}{2}(\boldsymbol{y}-\boldsymbol{X}\beta)^\top(\boldsymbol{y}-\boldsymbol{X}\beta)\right\},$$

with λ small enough, i.e. $\widehat{\beta}_\lambda^{\text{lasso}} = \widehat{\beta}^{\text{ols}}$

Lasso: Least Square with ℓ_1 Penalty



$$\widehat{\beta}_\lambda^{\text{lasso}} = \text{argmin}\left\{\frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top\beta)^2 + \lambda\sum_{j=1}^{k}|\beta_j|\right\}$$

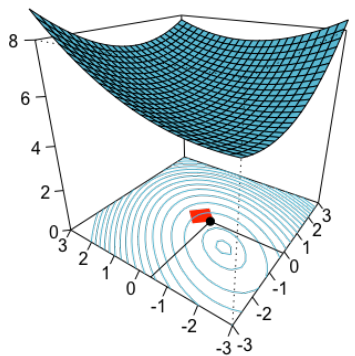
Lagrangian of constrained optimization problem

$$\min\left\{\frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top\beta)^2\right\} \text{ s.t. } \sum_{j=1}^{k}|\beta_j| \leq r_\lambda$$

or $$\min_{\beta:\ |\beta|^\top\mathbf{1}\leq r_\lambda}\left\{\frac{1}{2}(\boldsymbol{y}-\boldsymbol{X}\beta)^\top(\boldsymbol{y}-\boldsymbol{X}\beta)\right\},$$

with λ large enough, i.e. $\widehat{\beta}_\lambda^{\text{lasso}} \neq \widehat{\beta}^{\text{ols}}$

Lasso: Least Square with ℓ_1 Penalty



$$\widehat{\beta}_\lambda^{\text{lasso}} = \text{argmin}\left\{ \frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \beta)^2 + \lambda \sum_{j=1}^{k} |\beta_j| \right\}$$

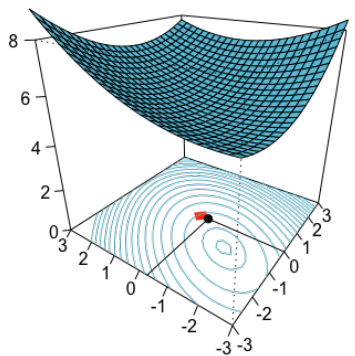
Lagrangian of constrained optimization problem

$$\min\left\{ \frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top \beta)^2 \right\} \text{ s.t. } \sum_{j=1}^{k} |\beta_j| \leq r_\lambda$$

or $$\min_{\beta:\ |\beta|^\top \mathbf{1} \leq r_\lambda} \left\{ \frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}\beta)^\top (\boldsymbol{y} - \boldsymbol{X}\beta) \right\},$$

with λ large enough, i.e. $\widehat{\beta}_\lambda^{\text{lasso}} \to \mathbf{0}$

Lasso: Least Square with ℓ_1 Penalty



$$\widehat{\beta}_\lambda^{\text{lasso}} = \text{argmin}\left\{\frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top\beta)^2 + \lambda\sum_{j=1}^{k}|\beta_j|\right\}$$

Lagrangian of constrained optimization problem

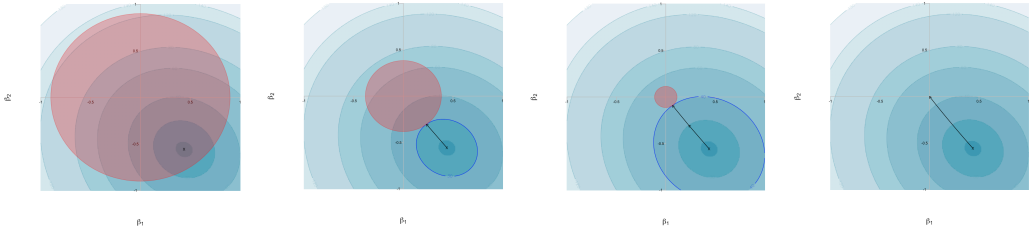
$$\min\left\{\frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\top\beta)^2\right\} \text{ s.t. } \sum_{j=1}^{k}|\beta_j| \leq r_\lambda$$

or $$\min_{\beta:\ |\beta|^\top\mathbf{1}\leq r_\lambda}\left\{\frac{1}{2}(\boldsymbol{y}-\boldsymbol{X}\beta)^\top(\boldsymbol{y}-\boldsymbol{X}\beta)\right\},$$

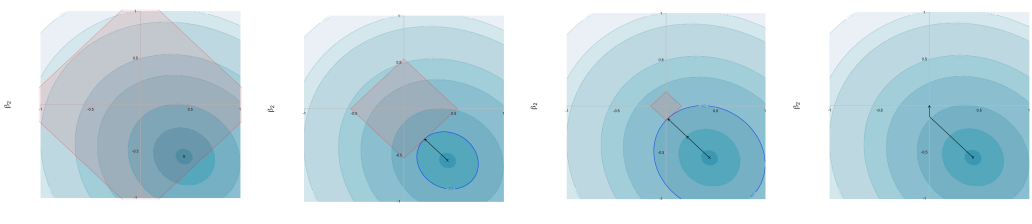
with λ large enough, i.e. $\widehat{\beta}_\lambda^{\text{lasso}} \to \mathbf{0}$

Lasso et Ridge

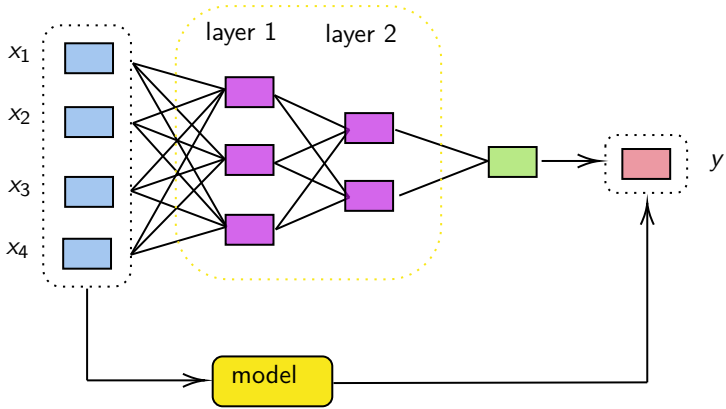
- Ridge regression



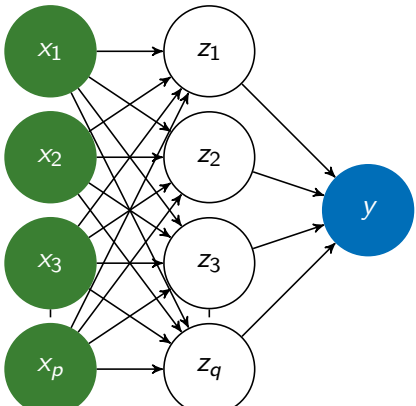
- Lasso regression



Neural Networks



Neural Networks



$$z_k = g\left(\sum_{j=1}^{p} b_{k,j}^{(1)} x_j\right)$$

so that $\mathbf{z}_{1\times q} = g(\mathbf{x}_{1\times p}\mathbf{B}^{(1)}_{p\times q})$

$$\hat{f}(\mathbf{x}) = \hat{y} = h(\mathbf{z}_{1\times q}\mathbf{B}^{(2)}_{q\times 1})$$

...

$$\hat{f}(\mathbf{x}) = h(g(\mathbf{x}_{1\times p}\mathbf{B}^{(1)}_{p\times q})\mathbf{B}^{(2)}_{q\times 1})$$

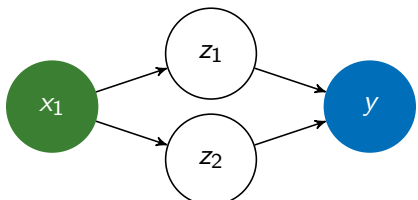
Neural Networks

Activition functions g, h,

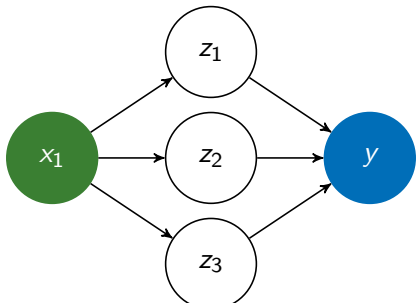
$$\begin{cases} \text{sigmoid (logistic) } h(z) = (1+\exp[-z])^{-1} \text{ with derivative } h' = h(1-h) \\ \text{hyperbolic tangent } h(z) = \tanh(z) \text{ with derivative } h' = 1 - h^2 \\ \text{ReLU (rectified linear unit) } h(z) = z_+ = z\mathbf{1}_{z>0} \text{ with derivative } h'(z) = \mathbf{1}_{z>0} \\ \text{GELU (Gaussian error linear unit) } h(z) = z\varphi(z) \text{ with derivative } h'(z) = \varphi(z) + z\varphi'(z) \end{cases}$$

where $\varphi(z) = \dfrac{1}{\sqrt{2\pi}} \exp\left(\dfrac{-z^2}{2}\right)$, $z \in \mathbb{R}$.

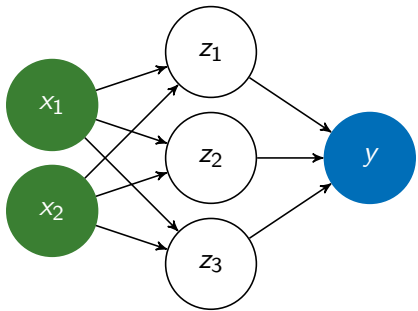
Neural Networks



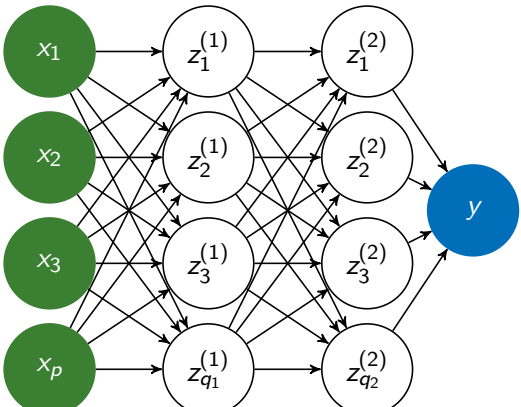
Neural Networks



Neural Networks



Neural Networks



$$z_k^{(1)} = g_1\left(\sum_{j=1}^{p} b_{k,j}^{(1)} x_j\right)$$

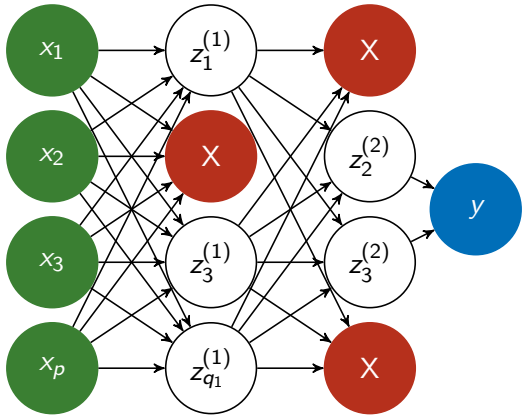
so that $\mathbf{z}_{1\times q_1}^{(1)} = g(\mathbf{x}_{1\times p}\mathbf{B}_{p\times q_1}^{(1)})$ and

$$z_k^{(2)} = g_2\left(\sum_{j=1}^{q_1} b_{k,j}^{(2)} z_j^{(1)}\right)$$

so that $\mathbf{z}_{1\times q_2}^{(2)} = g_2(\mathbf{z}_{1\times q_1}^{(1)}\mathbf{B}_{q_1\times q_2}^{(2)})$

$$\hat{f}(\mathbf{x}) = h(g_2(g_1(\mathbf{x}_{1\times p}\mathbf{B}_{p\times q_1}^{(1)})\mathbf{B}_{q_1\times q_2}^{(2)})\mathbf{B}_{q_2\times 1}^{(3)})$$

Neural Networks Dropout

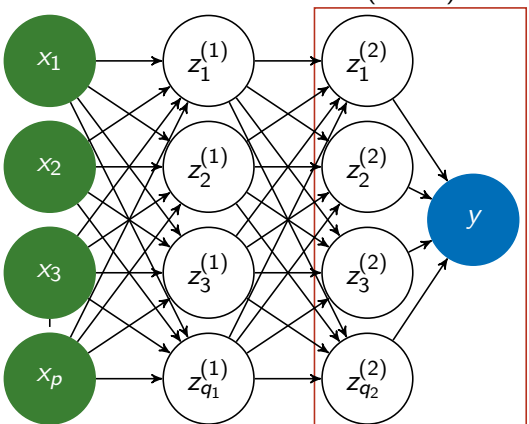


Feed-Forward Neural Network Architecture

> **Feedforward neural network**
> Feedforward refers to recognition-inference architecture of neural networks. Artificial neural network architectures are based on inputs multiplied by weights to obtain outputs (inputs-to-output): feedforward. Recurrent neural networks, or neural networks with loops allow information from later processing stages to feed back to earlier stages for sequence processing. However, at every stage of inference a feedforward multiplication remains the core, essential for backpropagation or backpropagation through time. Thus neural networks cannot contain feedback like negative feedback or positive feedback where the outputs feed back to the very same inputs and modify them, because this forms an infinite loop which is not possible to rewind in time to generate an error signal through backpropagation. W

Feed-Forward Neural Network Architecture

Feed-forward neural networks (FNNs)



feature extractor of depth 2

$$\boldsymbol{x} \mapsto \boldsymbol{z}^{(2)}(\boldsymbol{x}) \in \mathbb{R}^{q_2}$$

$$\mu(\boldsymbol{x}) = g^{-1}(\boldsymbol{z}^{(2)}(\boldsymbol{x})^\top \boldsymbol{b}^{(2)})$$

$\boldsymbol{b}^{(2)}$ is the output/readout parameter

Feed-Forward Neural Network Architecture

Universal approximation theorems are at the core of the great success of Neural Networks.

> **Universal approximation theorem**
> universal approximation theorems are theorems of the following form: Given a family of neural networks, for each function f from a certain function space, there exists a sequence of neural networks $\phi_1, \phi_2, \cdots$ from the family, such that $\phi_n \to f$ according to some criterion. The most popular version states that feedforward networks with non-polynomial activation functions are dense in the space of continuous functions between two Euclidean spaces, with respect to the compact convergence topology. W

Feed-Forward Neural Network Architecture

The main universality theorem statement says that any compactly supported continuous (regression) function can be approximated arbitrarily well by a suitable (and sufficiently large) Neural Networks.
This approximation can be w.r.t. different norms and the assumptions for such a statement to hold are comparably weak, e.g., the sigmoid activation function leads to a class of Neural Networks that are universal.

Feed-Forward Neural Network Architecture

> **Backpropagation**
> In machine learning, backpropagation[1] is a gradient estimation method commonly used for training a neural network to compute its parameter updates. It is an efficient application of the chain rule to neural networks. Backpropagation computes the gradient of a loss function with respect to the weights of the network for a single input–output example, and does so efficiently, computing the gradient one layer at a time, iterating backward from the last layer to avoid redundant calculations of intermediate terms in the chain rule; this can be derived through dynamic programming. W

"The term back-propagation is often misunderstood as meaning the whole learning algorithm for multilayer neural networks. Backpropagation refers only to the method for computing the gradient, while other algorithms, such as stochastic gradient descent, is used to perform learning using this gradient," Goodfellow et al. (2016)

Feed-Forward Neural Network Architecture

Convolutional neural network

A convolutional neural network (CNN) is a regularized type of feedforward neural network that learns features via filter (or kernel) optimization. This type of deep learning network has been applied to process and make predictions from many different types of data including text, images and audio. W

Recurrent neural network

Recurrent neural networks (RNNs) are a class of artificial neural networks designed for processing sequential data, such as text, speech, and time series,[1] where the order of elements is important. Unlike feedforward neural networks, which process inputs independently, RNNs utilize recurrent connections, where the output of a neuron at one time step is fed back as input to the network at the next time step. This enables RNNs to capture temporal dependencies and patterns within sequences. W

Feed-Forward Neural Network Architecture

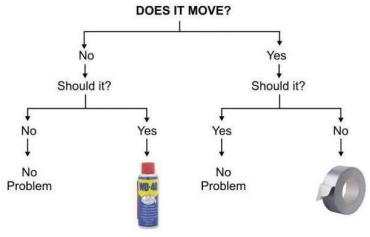
Autoencoder

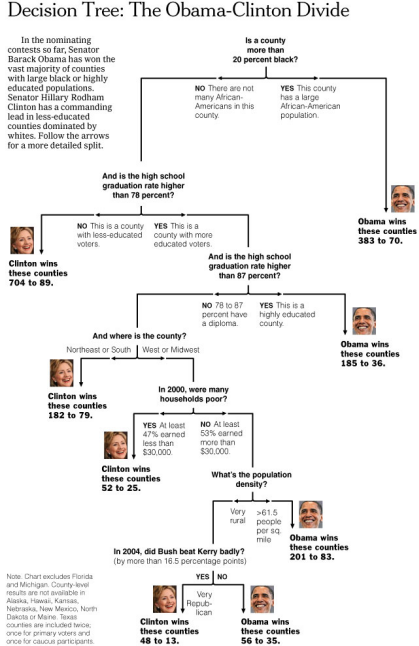
An autoencoder is a type of artificial neural network used to learn efficient codings of unlabeled data (unsupervised learning). An autoencoder learns two functions: an encoding function that transforms the input data, and a decoding function that recreates the input data from the encoded representation. The autoencoder learns an efficient representation (encoding) for a set of data, typically for dimensionality reduction, to generate lower-dimensional embeddings for subsequent use by other machine learning algorithms. W

Tree Terminology

- The final regions obtained are called terminal nodes or, more often, leaves of the tree.
- The other points where the splits occur are internal nodes of the tree.
- The segments of the tree that connect the nodes are branches.

via New York Times (2008)





Decision Trees

We must therefore subdivide (partition) $\mathcal{X}$ into separate disjoint regions,

$$\forall i,j,\ i\neq j \text{ (i.e. } R_i\neq R_j)\Rightarrow R_i\cap R_j=\emptyset) \text{ and } \bigcup_{i=1}^{m} R_i=\mathcal{X}.$$

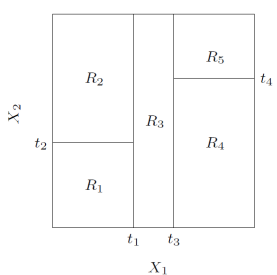
It can take the form:

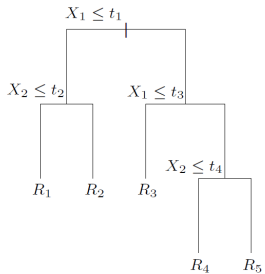
$$\hat{Y}_i=\begin{cases} c_1 & \text{if } \mathbf{x}_i\in R_1\\ c_2 & \text{if } \mathbf{x}_i\in R_2\\ \dots & \\ c_m & \text{if } \mathbf{x}_i\in R_m \end{cases} \text{ or for an arbitrary point } \mathbf{x},\ \hat{Y}=\begin{cases} c_1 & \text{if } \mathbf{x}\in R_1\\ c_2 & \text{if } \mathbf{x}\in R_2\\ \dots & \\ c_m & \text{if } \mathbf{x}\in R_m \end{cases}$$

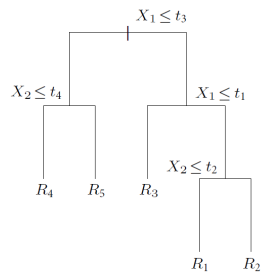
or can be presented as an additive model:

$$\hat{Y}=\sum_{j=1}^{m} c_j\mathbf{1}_{\mathbf{x}\in R_j}$$

Decision Trees







(from Friedman et al. (2001))

Objective

For a regression problem, we often work with the sum of squared errors (SSE) or the mean square error (MSE).

$$\text{MSE} = \sum_{j=1}^{J} \sum_{i:\mathbf{X}_i \in R_j} \left(y_i - \widehat{y}_{R_j} \right)^2,$$

where $widehaty_{R_j}$ is the predicted value in the R_j region. We then look for a partition $R_1, \ldots, R_J$ that minimizes the SSE.

For a classification problem, note that

$$\text{MSE}_j = \sum_{i:\mathbf{X}_i \in R_j} \left(y_i - \widehat{y}_{R_j} \right)^2 = n_{0,j}(0 - \widehat{y}_{R_j})^2 + n_{1,j}(1 - \widehat{y}_{R_j})^2$$

Objective

$$\max\{\text{MSE}_j\} = n_{0,j}\left(\frac{n_{0,j}}{n_{0,j}+n_{1,j}}\right)^2 + n_{1,j}\left(\frac{n_{0,j}}{n_{1,j}+n_{1,j}}\right)^2 = \frac{n_{0,j}n_{1,j}}{n_{0,j}+n_{1,j}}$$

so that

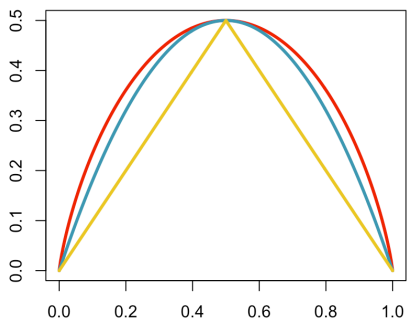
$$MSE = \sum_{j=1}^{J} \frac{n_{0,j}n_{1,j}}{n_{0,j}+n_{1,j}} = \sum_{j=1}^{J} n_j \cdot p_{1,j}(1-p_{i,j})$$

This is Gini impurity index.

Objective

Formally, impurity is a function ψ, with ψ positive, symmetrical ($\psi(p) = \psi(1-p)$), minimal in 0 (and 1).

- Misclassification, $\psi(p) = 1 - \max\{p, 1-p\}$
- Gini, $\psi(p) = p(1-p)$
- (cross) entropy, $\psi(p) = -p\log p - (1-p)\log(1-p)$



Objective

- Again, if we consider all possible partitions of the space of explanatory variables, we'll have to consider too many possibilities.
- Instead, we'll use a top-down greedy approach via recursive binary splitting.
- Descending algorithm: we start with all observations in the same class (root of the tree) and divide the space into smaller and smaller regions.
- Greedy algorithm: optimization is performed at each step without looking back or forward.

Recursive Binary Division

First step: we select the explanatory variable X_j and the point c such that the division into two regions

$$R_1(j,c) = \{X_1, \ldots, X_J | X_j < c\}\} R_2(j,c) \qquad = \{X_1, \ldots, X_J | X_j \geq c\}$$

leads to the largest positive reduction of the objective function. If the sum of squared errors has been chosen, we then seek to minimize

$$\sum_{i:\mathbf{X}_i \in R_1(j,c)} \left(Y_i - \widehat{Y}_{R_1(j,c)} \right)^2 + \sum_{i:\mathbf{X}_i \in R_2(j,c)} \left(Y_i - \widehat{Y}_{R_2(j,c)} \right)^2.$$

(or any impurity function)

Recursive Binary Division

Next steps: in step k, we repeat the procedure described in the first step for each of the regions created in step $k-1$, i.e., for each of the regions r, we must identify the explanatory variable X_j and the point c which will minimize the objective function after the division given by

$$\{\mathbf{X} \in r \mid X_j < c\} \text{ and } \{\mathbf{X} \in r \mid X_j \geq c\}.$$

The procedure is stopped when a certain criterion is reached.
In the absence of stopping criteria, the resulting tree will have n leaves *to* overfit.

Hierarchical Sequential Construction

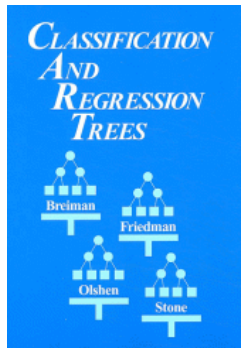
Clasiffication = grouping individuals into a limited number of classes

These classes are built up as we go along.

→ group together ifmilar individuals, séparer des individus ayant des caractéristiques proches.
History: dates from the 1960s,

then Breiman et al. (1984)

tools that have become popular in machine learning



Hierarchical Sequential Construction

Downward classification :

we select the most related of the explanatory variables

to the variable to be explained Y,

.3cm

$\rightarrow$ gives a first division of the sample

.3cm

we repeat in each class

.3cm

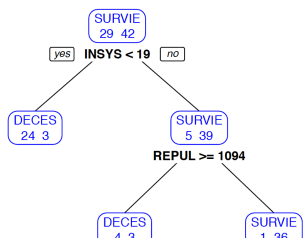
$\rightarrow$ each class must be the most homogeneous positive, in Y.

Difference from logistic regression

.3cm

sequential use of explanatory variables

.3cm



Cutting $\mathcal{X} \subset \mathbb{R}$

$Y \in \{0,1\}$ and $X \in \mathbb{R}$: cut according to threshold s,

$$\begin{cases} \tilde{X} = L \text{ if } X \leq s \\ \tilde{X} = R \text{ if } X > s \end{cases}$$

	$\tilde{X} = L$ $X \leq s$	$\tilde{X} = R$ $X > s$	
$Y = 0$	$n_{L,0}$	$n_{R,0}$	$n_{\cdot,0}$
$Y = 1$	$n_{L,1}$	$n_{R,1}$	$n_{\cdot,1}$
	$n_{L,\cdot}$	$n_{R,\cdot}$	n

Gini $\text{gini}(Y|\tilde{X}) = \sum_{x \in \{L,R\}} \frac{n_{x,\cdot}}{n} \sum_{y \in \{0,1\}} \frac{n_{x,y}}{n_{x,\cdot}} \left(1 - \frac{n_{x,y}}{n_{x,\cdot}}\right)$

Cutting $\mathcal{X} \subset \mathbb{R}$

$Y \in \{0,1\}$ and $X \in \mathbb{R}$: we cut according to a threshold s,

$$\begin{cases} \tilde{X} = L \text{ if } X \leq s \\ \tilde{X} = R \text{ if } X > s \end{cases}$$

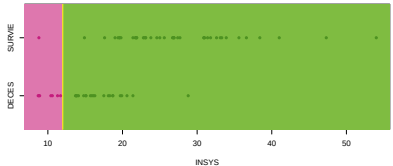
	$\tilde{X} = L$ $X \leq s$	$\tilde{X} = R$ $X > s$	
$Y = 0$	$n_{L,0}$	$n_{R,0}$	$n_{\cdot,0}$
$Y = 1$	$n_{L,1}$	$n_{R,1}$	$n_{\cdot,1}$
	$n_{L,\cdot}$	$n_{R,\cdot}$	n

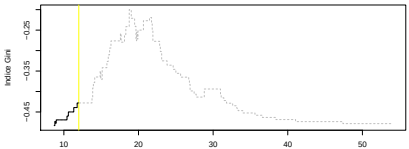
Entropy $\text{entropy}(Y|X) = -\sum_{x \in \{L,R\}} \frac{n_{x,\cdot}}{n} \sum_{y \in \{0,1\}} \frac{n_{x,y}}{n_{x,\cdot}} \log\left(\frac{n_{x,y}}{n_{x,\cdot}}\right)$

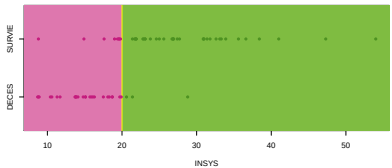
Cutting $\mathcal{X} \subset \mathbb{R}$

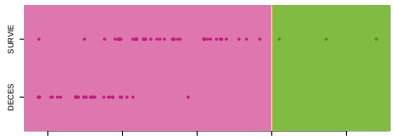
Gini index

$$\sum_{x \in \{A,B\}} \frac{n_{x,\cdot}}{n} \sum_{y \in \{0,1\}} \frac{n_{x,y}}{n_{x,\cdot}} \left(1 - \frac{n_{x,y}}{n_{x,\cdot}}\right)$$







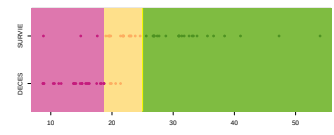


Cutting $\mathcal{X} \subset \mathbb{R}$

Fix threshold s, and find the second best cut

$$A = (-\infty, s_2] \quad B = (s_2, s] \quad C = (s, \infty)$$

	$\tilde{X} = A$ $X \leq s_2$	$\tilde{X} = B$ $X \in (s_2, s]$	$\tilde{X} = C$ $X > s$	
$Y = 0$	$n_{A,0}$	$n_{B,0}$	$n_{C,0}$	$n_{\cdot,0}$
$Y = 1$	$n_{A,1}$	$n_{B,1}$	$n_{C,1}$	$n_{\cdot,1}$
	$n_{A,\cdot}$	$n_{B,\cdot}$	$n_{C,\cdot}$	n



Using Gini index

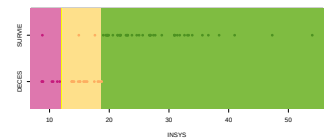
$$\sum_{x \in \{A,B,C\}} \frac{n_{x,\cdot}}{n} \sum_{y \in \{0,1\}} \frac{n_{x,y}}{n_{x,\cdot}} \left(1 - \frac{n_{x,y}}{n_{x,\cdot}}\right)$$

Cutting $\mathcal{X} \subset \mathbb{R}$

Fix threshold s, then seek for the second one

$$A = (-\infty, s] \quad B = (s, s_2] \quad C = (s_2, \infty)$$

	$\tilde{X} = A$ $X \leq s$	$\tilde{X} = B$ $X \in (s, s_2]$	$\tilde{X} = C$ $X > s_2$	
$Y = 0$	$n_{A,0}$	$n_{B,0}$	$n_{C,0}$	$n_{\cdot,0}$
$Y = 1$	$n_{A,1}$	$n_{B,1}$	$n_{C,1}$	$n_{\cdot,1}$
	$n_{A,\cdot}$	$n_{B,\cdot}$	$n_{C,\cdot}$	n

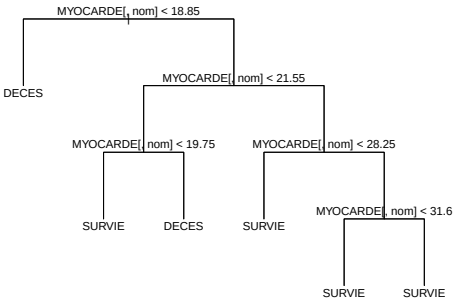


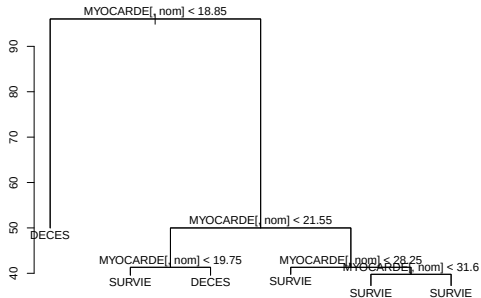
Gini index

$$\sum_{x \in \{A,B,C\}} \frac{n_{x,\cdot}}{n} \sum_{y \in \{0,1\}} \frac{n_{x,y}}{n_{x,\cdot}} \left(1 - \frac{n_{x,y}}{n_{x,\cdot}}\right)$$

Prunning Trees

Step 1 Construction of the tree by a recursive process of binary divisions
Step 2 Pruning the tree, removing branches that are too empty or not representative enough $\rightarrow$ need a pruning criterion (entropy gain)





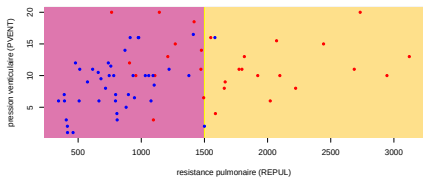
Multiple Numeric Variables

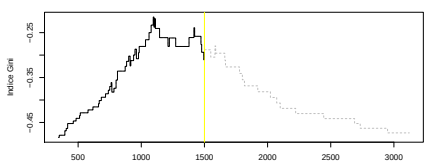
$Y \in \{0,1\}$ and $X_1, X_2 \in \mathbb{R}$: cut X_1 based on threshold s,

$$\begin{cases} \tilde{X} = A \text{ if } X_1 \leq s \\ \tilde{X} = B \text{ if } X_1 > s \end{cases}$$

	$\tilde{X} = A$ $X_1 \leq s$	$\tilde{X} = B$ $X_1 > s$	
$Y = 0$	$n_{A,0}$	$n_{B,0}$	$n_{\cdot,0}$
$Y = 1$	$n_{A,1}$	$n_{B,1}$	$n_{\cdot,1}$
	$n_{A,\cdot}$	$n_{B,\cdot}$	n

$$\sum_{x \in \{A,B\}} \frac{n_{x,\cdot}}{n} \sum_{y \in \{0,1\}} \frac{n_{x,y}}{n_{x,\cdot}} \left(1 - \frac{n_{x,y}}{n_{x,\cdot}}\right)$$





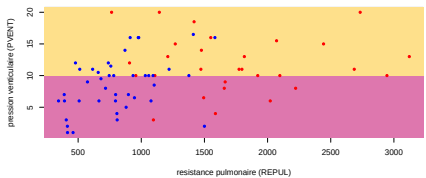
Multiple Numeric Variables

$Y \in \{0, 1\}$ and $X_1, X_2 \in \mathbb{R}$: cut X_2 based on threshold s,

$$\begin{cases} \tilde{X} = A \text{ if } X_2 \leq s \\ \tilde{X} = B \text{ if } X_2 > s \end{cases}$$

	$\tilde{X} = A$ $X_2 \leq s$	$\tilde{X} = B$ $X_2 > s$	
$Y = 0$	$n_{A,0}$	$n_{B,0}$	$n_{\cdot,0}$
$Y = 1$	$n_{A,1}$	$n_{B,1}$	$n_{\cdot,1}$
	$n_{A,\cdot}$	$n_{B,\cdot}$	n

$$\sum_{x \in \{A,B\}} \frac{n_{x,\cdot}}{n} \sum_{y \in \{0,1\}} \frac{n_{x,y}}{n_{x,\cdot}} \left(1 - \frac{n_{x,y}}{n_{x,\cdot}}\right)$$



Categorical Variables

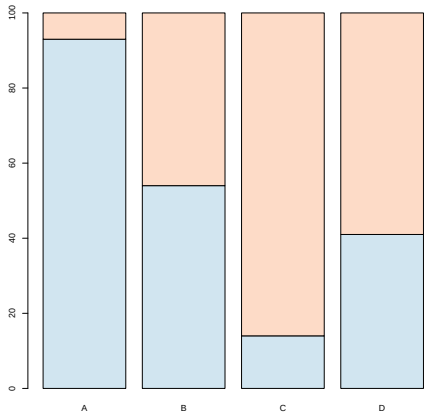
X taking values $\{a, b, c, d\}$.

Instead of making a tree by successive cuttings, we can make a tree by successive grouping.

$$\{a, b, c, d\}$$

$$\begin{cases} \{(a, b), c, d\} & \{(a, d), b, c\} & \{(a, d), b, c\} \\ \{(b, c), a, d\} & \{(b, d), a, c\} & \{(c, d), a, b\} \end{cases}$$

$$\{(b, c, a), d\} \{(b, c, d), a\} \{(b, c), (a, d)\}$$



Categorical Variables

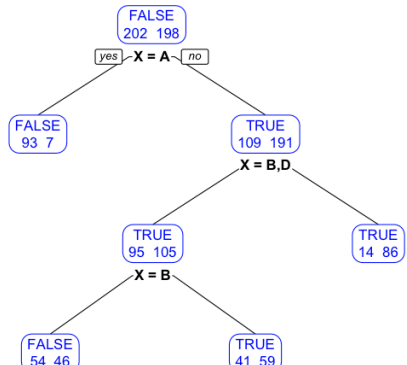
X taking values $\{a, b, c, d\}$.

Instead of making a tree by successive cuttings, we can make a tree by successive regrouping

$$\{a, b, c, d\}$$

$$\begin{cases} \{(a, b), c, d\} & \{(a, d), b, c\} & \{(a, d), b, c\} \\ \{(b, c), a, d\} & \{(b, d), a, c\} & \{(c, d), a, b\} \end{cases}$$

$$\{(b, c, a), d\} \{(b, c, d), a\} \{(b, c), (a, d)\}$$



Categorical Variables

X taking values $\{a, b, c, d\}$.

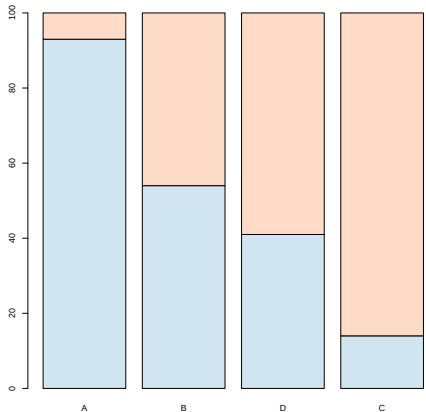
Classically, order the modalities

$$\{a, b, d, c\}$$

then find the optimum cut

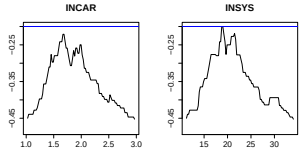
$$\{(a), (b, d, c)\} \ \{(a, b), (d, c)\} \ \{(a, b, d), (c)\}$$

$$\{(a), (b, d), (c)\} \ \{(a), (b), (d, c)\}$$



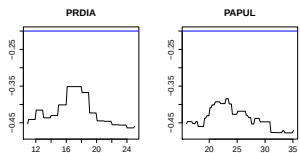
Implementation

$$N_L\colon \{x_{i,j} \leq s\} \quad N_R\colon \{x_{i,j} > s\}$$

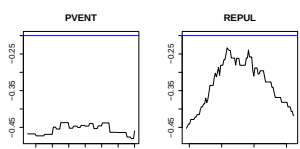


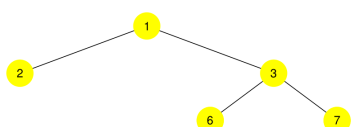
Solve $\max_{j\in\{1,\cdots,k\},s} \{\mathcal{I}(N_L, N_R)\}$

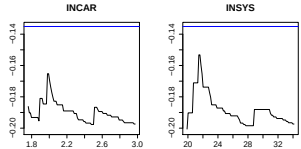
$\longleftarrow$ first split

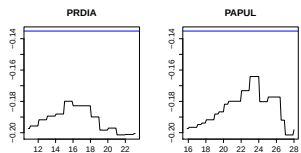


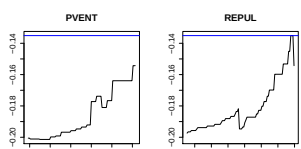
second split $\longrightarrow$







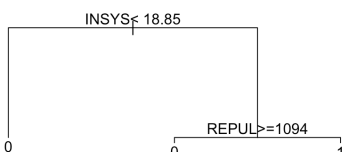




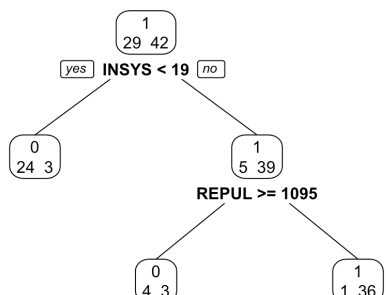
Trees with R

```
> loc = "http://freakonomandrics.free.fr/saporta.csv"
> myocarde = read.table(loc, header=TRUE,sep=";")
> levels(myocarde$PRONO) = c("0","1")
```

```
> library(rpart)
> cart = rpart(PRONO~., data=
    myocarde)
> plot(cart)
> text(cart)
```



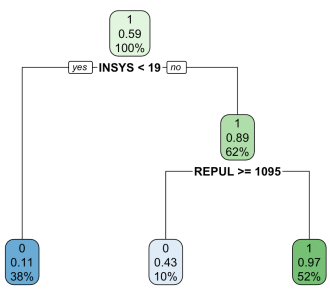

```
> library(rpart.plot)
> prp(cart, type=2, extra
    =1)
```



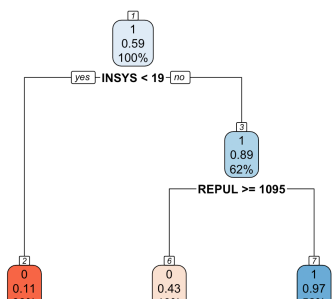
Trees with R

```
> print(cart, digits = 2)
1) root 71 29 1 (0.408 0.592)
  2) INSYS< 19 27  3 0 (0.889 0.111) *
  3) INSYS>=19 44  5 1 (0.114 0.886)
    6) REPUL>=1094.5 7  3 0 (0.571 0.429) *
    7) REPUL< 1094.5 37  1 1 (0.027 0.973) *
```

```
> rpart.plot(cart)
```



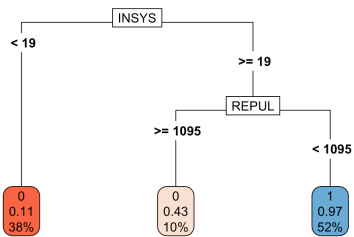

```
> rpart.plot(cart, box.palandte="RdBu", nn=TRUE
    )
```



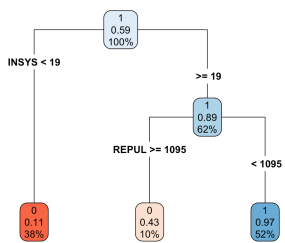
Trees with R

```
> rpart.rules(cart, extra = 4, cover = TRUE)
PRONO  0   1                                          cover
  0 [.89 .11] when INSYS <  19                          38%
  0 [.57 .43] when INSYS >= 19 & REPUL >= 1095          10%
  1 [.03 .97] when INSYS >= 19 & REPUL <  1095          52%
```

```
> rpart.plot(cart, box.palandte="
    RdBu", type=5)
```



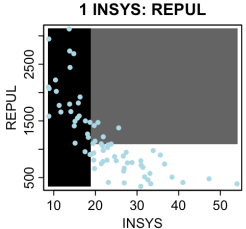

```
> rpart.plot(cart, box.palandte="
    RdBu", type=4)
```

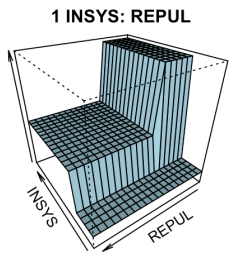


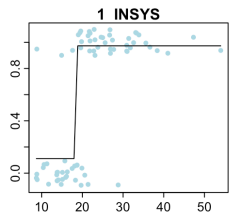
Trees with R

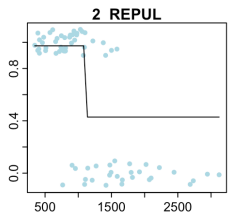
```
> library(plotmo)
> plotmo(cart, type = "prob",
    nresponse = "1")
```

```
> plotmo(cart, type = "prob",
    nresponse = "1", type2 = "image",
    ngrid2 = 200)
```





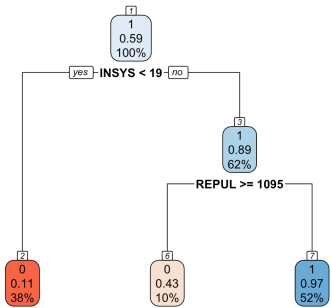




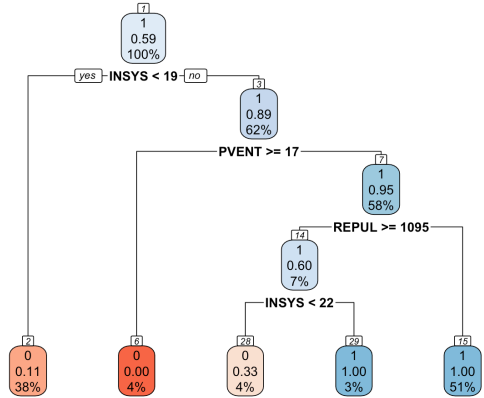

```
> path2root = function(node){
  if(node == 1) node
  else c(node, path.to.root(node %/% 2))}
```

Trees with R

```
> cart = rpart(PRONO~.,
    data=myocarde)
> rpart.plot(cart, box.
    palandte="RdBu", nn=
    TRUE)
```



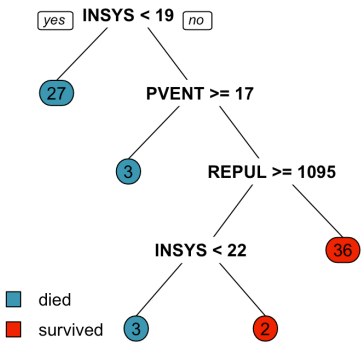

```
> cart = rpart(PRONO~.,data=
    myocarde, minsplit = 5,
  cp = 0.01)
rpart.plot(cart, box.palandte="
    RdBu", nn=TRUE)
```

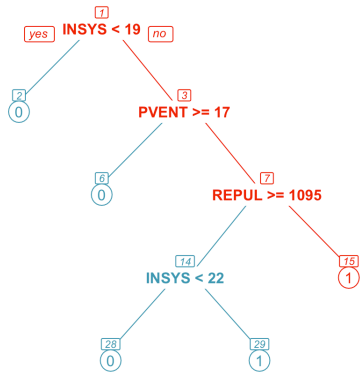


Trees with R

```
> boxcols =c("red","blue")
    [cart$frame$yval]
> prp(cart, faclen = 0,
    node.fun=only_count,
    box.col = boxcols)
```

```
> node = 15
> nodes = as.numeric(row.names(cart$frame))
> cols = ifelse(nodes %in%
path2root(node),"red","blue")
prp(cart, nn = TRUE, col = cols, branch.col =
    cols, split.col = cols)
```





Stopping criteria

- Stopping criteria often include a minimum number of observations in each region.
- Using as Stopping criteria the fact of having a decrease in the objective function that exceeds a certain threshold is generally not a good idea.
- Even if an appropriate Stopping criteria is selected, there's a good chance that the tree will lead to over-fitting, i.e. it will perform (too) well on a training database but will be poor on an evaluation database.

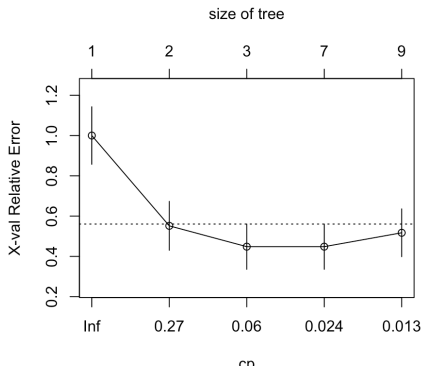
Formally, to find out if a leaf $\{N\}$ is cut into two $\{N_L, N_R\}$, we measure the variation in impurity

$$\Delta\mathcal{I}(N_L, N_R) = \mathcal{I}(N) - \mathcal{I}(N_L, N_R) = \mathcal{I}(N) - \left(\frac{n_L}{n}\mathcal{I}(N_L) + \frac{n_R}{n}\mathcal{I}(N_R)\right)$$

Stopping criteria

Stop if $\Delta\mathcal{I}(N_L, N_R)/\mathcal{I}(N)$ exceeds `cp` (complexity paramander, par défaut 1%).

```
> cart = rpart(PRONO ~ ., data =
    myocarde,
                minsplit=3)
> plotcp(cart)
```



Stopping criteria

```
> prune(cart, cp=0.06)
node), split, n, loss, yval, (yprob) *= terminal node

1) root 71 29 SURVIE (0.40845070 0.59154930)
  2) INSYS< 18.85 27  3 DECES (0.888889 0.111111) *
  3) INSYS>=18.85 44  5 SURVIE (0.113636 0.886363)
    6) PVENT>=17.25 3  0 DECES (1.000000 0.000000) *
    7) PVENT< 17.25 41  2 SURVIE (0.048780 0.951219) *
```

Pruning

- A better strategy is to (1) build a very large starting tree $\mathcal{T}_0$ and (2) cut some branches of the tree.
- We'll try to identify the $\mathcal{T}$ *subtree* $\subset \mathcal{T}_0$ that minimizes

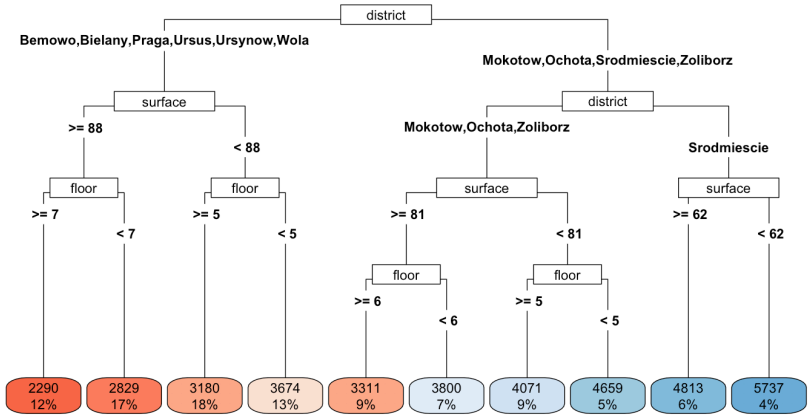
$$\sum_{m=1}^{|\mathcal{T}|} \sum_{i:\mathbf{X}_i \in R_m} \left(Y_i - \widehat{Y}_{R_m} \right)^2 + \alpha |\mathcal{T}|,$$

  where α is a (hyper-)complexity parameter and $|\mathcal{T}|$ represents the number of leaves in the $\mathcal{T}$ subtree.
- Similar to Ridge/Lasso regression: compromise between bias and variance, between precision and parsimony.
- The complexity parameter can be estimated by cross-validation.

Regression Trees, $y \in \mathbb{R}$

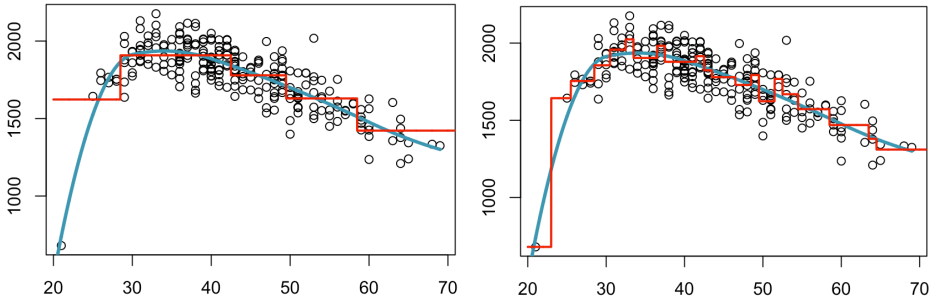
```
> library(DALEX)
> tree = rpart(m2.price~., data=apartments)
> rpart.plot(tree, box.palandte="RdBu", type=5)
```

Regression Trees, $y \in \mathbb{R}$



Regression Trees, $y \in \mathbb{R}$

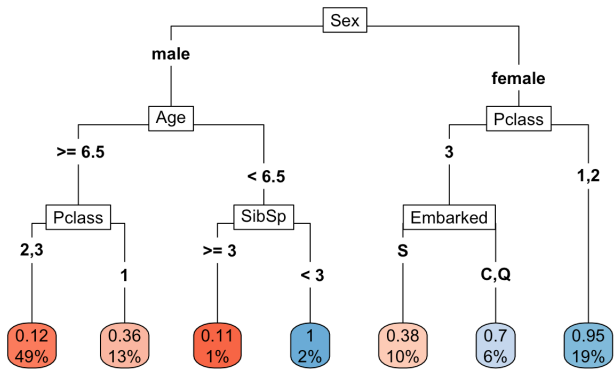
```
> tree = rpart(y~x, data=base)
> predict(tree, newdata = data.frame(x=x))
```



Classification Trees, $y \in \{0, 1\}$

```
> loc = "http://freakonomandrics.free.fr/titanic.RData"
> download.file(loc, "titanic.RData")
> load("titanic.RData")
> cart = rpart(Survived~.,data=base[,1:7])
> rpart.plot(cart, box.palandte="RdBu", type=5)
```

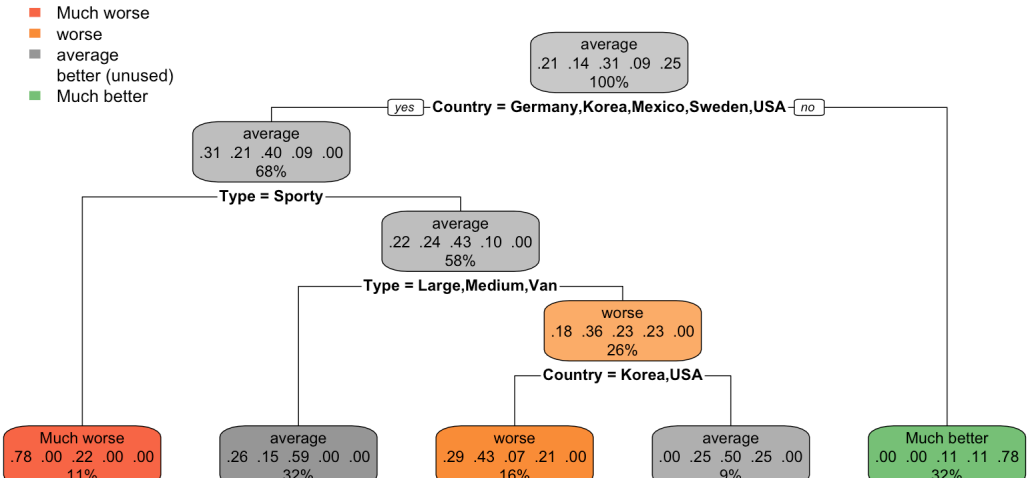
Classification Trees, $y \in \{0, 1\}$



Classification Trees, $y \in \{A, B, C, D\}$

```
> multi.class.model <- rpart(Reliability~., data = cu.summary)
> rpart.plot(multi.class.model)
```

Classification Trees, $y \in \{A, B, C, D\}$



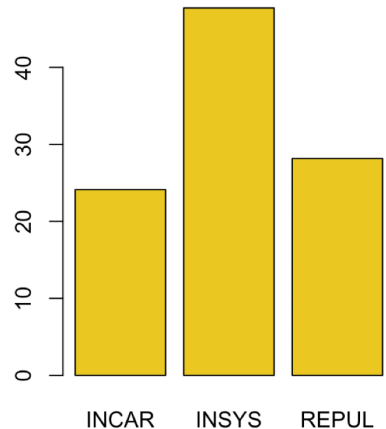
Tree Stability

Consider

```
> variable = rep(NA,10000)
> for(i in 1:10000){
+ tree = rpart(PRONO~., data=
    myocarde[sample(1:71,size=47),])
+ variable[i] = as.character(
    tree$frame[1,"var"])
+ }
> table(variable)/100
INCAR INSYS REPUL
24.12 47.72 28.16
```

The variable used in the first node is

- INSYS (47.7%)
- REPUL (28.2%)
- INCAR (24.1%)

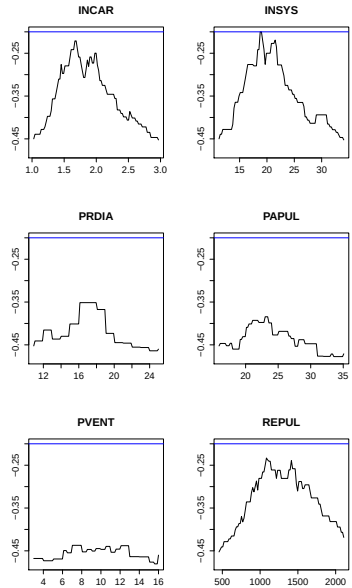


Variable Importance

We had already intuited the importance of these three variables

```
> cart = rpart(PRONO~., myocarde)
> split = summary(cart)$splits
> split
       count ncat    improve
INSYS     71   -1 0.58621312
REPUL     71    1 0.55440034
INCAR     71   -1 0.54257020
PRDIA     71    1 0.27284114
PAPUL     71    1 0.20466714
```

and we can calculate the same quantities at each node



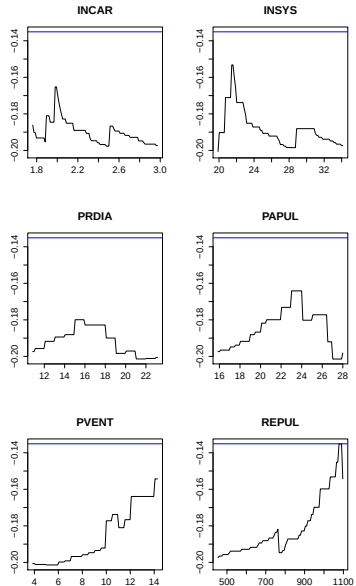
Variable Importance

for the second node

```
> split
       count ncat     improve
REPUL     27    1 0.18181818
PVENT     27   -1 0.10803571
PRDIA     27    1 0.10803571
PAPUL     27    1 0.10803571
INCAR     27    1 0.04705882
```

etc

```
> cart$variable.importance
 INSYS  REPUL  INCAR  PAPUL  PRDIA
    FRCAR  PVENT
20.113 19.132 15.895  5.959  5.214
    3.725  0.498
```



Decision Trees

... it's good!

- No assumptions necessary.
- Intuitive visualization.
- Natural integration of high-degree interaction.

... it's bad!

- Procedure quick to overlearn.
- Procedure unstable.

Countering overfit through regularization

A standard technique is to prune trees by cutting off the least "profitable" branches (in terms of impurity)

Consider a decision tree, penalizing too many regions m.

Regression tree $\sum_{r=1}^{m} \sum_{x_i \in R_r} (y_i - \hat{y}_{R_r})^2 + \lambda m$

i.e., with "error $+\lambda m$".

Countering overfit through regularization

With λ very large, we prune everything, return to a single region, and find$\hat{y} = \bar{y}$. Here, no overfit.

If λ is very small, we find the standard decision tree tree solution. Without stopping rules, we can form regions with a single observation such that the prediction error (on S_{train} is 0.

Countering overfit through regularization

Although it may seem difficult to optimize the regularized objective function, there's a little trick that lets us easily re-create the equivalent of the *lasso path*.

In other words, we'll get a nested tree sequence, from a very large tree with many branches to a trunk tree with a single region.

Starting from a very large tree (say, a very small one). We'll progressively prune the branches, always choosing the one that had reduced branch diversity the least.

Countering overfit through regularization

Another perspective is to group the two regions that increase the total error the least when recombined.

Ainif, we can create a sequence of nested trees.

It could be demonstrated that this is equivalent to optimizing :
$\sum_{r=1}^{m} \sum_{x_i \in R_r} (y_i - \hat{y}_{R_r})^2 + \lambda m$ on a value sequence of λ.

Countering overfit through regularization

We need to choose between these two extremes, which we'll do using validation.

An interesting approach:

- Train a decision tree with many branches (β very small)
- We generate the intermediate tree sequence by pruning the branches, starting with those that reduce heterogeneity the least.
- Evaluate the performance of all trees on S_{val}.
- We choose the tree with the best performance on S_{val}.

Countering overfit through regularization

Algorithm 8.1 *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.

3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

   Average the results for each value of α, and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Figure 1: Algorithme 8.1. Extrait de James et al. (2013).

Countering overfit through regularization

Although this pruning feature works well, it is rarely used in practice.

It's a slow process.

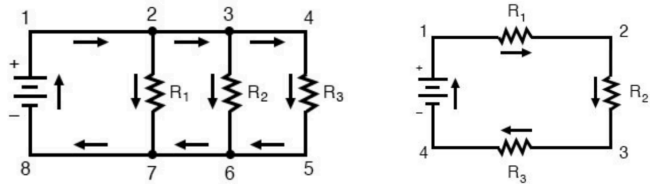
Above all, decision trees are rarely used outside the random forest and pruning is not used in forest training.

Ensemble learning

> **Ensemble learning**
> In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. W

See Mienye and Sun (2022) for a recent survey

Ensemble learning



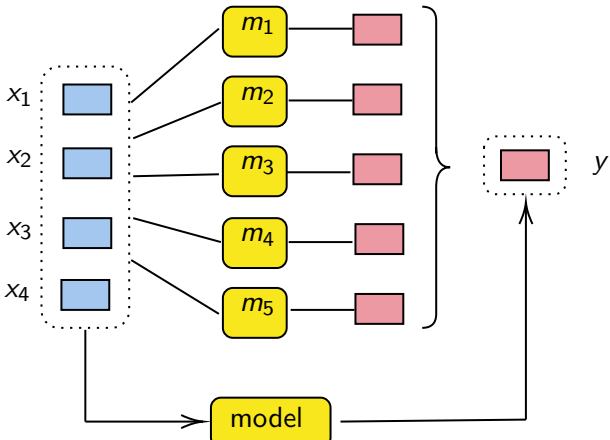
- in **parallel ensemble method**, the base learners are generated in parallel.

  when bagging, several estimators or models are built independently and then average their predictions,

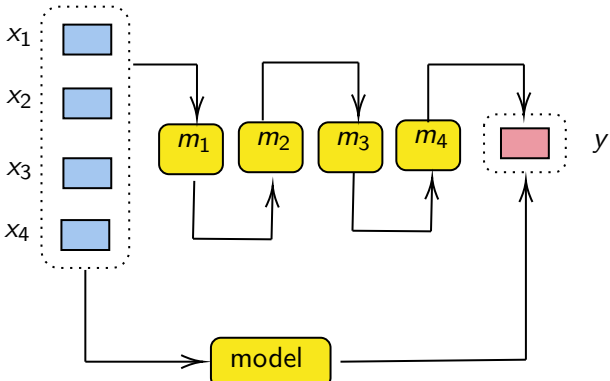
  when stacking, we multiple classifications or regression models via a meta-classifier or a meta-regressor, trained on the same training set,

- in **sequential ensemble method**, the base learners are generated sequentially. The overall performance can be boosted by weighing previously mislabeled examples with higher weight. This corresponds to boosting.

Ensemble learning (parallel learning)



Ensemble learning (series-sequential learning)



Ensemble Method

Galton (1907)guess the weight of the cow, county fair in Cornwall, England, 1906

Correct answer = 1198 lbs

787 participants, $x_1, \cdots, x_n$

- quantile 25%: 1162 lbs
- quantile 50%: 1207 lbs
- quantile 75%: 1236 lbs
- average: 1197 lbs

Pick a single prediction x_j v.s average $\overline{x}$,

$$\mathbb{E}[(x_j - t)^2]] = (\overline{x} - t)^2 + \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2$$

where t is the truth

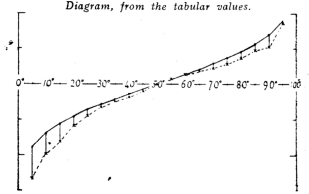
(ambiguity decomposition)

Distribution of the estimates of the dressed weight of a particular living ox, made by 787 different persons.

Degrees of the length of Array 0—100	Estimates in lbs.	Centiles: Observed deviates from 1207 lbs.	Centiles: Normal p.e. =37	Excess of Observed over Normal
5	1074	− 133	− 90	+ 43
10	1109	− 98	− 70	+ 28
15	1126	− 81	− 57	+ 24
20	1148	− 59	− 46	+ 13
q_1 25	1162	− 45	− 37	+ 8
30	1174	− 33	− 29	+ 4
35	1181	− 26	− 21	+ 5
40	1188	− 19	− 14	+ 5
45	1197	− 10	− 7	+ 3
m 50	1207	0	0	0
55	1214	+ 7	+ 7	0
60	1219	+ 12	+ 14	− 2
65	1225	+ 18	+ 21	− 3
70	1230	+ 23	+ 29	− 6
q_3 75	1236	+ 29	+ 37	− 8
80	1243	+ 36	+ 46	− 10
85	1254	+ 47	+ 57	− 10
90	1267	+ 52	+ 70	− 18
95	1293	+ 86	+ 90	− 4

q_1, q_3, the first and third quartiles, stand at 25° and 75° respectively.
m, the median or middlemost value, stands at 50°.
The dressed weight proved to be 1198 lbs.

Diagram, from the tabular values.



The continuous line is the normal curve with p.e.=37.
The broken line is drawn from the observations.
The lines connecting them, show the differences between the observed and the normal.

Ensemble Method

- **statistical interpretation**: hopefully the ensemble generalises better than a single chosen model
- **computational interpretation**: averaging can sometimes be a fast way of reaching closer to the optimal than direct optimisation
- **representational interpretation**: Averaging models of some model class can sometimes take you outside of that model class

Regression problem: averaging
Classification problem: voting (majority)

Votes

?

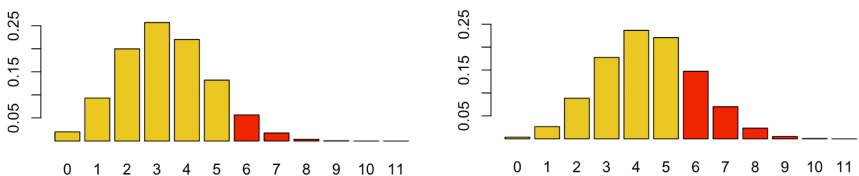
Condorcet (1785) in "Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix"

Let p denote the probability to be wrong, individually,

With n independent vote, the probability for the majority to be wrong is

$$\sum_{k\geq[(n+1)/2]} \binom{n}{k} p^k(1-p)^{n-k}$$

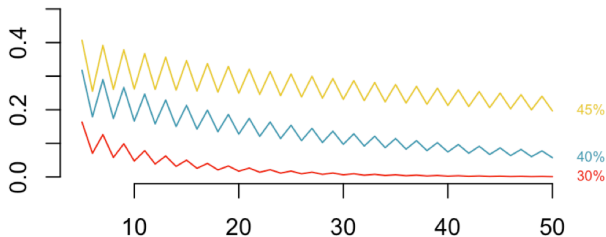
e.g. $n = 11$, $p = 30\%$ (left) and $p = 40\%$ (right)



Probability that the majority is wrong, 7.8% (left) and 24.6% (right).

Votes

Evolution of the probability that the majority is wrong, as a function of n, for various p



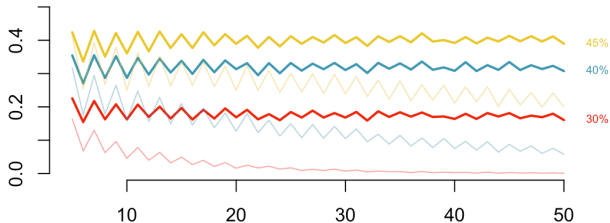
But this is valid only if votes are independent ! In ensemble learning, we want

- models to be reasonably* accurate
- models to be as independent as possible

* ensemble learning can turn a collection of poor learners (stumps (single-node decision trees) or linear) into a good one

Independence ?

with correlation among prediction, there is no real 'decrease'



- split training sample into m groups of n/m observations

  pb, groups are small, poor predictions
- split features into m groups

  pb, poor predictions if only small number of (true) predictors
- use boostrap

  pb if models are not independent enough

Overview

- Parallel ensemble methods (bagging & stacking)
  - Bagging
    In this method, models are generated using the same algorithm with random sub-samples of the dataset with the bootstrap sampling method to reduce the variance. In bootstrap sampling, some original examples appear more than once and some original examples are not present in the sample.
  - Random Forest
    In a random forest at each sample, a decision tree is used which collectively form a forest and hence, Random Forest. Rest is almost similar to a simple decision tree.
  - Stacking
    Stacking is an ensemble learning technique that combines multiple classifications or regression models via a meta-classifier or a meta-regressor. The base-level models are trained based on a complete training set, then the meta-model is trained on the outputs of the base level model as features.

Overview

- Sequential ensemble methods (Boosting Algorithm)
  - Gradient Boosted Decision Trees (GBDT)
    uses Gradient Descent Algorithm on Decision trees to reduce loss at each level. It generalizes models by allowing optimization of an arbitrary differentiable loss function. The next model is built on the errors of the previous model
  - XG Boost
    XG Boost works similar to GBDT, but it has some more features(below) which makes it more efficient. i) Regularization: Standard GBM implementation has no regularization like XGBoost, therefore it also helps to reduce overfitting. In fact, XGBoost is also known as the 'regularized boosting' technique. ii) Parallel Processing: XGBoost implements parallel processing at root nodes and becomes faster as compared to GBM. iii) High Flexibility: XGB allows for custom evaluation criteria. iv) Handling Missing Values: Can handle missing values on its own. v) Tree Pruning: XGB does

tree pruning on its own and reduces variance. vi) Built-in Cross-Validation: XGBoost allows the user to run cross-validation at each iteration of the boosting process. vii) Continue on Existing Model: User can start training an XGBoost model from its last iteration of the previous run.

- Ada Boost: each weak learner is developed as decision stumps (A stump is a tree with just a single split and two terminal nodes) that are used to classify the observations. After each classifier is trained, the classifier's weight is calculated based on its accuracy. More weight is assigned to the classifier when accuracy is more and vice-versa.
- Light GBM
  It splits the tree leaf-wise with the best fit whereas other boosting algorithms split the tree depth-wise or level-wise rather than leaf-wise and tries to get a pure leaf as soon as possible. This makes Light GBM faster than other boosting algorithms and hence the word 'Light'.

Overview

- Cat Boost "CatBoost" name comes from two words "Category" and "Boosting". CatBoost converts categorical values into numbers using various statistics on combinations of categorical features and combinations of categorical and numerical features. Hence we need not convert categorical variables to numerical like in other models.

Bayesian model averaging

Consider m models, suppose that one is the true one, but we don't know which one...
Let T denote the choice of the model, $T \in \{1, 2, \cdots, m\}$,

$$\mathbb{P}[Y = 1|\boldsymbol{X} = \boldsymbol{x}] = \sum_{t=1}^{m} \mathbb{P}[Y = 1, T = t|\boldsymbol{X} = \boldsymbol{x}]$$

$$\mathbb{P}[Y = 1|\boldsymbol{X} = \boldsymbol{x}] = \sum_{t=1}^{m} \underbrace{\mathbb{P}[Y = 1|\boldsymbol{X} = \boldsymbol{x}, T = t]}_{\widehat{y}^{(t)}} \underbrace{\mathbb{P}[T = t|\boldsymbol{X} = \boldsymbol{x}]}_{\omega_t}$$

which is a weighted average of predictions (or posterior class prediction), where weights

$$\omega_t = \mathbb{P}[T = t|\boldsymbol{X} = \boldsymbol{x}] \propto \mathbb{P}[\boldsymbol{X} = \boldsymbol{x}|\, T = t]$$

are likelihoods of models.

Stacking

Estimating model likelihood can be complicated
we can learn weights using a regression
(where individual model outputs are treated as features)

```
url = "http://freakonometrics.free.fr/german_credit.csv"
credit = read.csv(url, header = TRUE, sep = ",")
F = c(1,2,4,5,7,8,9,10,11,12,13,15,16,17,18,19,20)
for(i in F) credit[,i]=as.factor(credit[,i])
```

create a training and a testing datasets

```
set.seed(123)
i_test = sample(1:nrow(credit),size=333)
i_train = (1:nrow(credit))[-i_test]
```

Credit scoring with ensemble techniques

- model 1 : logistic regression

```
GLM = glm(Creditability~., data=credit[i_train,], family= binomial)
pGLM = predict(GLM, newdata=credit[i_test,], type="response")
classGLM = c(0,1)[1+(pGLM>.5)*1]
```

- model 2 : classification tree

```
library(rpart)
CART = rpart(Creditability~., data=credit[i_train,])
classCART = predict(CART, newdata=credit[i_test,], type="class")
```

- model 3 : SVM (with a Gaussian kernel)

```
library(kernlab)
SVM = ksvm(Creditability~., data=credit[i_train,], kernel = "rbfdot", C
    =1)
classSVM =predict(SVM, newdata=credit[i_test,])
```

Credit scoring with ensemble techniques

On the correlation of those 3 models,

```
> df = data.frame(as.numeric(classCART)-1,
+                 as.numeric(classGLM),
+                 as.numeric(classSVM)-1)
> names(df)=c("CART","GLM","SVM")
> cor(df)
          CART       GLM       SVM
CART 1.0000000 0.3777031 0.4888801
GLM  0.3777031 1.0000000 0.5528032
SVM  0.4888801 0.5528032 1.0000000
```

Consider a simple (majority based) voting procedure

```
> vote = ifelse(rowSums(df)>1.5,1,0)
```

Use AUC a classification metrics

```
> library(caret)
> auc = function(y) caret::confusionMatrix(data=as.factor(y),reference=
    credit[i_test,"Creditability"])$overall["Accuracy"]
> y = credit[i_test,"Creditability"]
```

Credit scoring with ensemble techniques

On the classification tree,

```
> table(df$CART,y)
      0   1
  0  42  46
  1  52 193
> auc(df$CART)
 Accuracy
0.7057057
```

on the GLM

```
> table(df$GLM,y)
      0   1
  0  43  43
  1  51 196
> auc(df$GLM)
 Accuracy
0.7177177
```

On the SVM

```
> table(df$SVM,y)
      0   1
  0  33  26
  1  61 213
> auc(df$SVM)
 Accuracy
0.7387387
```

on the majority votes

```
> table(vote,credit[i_test
    ,y])
vote   0   1
   0  37  29
   1  57 210
> auc(vote)
 Accuracy
0.7417417
```

Credit scoring with ensemble techniques

```
> df$CREDIT = as.numeric(credit[i_test,"Creditability"])-1
```

```
> reglm = lm(CREDIT~CART+GLM+SVM,data=df)
> df$STACKLM = predict(reglm)
> auc((df$STACKLM>.5)*1)
 Accuracy
0.7627628
```

```
> reg = glm(CREDIT~CART+GLM+SVM,data=df,family=binomial)
> df$STACKGLM = predict(reg, type="response")
> auc((df$STACKGLM>.5)*1)
 Accuracy
0.7627628
```

idea of slacking

Credit scoring with ensemble techniques

Instead of a (majority based) voting procedure, why not consider predictive probabilities?

```
> pGLM = pGLM
> pCART = predict(CART, newdata=credit[i_test,], type="prob")[,2]
> SVM = ksvm(Creditability~., data=credit[i_train,], kernel = "rbfdot",C
    =1, prob.model = TRUE)
> pSVM = predict(SVM, newdata=credit[i_test,],type="probabilities")[,2]
> pdf = data.frame(as.numeric(pCART),
+                  as.numeric(pGLM),
+                  as.numeric(pSVM))
> names(pdf)=c("CART","GLM","SVM")
> cor(pdf)
          CART      GLM       SVM
CART 1.0000000 0.516073 0.6101085
GLM  0.5160730 1.000000 0.8641040
SVM  0.6101085 0.864104 1.0000000
```

Credit scoring with ensemble techniques

Instead of a (majority based) voting procedure, why not consider predictive probabilities?

```
> pGLM = pGLM
> pCART = predict(CART, newdata=credit[i_test,], type="prob")[,2]
> SVM = ksvm(Creditability~., data=credit[i_train,], kernel = "rbfdot",C
    =1, prob.model = TRUE)
> pSVM = predict(SVM, newdata=credit[i_test,],type="probabilities")[,2]
> pdf = data.frame(as.numeric(pCART),
+                  as.numeric(pGLM),
+                  as.numeric(pSVM))
> names(pdf)=c("CART","GLM","SVM")
> cor(pdf)
          CART      GLM       SVM
CART 1.0000000 0.516073 0.6101085
GLM  0.5160730 1.000000 0.8641040
SVM  0.6101085 0.864104 1.0000000
```

Credit scoring with ensemble techniques

One can consider the average (probability) prediction

$$\widehat{y}_i = \text{average}(\widehat{y}_i^{tree}, \widehat{y}_i^{glm}, \widehat{y}_i^{svm}) = \frac{1}{3}(\widehat{y}_i^{tree} + \widehat{y}_i^{glm} + \widehat{y}_i^{svm})$$

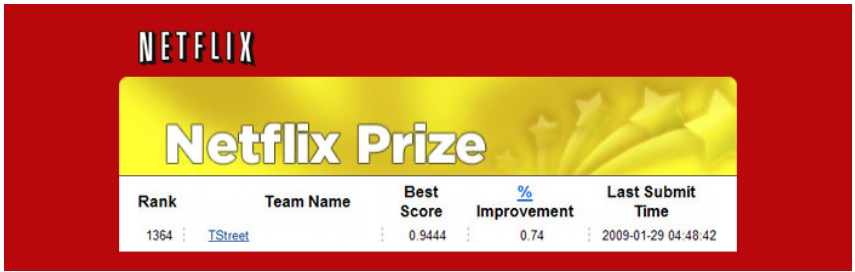
```
> auc((apply(pdf,1,mean)>.5)*1)
 Accuracy
0.7417417
```

or

$$\widehat{y}_i = \text{median}(\widehat{y}_i^{tree}, \widehat{y}_i^{glm}, \widehat{y}_i^{svm})$$

```
> auc((apply(pdf,1,median)>.5)*1)
 Accuracy
0.7477477
```

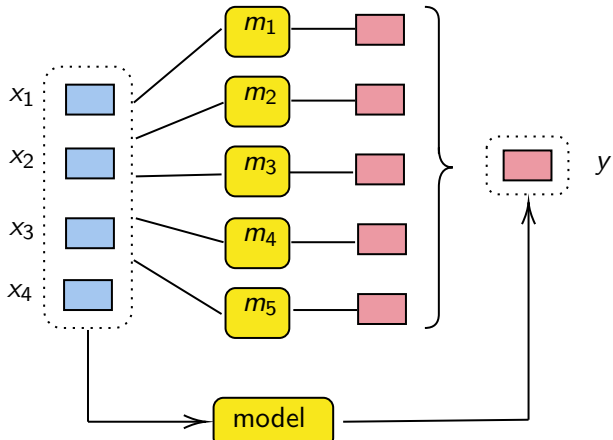
Ensemble techniques: Netflix price



2009: US$1,000,000 Netflix Prize, Winner BellKor's Pragmatic Chaos (ensemble of 107 models)

"Our experience is that most efforts should be concentrated in deriving substantially different approaches, rather than refining a simple technique [...] We strongly believe that the success of an ensemble approach depends on the ability of its various predictors to expose different complementing aspects of the data. Experience shows that this is very different than optimizing the accuracy of each individual predictor"

Bagging (Bootstrap Aggregating)



Bagging (Bootstrap Aggregating)

> Bootstrap aggregating, also called bagging (from bootstrap aggregating) or bootstrapping, is a machine learning (ML) ensemble meta-algorithm designed to improve the stability and accuracy of ML classification and regression algorithms. It also reduces variance and overfitting. $\mathbb{W}$

$$\widehat{m}_{\text{bag}}(\boldsymbol{x}_i) = \frac{1}{B}\sum_{j=1}^{B} \widehat{m}_j(\boldsymbol{x}_i), \quad \text{where models } \widehat{m}_j \text{ are trained on bootstraped sample} \tag{2}$$

$$\widehat{m}_{\text{bag}}(\boldsymbol{x}_i) = \underset{k}{\text{argmax}} \sum_{j=1}^{B} \widehat{m}_j(\boldsymbol{x}_i) \tag{3}$$

$$\mathbb{E}_D[\widehat{m}_{\text{bag}}(\boldsymbol{x})] = \frac{1}{B}\sum_{j=1}^{B} \mathbb{E}_D[\widehat{m}_j(\boldsymbol{x})] = \frac{1}{B} B \cdot \mathbb{E}_D[\widehat{m}_j(\boldsymbol{x})] = \mathbb{E}_D[\widehat{m}_j(\boldsymbol{x})]$$

Bagging (Bootstrap Aggregating)

$$\begin{aligned}\text{Var}(\widehat{m}_{\text{bag}}(\boldsymbol{x})) &= \text{Var}\left(\frac{1}{B}\sum_{b=1}^{B}\widehat{m}_b(\boldsymbol{x})\right)\\ &= \frac{1}{B^2}\text{Var}\left(\sum_{b=1}^{B}\widehat{m}_b(\boldsymbol{x})\right)\\ &= \frac{1}{B^2}\sum_{b_1=1}^{B}\sum_{b_2=1}^{B}\text{Cov}(\widehat{m}_{b_1}(\boldsymbol{x}),\widehat{m}_{b_2}(\boldsymbol{x}))\\ &\leq \frac{1}{B^2}B^2\text{Var}(\widehat{m}_b(\boldsymbol{x})) = \text{Var}(\widehat{m}_b(\boldsymbol{x}))\end{aligned}$$

because, since $b_1 \neq b_2$, we have $\text{Corr}[\widehat{m}_{b_1}(\boldsymbol{x}),\widehat{m}_{b_2}(\boldsymbol{x})] \leq 1$.

Bagging (Bootstrap Aggregating)

Actually, if $\text{Var}(\widehat{m}_b(\boldsymbol{x})) = \sigma^2(\boldsymbol{x})$ and $\text{Corr}[\widehat{m}_{b_1}(\mathbf{x}), \widehat{m}_{b_2}(\boldsymbol{x})] = r(\boldsymbol{x})$,

$$\text{Var}(\widehat{m}_{\text{bag}}(\boldsymbol{x})) = r(\boldsymbol{x})\sigma^2(\boldsymbol{x}) + \frac{1 - r(\boldsymbol{x})}{B}\sigma^2(\boldsymbol{x})$$

The greater the aggregation of different models, the lower the variability.

Tree instability makes them good candidates for aggregation.

Random Forests

Bagging benefits unstable techniques such as decision trees or neural networks. or neural networks.

In the most famous version of random forests, we push the envelope to create even more different trees.

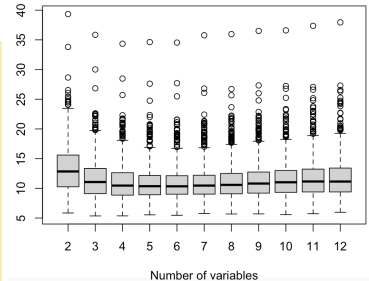
To do this, when creating the trees, we'll randomly sample a number of predictors, say m, from which we'll select the best partitioning. from which we'll select the best partitioning.
Assuming $m = \sqrt{p}$, at each stage of tree formation (each node), we consider only a subset of the predictors.

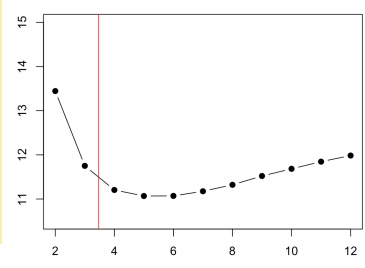
Consequently, trees are faster to form.

But more importantly, they're even more different, capturing all different aspects of the data.

Random Forests

```
> library(randomForest)
> library(ISLR2)
> set.seed(123)
> n = nrow(Boston)
> sim = function(t){
+   train = sample(n, size = n*.7)
+   subsim = function(i){
+     rf.boston = randomForest(medv ~ ., data = Boston, subset = train, mtry = i)
+     yhat.rf = predict(rf.boston, newdata = Boston[-train, ])
+     mean((yhat.rf - Boston[-train, "medv"])^2)
+   }
+   Vectorize(subsim)(2:12)
+ }
> Vectorize(sim)(1:499)
```

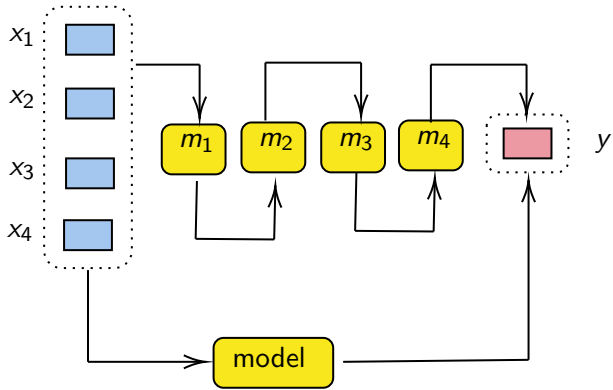




Boosting

> The concept of boosting is based on the question posed by Kearns and Valiant (Kearns (1988); Kearns and Valiant (1989)) "Can a set of weak learners create a single strong learner?" A weak learner is defined as a classifier that is only slightly correlated with the true classification. A strong learner is a classifier that is arbitrarily well-correlated with the true classification. Robert Schapire answered the question in the affirmative in a paper published in 1990. W

Boosting



Boosting

Algorithm 2: Boosting (version 1)

initialization : k (number of trees), γ, $f_0(\boldsymbol{x}) = \overline{y}$;
for $t = 1, 2, ...k$ **do**
    compute $r_{i,t} \leftarrow y_i - f_{t-1}(\boldsymbol{x}_i)$;
    fit a model $r_{i,t} \sim h(\boldsymbol{x}_i)$ for some tree h;
    update $f_t(\cdot) = f_{t-1}(\cdot) + \gamma h(\cdot)$

Algorithm 3: Boosting (version 2)

initialization : k (number of trees), $f_0(\boldsymbol{x}) = \underset{\gamma}{\text{argmin}} \sum_{i=1}^{n} \ell(y_i, \gamma)$;
for $t = 1, 2, ...k$ **do**
    compute $r_{i,t} \leftarrow \left.\dfrac{\partial \ell(y_i, \widehat{y})}{\partial \widehat{y}}\right|_{\widehat{y}=f_{t-1}(\boldsymbol{x}_i)}$;
    fit a model $r_{i,t} \sim h(\boldsymbol{x}_i)$ for some tree h;
    update $f_t(\cdot) = f_{t-1}(\cdot) + \gamma h(\cdot)$

Algorithme AdaBoost

> AdaBoost (short for Adaptive Boosting) is a statistical classification meta-algorithm formulated by Yoav Freund and Robert Schapire in 1995, who won the 2003 Gödel Prize for their work. It can be used in conjunction with many types of learning algorithm to improve performance. The output of multiple weak learners is combined into a weighted sum that represents the final output of the boosted classifier. Usually, AdaBoost is presented for binary classification, although it can be generalized to multiple classes or bounded intervals of real values. AdaBoost is adaptive in the sense that subsequent weak learners (models) are adjusted in favor of instances misclassified by previous models. In some problems, it can be less susceptible to overfitting than other learning algorithms. W

An alternative is to see the update through weights (more weight if classification is not correct)

Algorithme AdaBoost

> At each iteration of the training process, a weight $w_{i,t}$ is assigned to each sample in the training set equal to the current error on that sample. W

On s'intéresse à un problème de classification binaire (0 ou 1) à partir d'une base de données $\mathcal{D} = \{Y_i, \mathbf{X}_i\}_{i=1,\ldots,n}$.

1. Toutes les observations reçoivent un poids identique: $p_1(\mathbf{X}_i) = 1/n$, $i = 1, \ldots, n$.

Algorithme AdaBoost (suite)

2. Pour l'étape $t = 1, \ldots, T$,

   2a. Piger au hasard (en utilisant les poids $p_t(\mathbf{X}_i)$) une base de données d'entrainement $\mathcal{D}_t = (\mathcal{D}, p_t)$.

   2b. Apprendre une règle de classification r_t sur cette base de données.

   2c. Calculer l'erreur apparente

$$\epsilon_t = p_t\left(r_t(\mathbf{X}_i) \neq Y_i\right) = \sum_{i:r_t(\mathbf{X}_i)\neq Y_i} p_t(i)$$

   pour $\mathcal{D}_t$ et évaluer $\alpha_t = 0.5 \ln((1-\epsilon_t)/\epsilon_t)$.

   2d. Mettre à jour les poids:

$$p_{t+1}(\mathbf{X}_i) = \begin{cases} \frac{p_t(\mathbf{X}_i)}{Z_t} e^{-\alpha_t}, & r_t(\mathbf{X}_i) = Y_i \\ \frac{p_t(\mathbf{X}_i)}{Z_t} e^{+\alpha_t}, & r_t(\mathbf{X}_i) \neq Y_i, \end{cases}$$

   où Z_t est une constante de normalisation assurant que $\sum_{i=1}^{n} p_{t+1}(\mathbf{X}_i) = 1$.

Algorithme AdaBoost (suite)

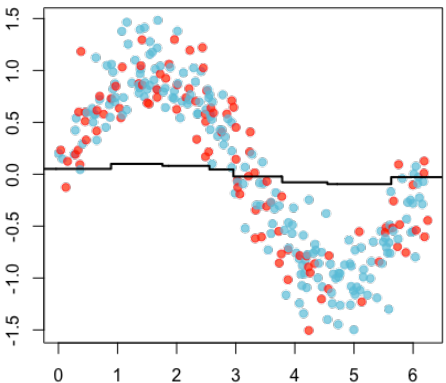
3. La classification finale faite par le modèle est alors

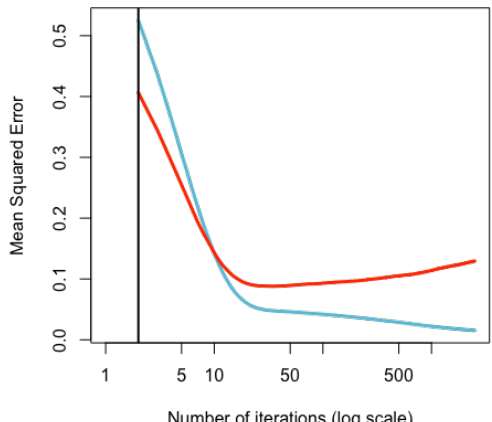
$$R(\mathbf{X}_i) = \begin{cases} 1, & \sum_{t=1}^{T} \alpha_t r_t(\mathbf{X}_i) > 0 \\ -1, & \sum_{t=1}^{T} \alpha_t r_t(\mathbf{X}_i) < 0. \end{cases}$$

Algorithm 4: Boosting (Adaboost)

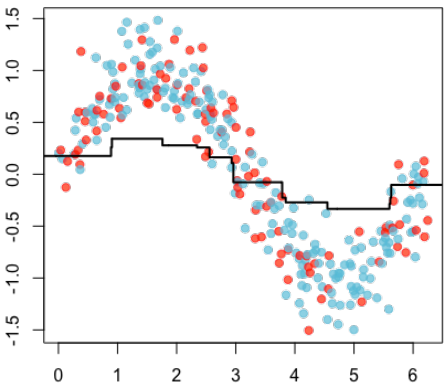
initialization : k, $\omega_i \leftarrow 1/n$;
for $t = 1, 2, ...k$ **do**
    error rate $e_t \leftarrow \dfrac{\sum_{i=1}^{n} \omega_i \mathbf{1}(y_i \neq f_{t-1}(\boldsymbol{x}_i))}{\sum_{i=1}^{n} \omega_i}$;
    set $\alpha_t \leftarrow \log(1 - e_t) - \log(e_t)$;
    update $\omega_i \leftarrow \omega_i \, e^{\alpha_t \mathbf{1}(y_i \neq f_{t-1}(\boldsymbol{x}_i))}$;
    update $f_t(\cdot) \leftarrow f_{t-1}(\cdot) + \alpha_t h(\cdot)$

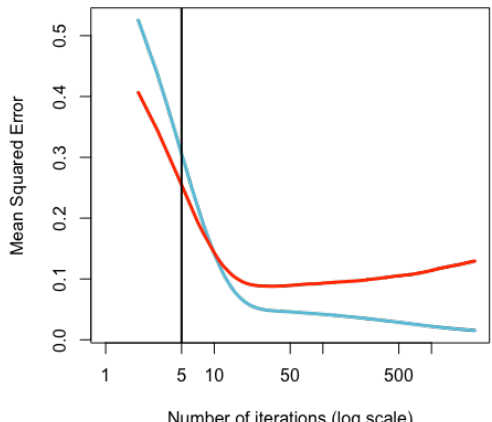
Boosting as Sequential Learning



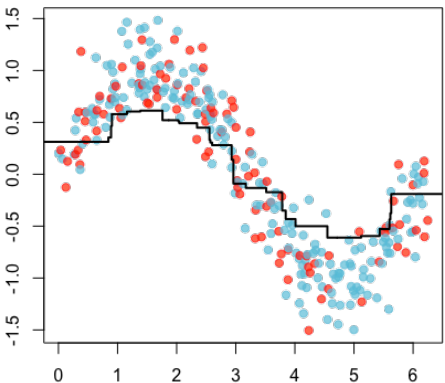


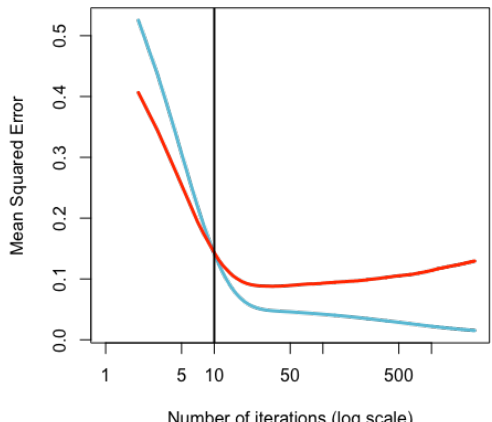
Boosting as Sequential Learning



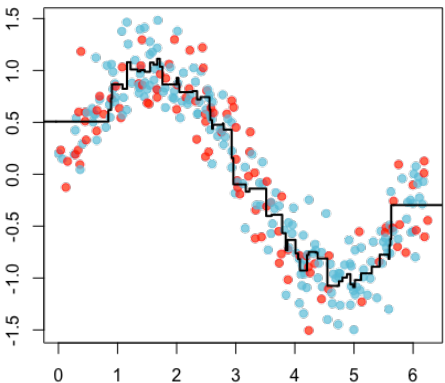


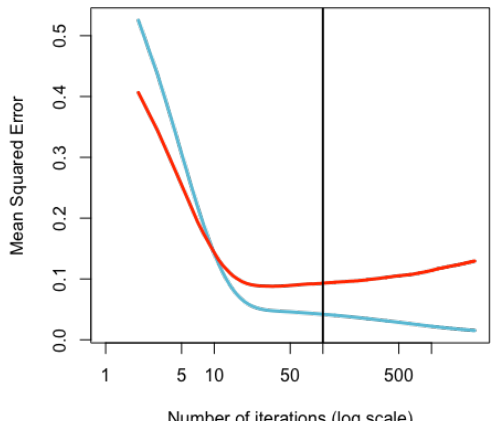
Boosting as Sequential Learning



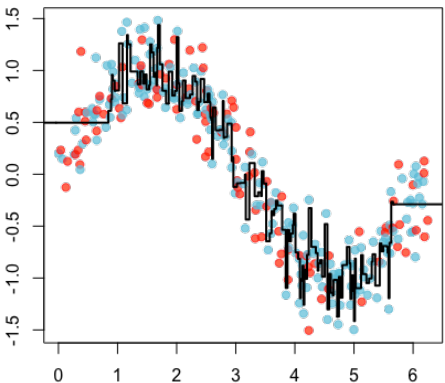


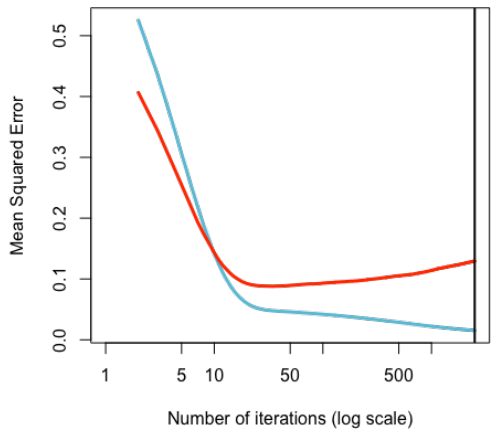
Boosting as Sequential Learning





Boosting as Sequential Learning





Model Selection

Ho do we choose an appropriate hypothesis set or learning algorithm for a given problem? For a given binary hypothesis class $\mathcal{H}$ and a function $h \in \mathcal{H}$, we have that

$$\mathcal{R}(h) - \mathcal{R}^\star = \underbrace{\mathcal{R}(h) - \inf_{g \in \mathcal{H}} \{\mathcal{R}(g)\}}_{\text{estimation}} + \inf_{g \in \mathcal{H}} \{\mathcal{R}(g)\} - \mathcal{R}^\star$$

Oracles

An **oracle** is a theoretical black-box that provides access to information not directly observable, used to guide learning algorithms.

- Typically used in computational learning theory.
- Allows querying labels, concepts, or optimal actions.
- Encapsulates external knowledge or supervision.

Definition

A **labeling oracle** $\mathcal{O}_L$ takes an input $x \in \mathcal{X}$ and returns the true label $y \in \mathcal{Y}$:

$$\mathcal{O}_L(x) = f(x)$$

where $f : \mathcal{X} \to \mathcal{Y}$ is the unknown target function.

- Used in PAC learning to model labeled data access.
- Sample complexity depends on accuracy ϵ and confidence δ.

Oracles

Definition

A **membership oracle** $\mathcal{O}_M$ allows querying whether a hypothesis h agrees with target concept c on input x:

$$\mathcal{O}_M(x) = c(x) \in \{0, 1\}$$

- Supports efficient learning via active queries.
- Key component in the Angluin's model of exact learning.

Definition

An **equivalence oracle** $\mathcal{O}_E$ takes a hypothesis h and returns:

- `YES`, if $h \equiv c$
- A counterexample x such that $h(x) \neq c(x)$, otherwise

- Used to verify hypothesis correctness.
- Forms basis of learning via hypothesis testing.
- Allows learning in $\log |C|$ steps for finite concept class C.

Model Selection

Definition

Model selection is the process of choosing the best model $\hat{h} \in \mathcal{H}$ from a hypothesis class $\mathcal{H}$ based on observed data.

- Key step in both training and generalization.
- Goal: Minimize the true risk:

$$R(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(h(x), y)]$$

- Cannot access $R(h)$ directly — we use empirical estimates.

Empirical Risk

Given data $S = \{(x_i, y_i)\}_{i=1}^n$, define:

$$\hat{R}_S(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(h(x_i), y_i)$$

Model Selection

- ERM selects:

$$\hat{h} = \arg\min_{h \in \mathcal{H}} \hat{R}_S(h)$$

- Tends to overfit if $\mathcal{H}$ is too complex.
- Bias-variance tradeoff central to model selection.

Oracle Comparison

Let $h^* = \arg\min_{h \in \mathcal{H}} R(h)$ be the **oracle model**.

A learning algorithm is **oracle-efficient** if it achieves:

$$R(\hat{h}) \leq R(h^*) + \epsilon_n$$

with high probability, where $\epsilon_n \to 0$ as $n \to \infty$.

- Bounds deviation from oracle performance.
- Central in statistical learning theory.

Model Selection

- Consider a sequence of model classes:

$$\mathcal{H}_1 \subset \mathcal{H}_2 \subset \cdots \subset \mathcal{H}_k$$

- Tradeoff between empirical risk and complexity penalty:

$$\hat{h}_j = \arg\min_{h \in \mathcal{H}_j} \hat{R}_S(h) + \lambda \cdot \text{pen}(h)$$

- Choose $\hat{h}$ from optimal $\mathcal{H}_j$:

$$\hat{h} = \arg\min_j \left\{ \hat{R}_S(\hat{h}_j) + \text{pen}(\mathcal{H}_j) \right\}$$

Examples

VC dimension, Rademacher complexity, AIC, BIC.

Model Selection

- Cross-validation estimates generalization error:

$$\mathrm{CV}_k = \frac{1}{k} \sum_{i=1}^{k} \hat{R}_{S^{(i)}}(h^{(i)})$$

- Select model minimizing CV_k estimate.
- Connects to oracle via goal: simulate access to $R(h)$.
- Summary:
  - Model selection balances fit vs. complexity.
  - Theoretical bounds use oracle comparisons.
  - Tools: ERM, SRM, CV, regularization.

Leave-one-Out Cross Validation

> In statistics, the jackknife (jackknife cross-validation) is a cross-validation technique and, therefore, a form of resampling. It is especially useful for bias and variance estimation. The jackknife pre-dates other common resampling methods such as the bootstrap. Given a sample of size n, a jackknife estimator can be built by aggregating the parameter estimates from each subsample of size $(n-1)$ obtained by omitting one observation. W

Leave-one-Out Cross Validation

For a linear regression, $\boldsymbol{y} = \boldsymbol{X}\beta + \varepsilon$,

$$\widehat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{y} \text{ where } \boldsymbol{H} = \boldsymbol{X}^\top(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}$$

since $\widehat{\beta} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}$.
If we remove observation i,

$$\widehat{\beta}_{-i} = \left((\boldsymbol{X}\boldsymbol{X}^\top)^{-1} + \frac{(\boldsymbol{X}\boldsymbol{X}^\top)^{-1}\boldsymbol{x}_i\boldsymbol{x}_i^\top(\boldsymbol{X}\boldsymbol{X}^\top)^{-1}}{1-\boldsymbol{x}_i^\top(\boldsymbol{X}\boldsymbol{X}^\top)^{-1}\boldsymbol{x}_i}\right)(\boldsymbol{X}\boldsymbol{y} - \boldsymbol{x}_i y_i)$$

and therefore $\widehat{y}_{-i} = \widehat{m}_{-1}(\boldsymbol{x}_i) = \boldsymbol{x}_i^\top\widehat{\beta}_{-i}$ becomes

$$\widehat{y}_{-i} = \left(\boldsymbol{x}_i^\top(\boldsymbol{X}\boldsymbol{X}^\top)^{-1} + \frac{\boldsymbol{x}_i^\top(\boldsymbol{X}\boldsymbol{X}^\top)^{-1}\boldsymbol{x}_i\boldsymbol{x}_i^\top(\boldsymbol{X}\boldsymbol{X}^\top)^{-1}}{1-\boldsymbol{x}_i^\top(\boldsymbol{X}\boldsymbol{X}^\top)^{-1}\boldsymbol{x}_i}\right)(\boldsymbol{X}\boldsymbol{y} - \boldsymbol{x}_i y_i)$$

$$= \frac{1}{1-\boldsymbol{x}_i^\top(\boldsymbol{X}\boldsymbol{X}^\top)^{-1}\boldsymbol{x}_i}\boldsymbol{x}_i^\top(\boldsymbol{X}\boldsymbol{X}^\top)^{-1}(\boldsymbol{X}\boldsymbol{y} - \boldsymbol{x}_i y_i)$$

Leave-one-Out Cross Validation

$$= \frac{1}{1 - \boldsymbol{x}_i^\top (\boldsymbol{X}\boldsymbol{X}^\top)^{-1}\boldsymbol{x}_i} \left(\boldsymbol{x}_i^\top (\boldsymbol{X}\boldsymbol{X}^\top)^{-1}\boldsymbol{X}\boldsymbol{y} - \boldsymbol{x}_i^\top (\boldsymbol{X}\boldsymbol{X}^\top)^{-1}\boldsymbol{x}_i y_i \right),$$

i.e., there is an explicit formula.

Cross Validation

Definition 2.22: K-fold Cross Validation

Partition the index set $\mathcal{I} = \{1, 2, \cdots, n\}$ into K roughly equally sized folds $\mathcal{I}_1, \cdots, \mathcal{I}_K$. For $k = 1, 2, \cdots, K$,

$$\widehat{\mu}_{(-k)} \in \underset{m \in \mathcal{M}}{\operatorname{argmin}} \left\{ \sum_{i \in \mathcal{I} \setminus \mathcal{I}_k} \ell(y_i, m(\boldsymbol{x}_i)) \right\}$$

and

$$\widehat{R}_{CV} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{\text{card}(\mathcal{I}_k)} \sum_{i \in \mathcal{I}_k} \ell(y_i, \widehat{\mu}_{(-k)}(\boldsymbol{x}_i))$$

bootstrap

Wrap up...

Hold-Out Cross Validation

1. Split $\{1, 2, \cdots, n\}$ in $\mathcal{D}_{ent}$ (training) and $\mathcal{D}_{test}$ (test)
2. Estimate $\widehat{m}$ on sample $(y_i, \boldsymbol{x}_i)$, $i \in \mathcal{D}_{ent}$: $\widehat{m}_{ent}$
3. Compute $\dfrac{1}{|\mathcal{D}_{test}|} \displaystyle\sum_{i \in V} \ell(y_i, \widehat{m}_{ent}(\boldsymbol{x}_i))$

Leave-one-Out Cross Validation (LOOCV)

1. Estimate n models : $\widehat{m}_{-j}$ on sample $(y_i, \boldsymbol{x}_i)$, $i \in \{1, \cdots, n\} \backslash \{j\}$
2. Compute $\dfrac{1}{n} \displaystyle\sum_{i=1}^{n} \ell(y_i, \widehat{m}_{-i}(\boldsymbol{x}_i))$

Wrap up...

K-Fold Cross Validation

1. Split $\{1, 2, \cdots, n\}$ in K groups $V_1, \cdots, V_K$
2. Estimate K models : $\widehat{m}_k$ on sample $(y_i, \boldsymbol{x}_i)$, $i \in \{1, \cdots, n\} \backslash V_k$
3. Compute $\frac{1}{K} \sum_{k=1}^{K} \frac{1}{|V_k|} \sum_{i \in V_k} \ell(y_i, \widehat{m}_k(\boldsymbol{x}_i))$

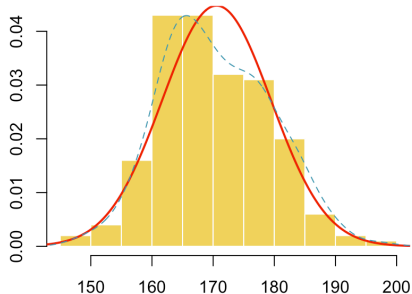
Bootstrap Cross Validation

1. Generate B boostrap samples from $\{1, 2, \cdots, n\}$, $I_1, \cdots, I_B$
2. Estimate B models : $\widehat{m}_b$ on sample $(y_i, \boldsymbol{x}_i)$, $i \in I_b$
3. Compute $\frac{1}{B} \sum_{b=1}^{B} \frac{1}{n - |I_b|} \sum_{i \notin I_b} \ell(y_i, \widehat{m}_b(\boldsymbol{x}_i))$

Unobserved Heterogeneity

Height of students 1. Gaussian model, $f(x) = \phi_{\bar{x},s^2}(x)$

```
> Davis = read.table("http://freakonometrics.
    free.fr/Davis.txt")
> X = Davis$height
> hist(X, proba=TRUE)
> (param = fitdistr(X,"normal")$estimate)
     mean        sd
170.02000  11.97788
> f1 = function(x) dnorm(x,param[1],param[2])
> x = seq(100,210,by=.2)
> lines(x,f1(x),lwd=2)
```



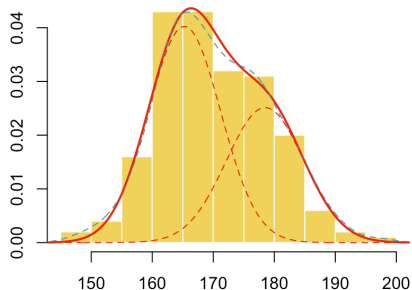

```
> logdf = function(x,p){
 p1 = p[1]
 m1 = p[2]; s1 = p[4]
 m2 = p[3]; s2 = p[5]
 log(p1*dnorm(x,m1,s1)+(1-p1)*dnorm(x,m2,s2))}
```

Parametric Statistical Models

Height of students 2. Mixture of 2 Gaussians,

$$f(x) = p\phi_{\mu_1,\sigma_1^2}(x) + (1-p)\phi_{\mu_2,\sigma_2^2}(x)$$

```
> logL = function(parameter) -sum(
    logdf(X,parameter))
> Amat = matrix(c
    (1,-1,0,0,0,0,0,0,0,0,0,1,0,0,0,
    0,0,0,0,0,0,1), 4, 5)
> bvec = c(0,-1,0,0)
> (param12 = constrOptim(c
    (.5,160,180,10,10), logL, NULL,
    ui = Amat, ci = bvec)$par)
[1]   0.5996263 165.2690084
    178.4991624   5.9447675
      6.3564746
```

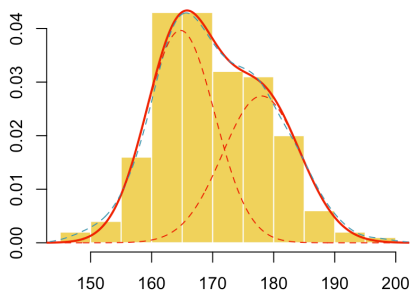


Parametric Statistical Models

Height of students 3. Conditional Gaussian

$$f(x) = \mathbb{P}(F) \cdot \phi_{\bar{x}_F, s_F^2}(x) + \mathbb{P}(M) \cdot \phi_{\bar{x}_M, s_M^2}(x)$$

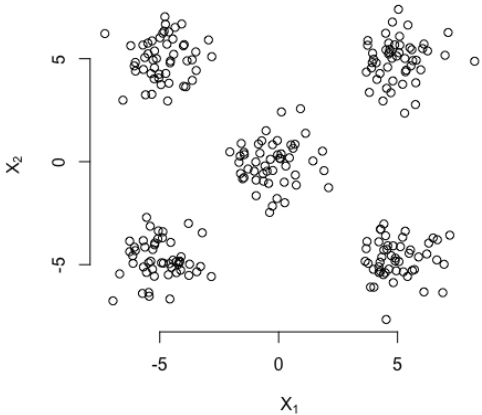
```
> (pM = mean(Davis$sex=="F"))
[1] 0.56
> (paramF = fitdistr(X[Davis$sex=="F
    "],"normal")$estimate)
      mean         sd
163.74107  11.59183
> (paramM = fitdistr(X[Davis$sex=="M
    "],"normal")$estimate)
      mean         sd
178.011364   6.404001
```

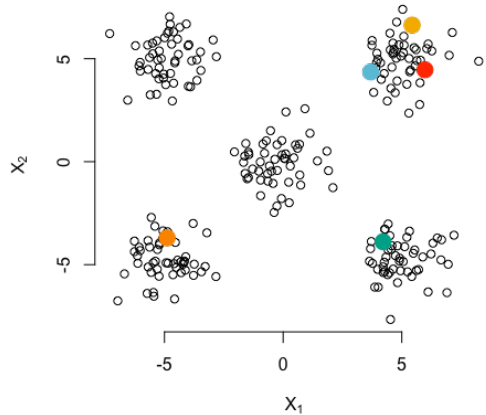


k-means

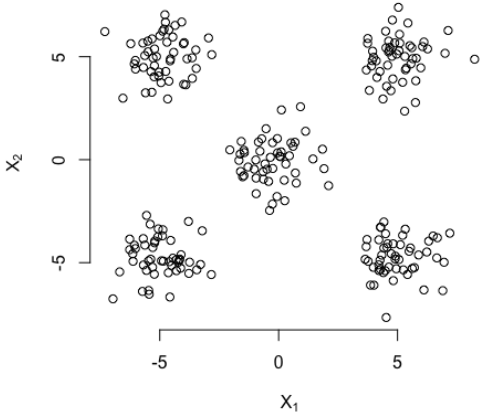
> k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. W

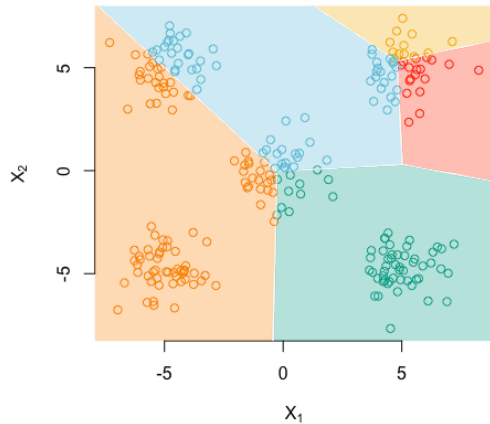
k-means



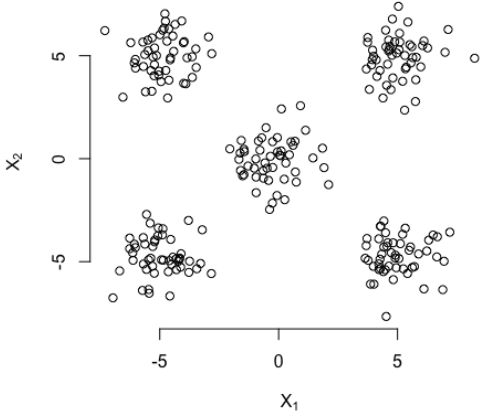


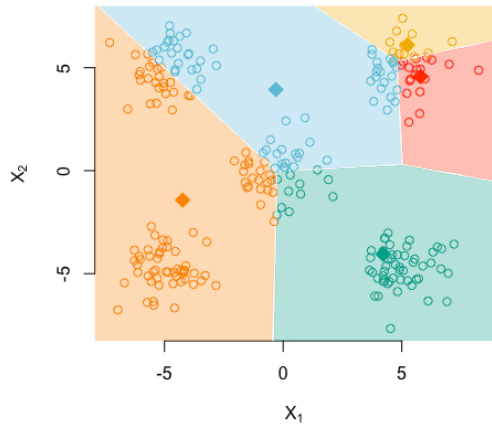
k-means



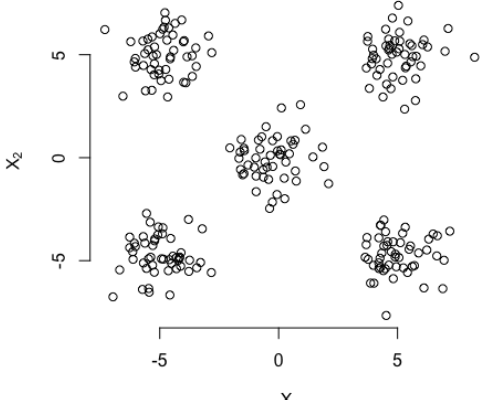


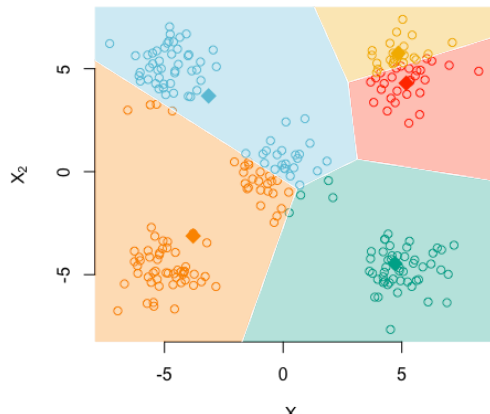
k-means



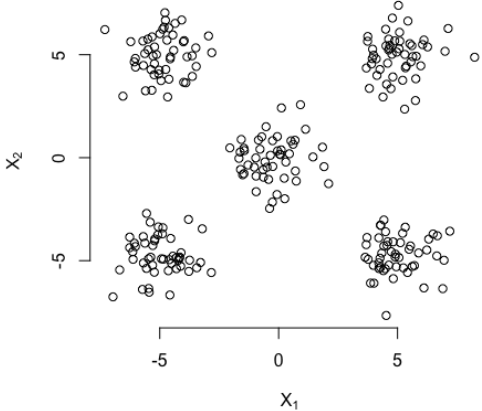


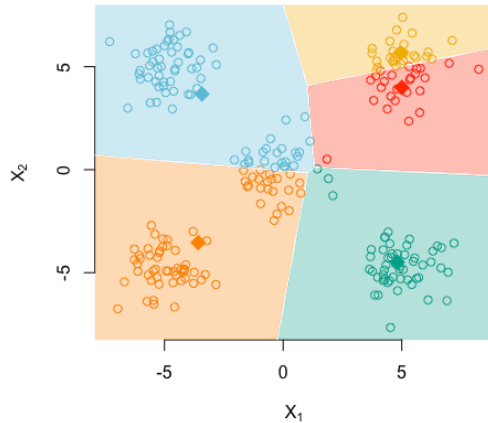
k-means



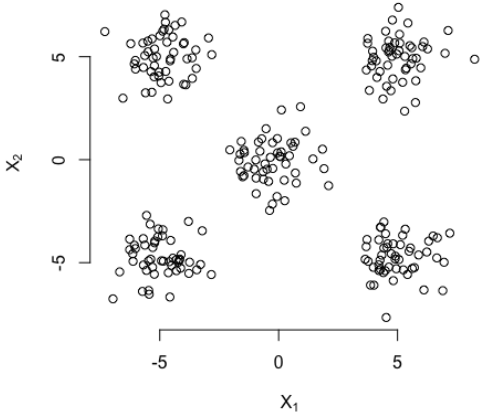


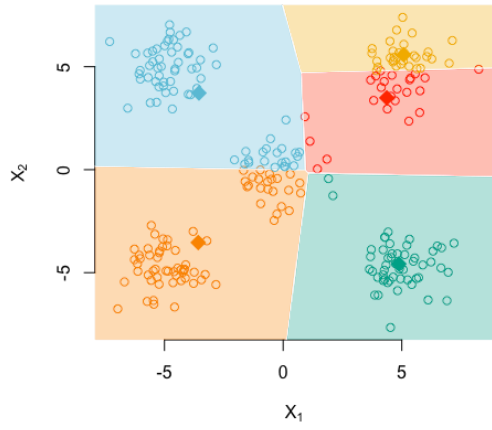
k-means



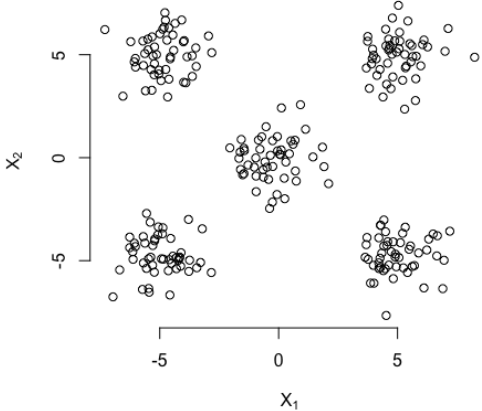


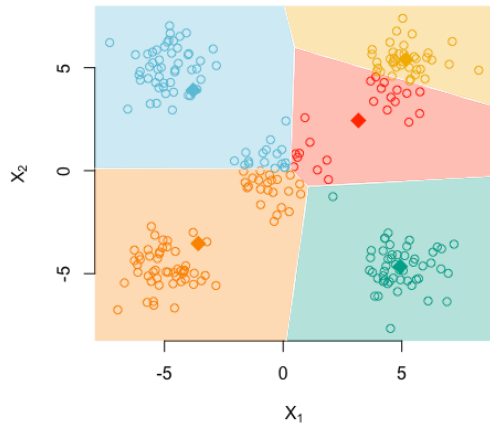
k-means



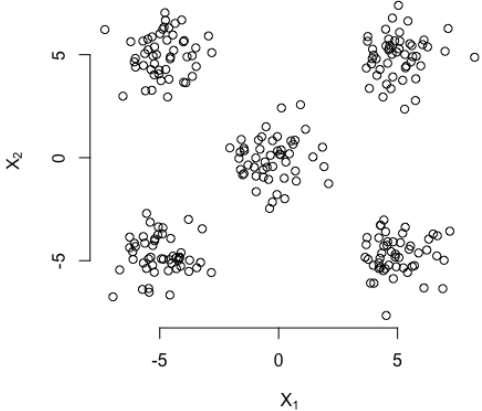


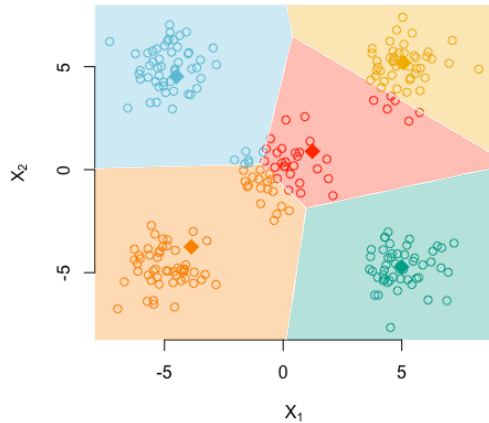
k-means



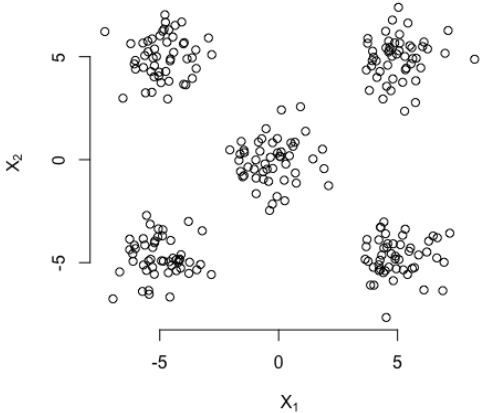


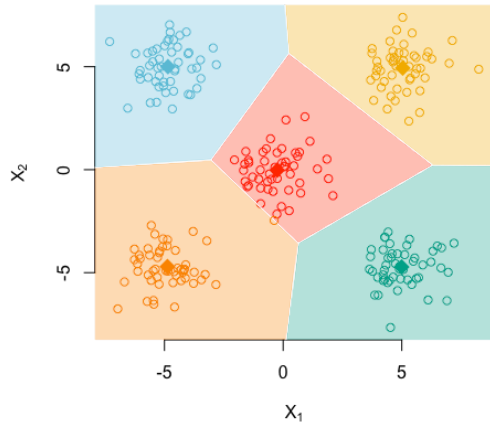
k-means



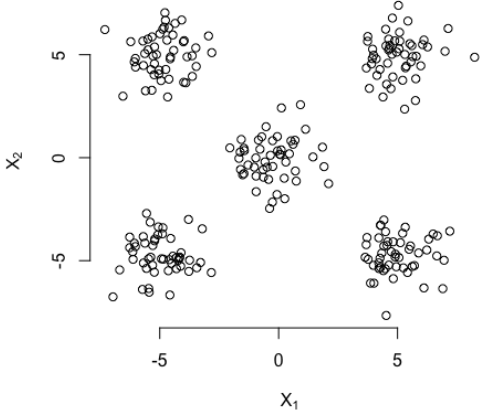


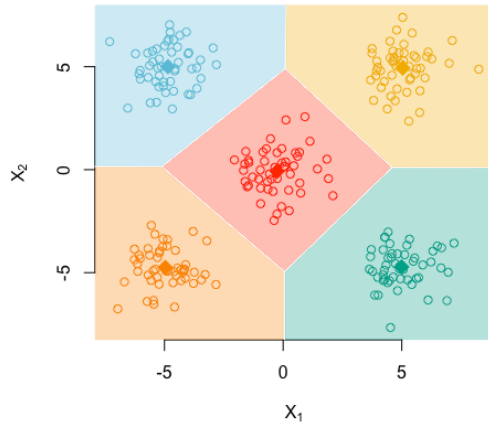
k-means





k-means





k-means

...

Given n points $\boldsymbol{x}_i$ in $\mathbb{R}^d$, we want to find $\boldsymbol{\mu}_1^\star, \cdots, \boldsymbol{\mu}_k^\star$, solutions of

$$V_{k-\text{means}} = \min \left\{ \sum_{j=1}^{k} \sum_{i \in S_j} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2 \right\}$$

that we can write

$$V_{k-\text{means}} = \min \left\{ \sum_{i=1}^{n} \min_{j=1,\cdots,k} \underbrace{\|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2}_{\text{within inertia}} \right\}$$

Finding the optimal solution to the k-means clustering problem for n observations in d dimensions can be (exactly) solved in time $O(n^{dk+1} \log n)$... which is very long.

Given an initial set of k-means, $\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_k$, alternate between the two following steps (also called Lloyd's algorithm, from ?

k-means

Assignment step: we partition observations according to the Voronoi diagram generated by $\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_k$'s,

$$S_i^{(t)} = \{\boldsymbol{x}_j : \|\boldsymbol{x}_j - \boldsymbol{\mu}_i^{(t)}\|^2 \leq \|\boldsymbol{x}_j - \boldsymbol{\mu}_{i'}^{(t)}\|^2 \ \forall i' \in \{1, \cdots, k\}\},$$

where each $\boldsymbol{x}_j$ is assigned to exactly one class $S_i^{(t)}$, at step t.

Update step: we compute the new centroids of the clusters,

$$\boldsymbol{\mu}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\boldsymbol{x}_j \in S_i^{(t)}} \boldsymbol{x}_j$$

k-means

Key why this might works :

$$\frac{1}{\#I_j}\sum_{i,i'\in I}\sum_{j=1}^{k}[x_{il}-x_{i'l}]^2 = 2\sum_{i\in I}\sum_{j=1}^{k}[x_{il}-\bar{x}_{jl}]^2 \text{ where } \bar{x}_{jl}=\frac{1}{\#I_j}\sum_{i\in I_j}x_{i,l}$$

The value of

$$\min_{I_1,\cdots,I_K}\left\{\sum_{j=1}^{K}\frac{1}{\#I_j}\sum_{i,i'\in I_j}\sum_{j=1}^{k}[x_{il}-x_{i'l}]^2\right\}$$

will necessarily decrease (but no guarantee to reach the minimum, since the function is not convex).

k-means

The center of gravity of a point cloud is defined by

$$g = \sum_{i=1}^{n} p_i \mathbf{x}_i \ = \ \bar{x}.$$

The total inertia of a point cloud is given by

$$I_T = \sum_{i=1}^{n} p_i d^2(\mathbf{x}_i, g),$$

where $d()$ is the Euclidean distance.

Inertie

In a classification procedure, the point cloud will be partitioned into K groups $C_1, \ldots, C_K$ whose respective weights are given by P_j such that

$$P_j = \sum_{i:\mathbf{x}_i \in C_j} p_i.$$

The center of gravity of each group is given by

$$g_j = \frac{1}{P_j} \sum_{i:\mathbf{x}_i \in C_j} p_i \mathbf{x}_i.$$

Inertia

In a classification procedure, the point cloud will be partitioned into K groups $C_1, \ldots, C_K$ whose respective weights are given by P_j such that

$$P_j = \sum_{i:\mathbf{x}_i \in C_j} p_i.$$

The center of gravity of each group is given by

$$g_j = \frac{1}{P_j} \sum_{i:\mathbf{x}_i \in C_j} p_i \mathbf{x}_i.$$

Hierarchical Clustering

The similarity between two groups can be determined in several ways.

Simple linkage (simple linkage): the proximity between two groups (A and B) is given by

$$\delta(A, B) = \min_{a_i \in A, b_j \in B} (d(a_i, b_j)),$$

i.e. we use the minimum of the distance between each of the point of group A and each point of group B.

Ascending Hierarchical Algorithms

Complete Linkage: the proximity between two groups (A and B) is given by

$$\delta(A, B) = \max_{a_i \in A, b_j \in B} (d(a_i, b_j)),$$

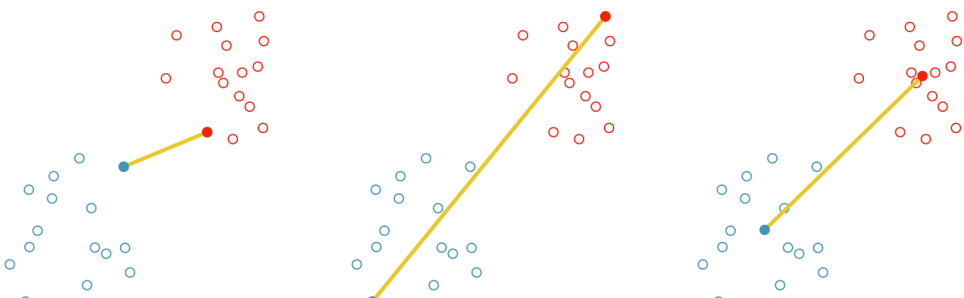
i.e. we use the maximum of the distance between each of the point of group A and each point of group B.

Ascending hierarchical algorithms

Average Linkage: the proximity between two groups groups (A and B) is given by

$$\delta(A, B) = \frac{1}{\text{card}(A)\text{card}(B)} \sum_{a_i \in A} \sum_{b_j \in B} (d(a_i, b_j)).$$

This time, we use the average of the distances between each of the points in group A and each of the points in group B.



Ascending Hierarchical Algorithms

Ward algorithm: the inertia of the point cloud is decomposed and the loss of information (loss of inertia between groups I_B) is minimized at each step. minimizes the loss of information (loss of inertia between groups I_B) at each stage: we therefore optimize the same criterion as for a dynamic reallocation classification algorithm.

At each stage, the loss of information due to the grouping of the groups A and B is given by

$$\delta(A,B) = \frac{P_A P_B}{P_A + P_B} d^2(g_A, g_B).$$

We also define the loss of information due to grouping groups $C = A \cup B$ and D by

$$\delta(C,D) = \frac{(P_A + P_D)\delta(A,D) + (P_B + P_D)\delta(B,D) - P_D\delta(A,B)}{P_A + P_B + P_D}.$$

Kaufman and Rousseeuw (2009) REPRENDRE

- Hierarchical clustering is a method of cluster analysis that builds a hierarchy of clusters.
- Two main approaches:

Spectral decomposition "⭐⭐⭐"

- If $\mathbf{M}$ is a $p \times p$ matrix, then $\vec{\boldsymbol{v}}$ is an eigenvector of $\mathbf{M}$ (associated with eigenvalue λ) if $\mathbf{M}\vec{\boldsymbol{v}} = \lambda\vec{\boldsymbol{v}}$.
- Every real symmetric $p \times p$ matrix $\mathbf{M}$ is diagonalizable in an orthonormal basis.
- That is, $\mathbf{M} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$ where $\boldsymbol{\Lambda} = \text{diag}(\boldsymbol{\lambda})$, and $\mathbf{V} = [\vec{\boldsymbol{v}}_1, \cdots, \vec{\boldsymbol{v}}_n]$ with $\mathbf{V}^{-1} = \mathbf{V}^\top$.
- If $\mathbf{M}$ is invertible ($\lambda_j \neq 0$), then $\mathbf{M}^{-1} = \mathbf{V}\boldsymbol{\Lambda}^{-1}\mathbf{V}^\top$ where $\boldsymbol{\Lambda} = \text{diag}(\boldsymbol{\lambda})$, and $\mathbf{V} = [\vec{\boldsymbol{v}}_1, \cdots, \vec{\boldsymbol{v}}_n]$.

Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction technique.

It identifies new axes (principal components) that maximize the variance in the data.

Given a data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ (centered), PCA finds orthonormal vectors $\vec{v}_1, \ldots, \vec{v}_k$ such that:

$$\vec{v}_1 = \arg \max_{\|\vec{v}\|=1} \text{Var}(\mathbf{X}\vec{v})$$

The sample covariance matrix is:

$$\mathbf{S} = \frac{1}{n-1}\mathbf{X}^\top \mathbf{X}$$

PCA seeks the eigenvectors of $\mathbf{S}$.

Principal Component Analysis

Eigen decomposition:

$$\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$$

where $\mathbf{V}$ contains the principal components (eigenvectors).

Let $\mathbf{V}_k$ be the matrix with the first k eigenvectors.

The projection of $\mathbf{X}$ onto the k-dimensional subspace:

$$\mathbf{Z} = \mathbf{X}\mathbf{V}_k$$

$\mathbf{Z}$ is the low-dimensional representation of the data.

The eigenvalues $\lambda_1, \ldots, \lambda_p$ represent the variance explained by each principal component.

The proportion of variance explained by the i-th component:

$$\frac{\lambda_i}{\sum_{j=1}^{p} \lambda_j}$$

Principal Component Analysis

Often use a scree plot to determine how many components to keep.

PCA rotates the coordinate axes to align with directions of maximum variance.

The first principal component defines the direction of highest variance.

Each subsequent component is orthogonal to the previous ones.

PCA is often viewed from two perspectives:

Individuals: the rows of the data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$

Variables: the columns of $\mathbf{X}$

Let $\mathbf{X}$ be centered (and possibly scaled), then:

$$\mathbf{S} = \frac{1}{n-1}\mathbf{X}^\top \mathbf{X} \quad \text{(covariance of variables)}$$

$$\mathbf{G} = \frac{1}{p}\mathbf{X}\mathbf{X}^\top \quad \text{(Gram matrix of individuals)}$$

These matrices are linked through the **duality** of eigen decompositions.

Principal Component Analysis

Eigen decomposition of the covariance matrix:

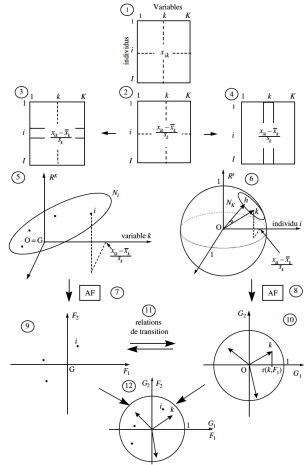
$$\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top, \quad \mathbf{V}^\top\mathbf{V} = \mathbf{I}$$

Principal component scores (individuals in PC space):

$$\mathbf{Z} = \mathbf{X}\mathbf{V}$$

Dual decomposition (of the Gram matrix):

$$\mathbf{G} = \mathbf{Z}\mathbf{Z}^\top = \mathbf{X}\mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top\mathbf{X}^\top$$



Individuals and variables are linked via the same eigenvalues.

A **biplot** represents both individuals and variables in the same low-dimensional space.

Principal Component Analysis

Let $\mathbf{X}$ be standardized. The loading vectors (columns of $\mathbf{V}$) can be plotted together with:

$$\text{Scores (individuals)} : \mathbf{Z} = \mathbf{XV}$$

$$\text{Loadings (variables)} : \mathbf{V}\sqrt{\mathbf{\Lambda}}$$

The angles between variable vectors approximate correlations.

The projection of an individual onto a variable axis estimates its value on that variable.

Partial Least Squares (PLS) regression finds components that explain variance in both $\mathbf{X}$ and $\mathbf{Y}$.

It is useful when predictors are highly collinear or $p > n$.

PLS finds latent vectors $\mathbf{t}_i = \mathbf{X}\vec{w}_i$ such that:

$$\text{Cov}^2(\mathbf{X}\vec{w}_i, \mathbf{Y}\vec{c}_i) \quad \text{is maximized}$$

Principal Component Analysis

Iteratively extracts components from $\mathbf{X}$ and $\mathbf{Y}$.

PCA: maximizes variance in $\mathbf{X}$ only.

PLS: maximizes covariance between $\mathbf{X}$ and $\mathbf{Y}$.

PLS components are better suited for prediction tasks.

Autoencoders

Definition 2.23: Dissimilarity

$L : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}_+$ such that $L(\boldsymbol{x}, \boldsymbol{x}') = 0$ if and only if $\boldsymbol{x} = \boldsymbol{x}'$.

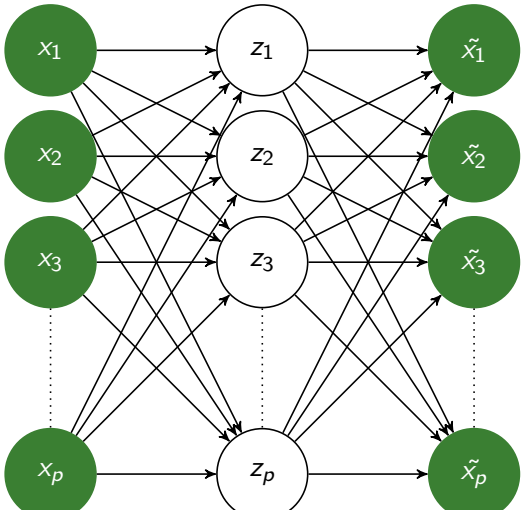
Definition 2.24: Auto-encoder

Let $p < q$. An auto-encoder is a pair (Φ, Ψ) of mappings, called **encoder** and **decoder** respectively,

$$\Phi : \mathbb{R}^q \to \mathbb{R}^p \text{ and } \Psi : \mathbb{R}^p \to \mathbb{R}^q$$

such that their composition $\Psi \circ \Phi$ has a small reconstruction error w.r.t. the chosen dissimilarity function L, i.e. for all $\boldsymbol{x}$, $L(\boldsymbol{x}, \Psi \circ \Phi(\boldsymbol{x}))$ is small.

Autoencoders



or

$$\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top$$

$$(\boldsymbol{V}\boldsymbol{X})^\top(\boldsymbol{X}\boldsymbol{V}) = \boldsymbol{V}^\top\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{\Lambda}$$

Spectral decomposition

- Si $\mathbf{M}$ is a $p \times p$ matrix, $\vec{\boldsymbol{v}}$ is an eigenvector $\mathbf{M}$ (associated to λ) if $\mathbf{M}\vec{\boldsymbol{v}} = \lambda\vec{\boldsymbol{v}}$
- If $\mathbf{M}$ can be diagonalized, $\mathbf{M} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$ where $\boldsymbol{\Lambda} = \text{diag}(\boldsymbol{\lambda})$, and $\mathbf{V} = [\vec{\boldsymbol{v}}_1, \cdots, \vec{\boldsymbol{v}}_n]$.
- If $\mathbf{M}$ is invertible, $\mathbf{M}^{-1} = \mathbf{V}\boldsymbol{\Lambda}^{-1}\mathbf{V}^\top$ where $\boldsymbol{\Lambda} = \text{diag}(\boldsymbol{\lambda})$, et $\mathbf{V} = [\vec{\boldsymbol{v}}_1, \cdots, \vec{\boldsymbol{v}}_n]$.
- **Eckart–Young–Mirsky** theorem (Eckart and Young (1936), Mirsky (1960)):

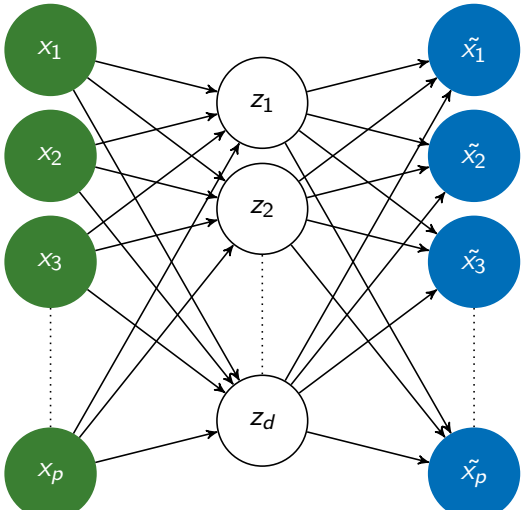
$$\mathbf{M}^\star \in \underset{\mathbf{H}:p\times p}{\text{argmin}} \left\{ \|\mathbf{M} - \mathbf{H}\|_F \text{ s.t. } \text{rank}(\mathbf{H}) \leq d \right\}$$

Si $\mathbf{M} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top = \begin{bmatrix} \mathbf{V}_d & \mathbf{V}_{p-d} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Lambda}_r & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Lambda}_{n-r} \end{bmatrix} \begin{bmatrix} \mathbf{V}_r & \mathbf{V}_{p-d} \end{bmatrix}^\top$, $\mathbf{M}^\star = \mathbf{V}_d\boldsymbol{\Lambda}_d\mathbf{V}_d^\top$

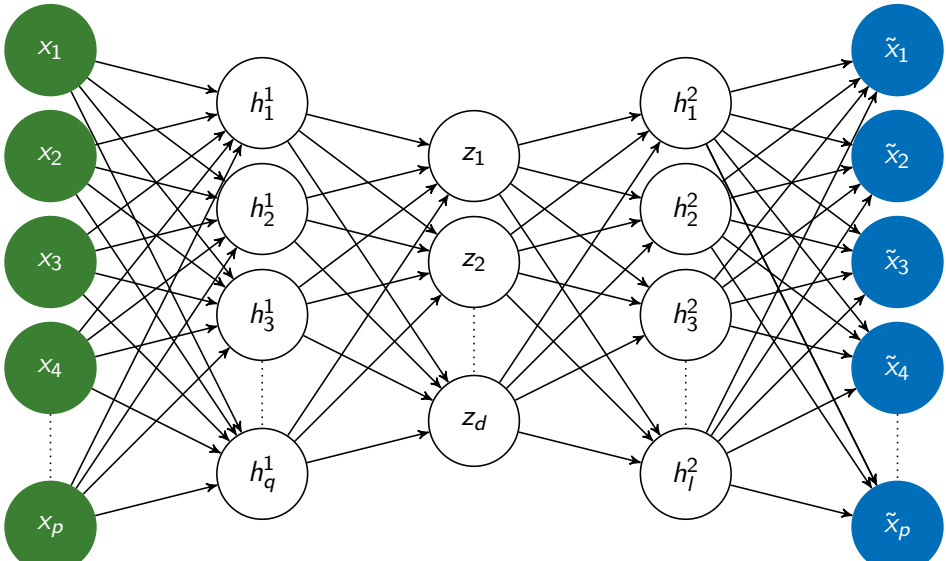
(that will be unique if $\lambda_d > \lambda_{d+1}$)

- The approximation is $\|\mathbf{M} - \mathbf{M}^\star\|_F = \sqrt{\lambda_{d+1}^2 + \cdots + \lambda_p^2}$

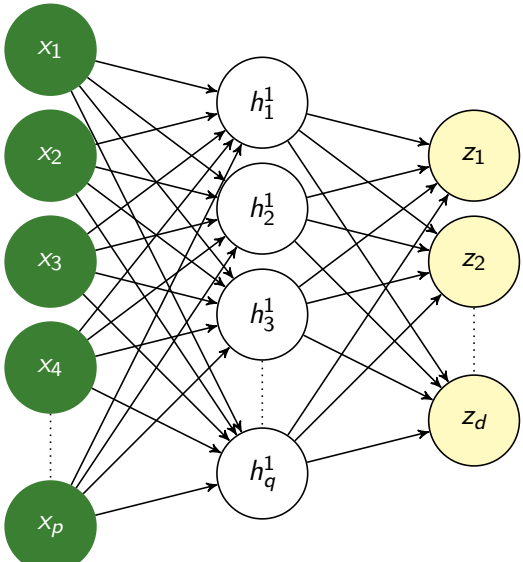
Principal Component Analysis



Autoencoders



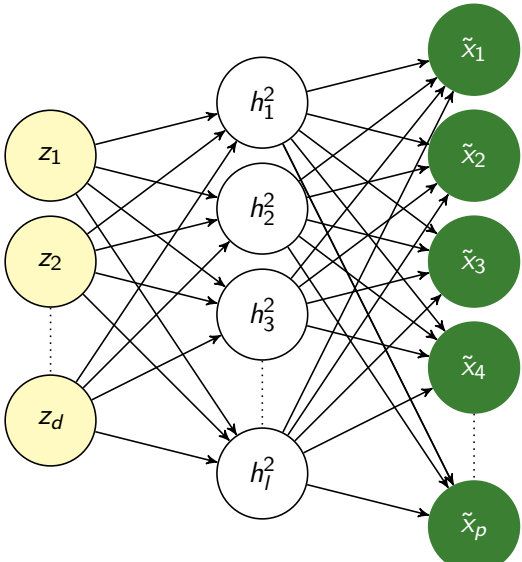
Encoder



Let $f : \mathcal{X} \to \mathcal{X}$ be an encoding function then, for example:

$$\mathbf{z} = f(\mathbf{x}) = \phi_2(\phi_1(\mathbf{x}_{1\times p}\mathbf{B}^{(1)}_{p\times q})\mathbf{B}^{(2)}_{q\times d})$$

Decoder



Let $g : \mathcal{Z} \to \mathcal{X}$ be a neural network-type then, for example:

$$\tilde{x} = g(\mathbf{z}) = \phi_4(\phi_3(\mathbf{z}_{1\times d}\mathbf{B}^{(3)}_{d\times l})\mathbf{B}^{(4)}_{l\times p})$$

Autoencoders

The error function of a linear autoencoder (PCA) is

$$\|\boldsymbol{X}-\widetilde{\boldsymbol{X}}\|^2=\|\boldsymbol{X}-\boldsymbol{P}^\top\boldsymbol{P}\boldsymbol{X}\|^2=\sum_{i=1}^{n}\left(\boldsymbol{P}^\top\boldsymbol{P}\boldsymbol{x}_i-\boldsymbol{x}_i\right)^\top\left(\boldsymbol{P}^\top\boldsymbol{P}\boldsymbol{x}_i-\boldsymbol{x}_i\right)$$

The error function of a nonlinear autoencoder is

$$\|\boldsymbol{X}-\widetilde{\boldsymbol{X}}\|^2=\|\boldsymbol{X}-g\circ f(\boldsymbol{X})\|^2=\sum_{i=1}^{n}\left(g\circ f(\boldsymbol{x}_i)-\boldsymbol{x}_i\right)^\top\left(g\circ f(\boldsymbol{x}_i)-\boldsymbol{x}_i\right)$$

Autoencoders

Encoder Φ_p / Decoder Ψ_p

$$\Psi_p \circ \Phi_p : \boldsymbol{X} \mapsto \sum_{j=1}^{p} \langle \boldsymbol{X}, \vec{\boldsymbol{v}}_j \rangle \vec{\boldsymbol{v}}_j$$

with reconstruction error of

$$\boldsymbol{X} - \Psi_p \circ \Phi_p(\boldsymbol{X}) = \sum_{j=p+1}^{q} \langle \boldsymbol{x}, \vec{\boldsymbol{v}}_j \rangle \vec{\boldsymbol{v}}_j$$

Generative Modeling

Definition

A **generative model** learns the joint probability distribution $p(x, y)$ of the data and labels, and can generate new samples from it:

$$p(x) = \int p(x|z)p(z)dz$$

where $p(x|z)$ is the likelihood, and $p(z)$ is the prior distribution of hidden variables.

The goal is to learn how data is generated.

Examples include Gaussian Mixture Models (GMMs) and Variational Autoencoders (VAEs).

Can be used for tasks such as image generation, text synthesis, and anomaly detection.

Generative Modeling

Definition

A **Gaussian Mixture Model** (GMM) is a specific case of a mixture model where each component is a Gaussian distribution:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \sigma_k^2)$$

where:

$\mathcal{N}(x|\mu_k, \sigma_k^2)$ is a Gaussian distribution with mean μ_k and variance σ_k^2.

π_k are the mixture coefficients, satisfying $\sum_k \pi_k = 1$.

GMMs are widely used in clustering and density estimation tasks.

The Expectation-Maximization (EM) algorithm is typically used for training GMMs.

Generative Modeling

Generative Process

Generative AI methods aim to model the data distribution $p(x)$ and generate new samples x'. For example, in a GMM:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \sigma_k^2)$$

The goal is to sample new data $x' \sim p(x)$.

Generative models are foundational for advanced AI techniques, such as GANs and VAEs.

These models aim to capture data distributions in a way that allows realistic data generation.

Example: Generative models can create images, music, or even text based on learned patterns.

Generative Modeling

Generative Adversarial Networks (GANs): Two neural networks (generator and discriminator) compete to generate realistic data.

Variational Autoencoders (VAEs): Use probabilistic graphical models to learn latent variable representations.

Both GANs and VAEs use latent variables to model complex data distributions and generate new samples.

Generative Process in VAEs

A VAE learns a probabilistic mapping:

$$p(x) = \int p(x|z)p(z)dz$$

where $p(x|z)$ is the likelihood, and $p(z)$ is the prior on the latent variable z.

These models are used for unsupervised learning, data generation, and anomaly detection.

Variational Autoencoders

Kingma and Welling (2013)

- Variational Autoencoders (VAEs) are generative models that learn latent representations of data.
- Unlike standard autoencoders, VAEs impose a probabilistic structure on the latent space.
- VAEs use variational inference to approximate intractable posterior distributions.

Gaussian Latent Variable Model

- Encoder (inference/recognition model):

$$q(\boldsymbol{z} \mid \boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}_x(\boldsymbol{x}), \boldsymbol{\Sigma}_x(\boldsymbol{x}))$$

- Decoder (generative model):

$$p(\boldsymbol{x} \mid \boldsymbol{z}) \sim \mathcal{N}(\boldsymbol{\mu}_z(\boldsymbol{z}), \boldsymbol{\Sigma}_z(\boldsymbol{z}))$$

Variational Autoencoders

- the decoder is the generative component: once trained, it can take random samples $\boldsymbol{z} \sim \pi$ from the latent space and produce synthetic data points by generating new data $\boldsymbol{x}' \sim p(\cdot \mid \boldsymbol{z})$ that resembles the original dataset.

Latent Variable Model

- Assume observed data $\boldsymbol{x} \in \mathbb{R}^d$ is generated from a latent variable $\boldsymbol{z} \in \mathbb{R}^k$, i.e., there exists an (unknown) deterministic mapping $\boldsymbol{x} \mapsto \boldsymbol{z}$
- The generative process:

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{z})\pi(\boldsymbol{z})\mathrm{d}\boldsymbol{z} \tag{10}$$

- The posterior distribution $p(\boldsymbol{z}|\boldsymbol{x})$ is typically intractable.

Variational Inference

- Instead of directly computing $p(\boldsymbol{z}|\boldsymbol{x})$, approximate it using $q(\boldsymbol{z}|\boldsymbol{x})$.

Variational Autoencoders

- Use Kullback-Leibler (KL) divergence to measure approximation quality:

$$D_{KL}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}|\boldsymbol{x})) \tag{11}$$

Evidence Lower Bound (ELBO)

- The marginal log-likelihood can be rewritten as:

$$\log p(x) = \mathbb{E}_{q(z|x)}\left[\log \frac{p(x|z)p(z)}{q(z|x)}\right] + D_{KL}(q(z|x)||p(z|x)) \tag{12}$$

- The ELBO is defined as:

$$\mathcal{L}(x) = \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{KL}(q(z|x)||p(z)) \tag{13}$$

- Maximizing ELBO minimizes $D_{KL}(q(z|x)||p(z|x))$.

Reparameterization Trick

Variational Autoencoders

- Directly sampling from $q(z|x)$ is not differentiable.
- Introduce a reparameterization:

$$z = \mu(x) + \sigma(x) \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \tag{14}$$

- This allows gradients to flow through μ and σ during optimization.

VAE Architecture

- Encoder: $q(z|x) = \mathcal{N}(z; \mu(x), \Sigma(x))$
- Decoder: $p(x|z) \sim \mathcal{N}(x; f(z), \sigma^2 I)$
- Loss function:

$$\mathcal{L} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{KL}(q(z|x)||p(z)) \tag{15}$$

KL Divergence Computation

- Assume $q(z|x) = \mathcal{N}(z; \mu, \sigma^2 I)$ and $p(z) = \mathcal{N}(0, I)$.

Variational Autoencoders

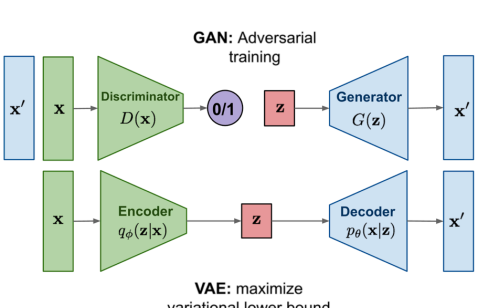
- Then,

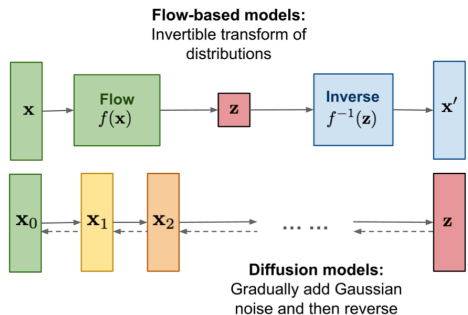
$$D_{KL}(q(z|x)||p(z)) = \frac{1}{2}\sum_{i=1}^{k}\left(\sigma_i^2 + \mu_i^2 - 1 - \log\sigma_i^2\right) \tag{16}$$

Sampling and Generation

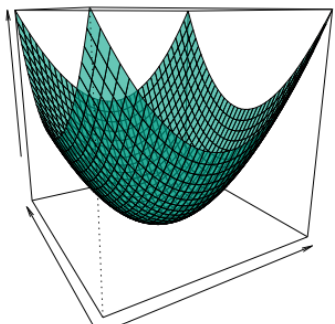
- After training, generate new samples by drawing $z \sim \mathcal{N}(0, I)$ and decoding $x \sim p(x|z)$.
- Ensures meaningful interpolation in latent space.

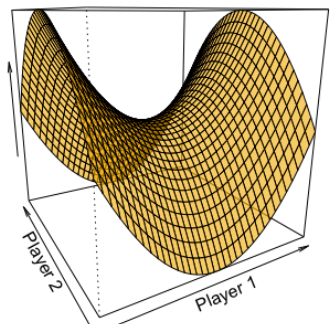
Overview


GAN: Adversarial training
x′
x
Discriminator
$D(\mathbf{x})$
0/1
z
Generator
$G(\mathbf{z})$
x′
x
Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
z
Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
x′
VAE: maximize variational lower bound


Flow-based models:
Invertible transform of distributions
x
Flow
$f(\mathbf{x})$
z
Inverse
$f^{-1}(\mathbf{z})$
x′
$\mathbf{x}_0$
$\mathbf{x}_1$
$\mathbf{x}_2$
... ...
z
Diffusion models:
Gradually add Gaussian noise and then reverse

Adversarial learning





Adversarial learning

Definition

Adversarial learning focuses on the study of models' vulnerabilities when exposed to perturbations in the input data.

> Adversarial examples are intentionally crafted inputs that cause machine learning models to make incorrect predictions.
>
> Adversarial learning explores how to defend against these attacks or how to use adversarial examples for training.

Adversarial learning

Definition 2.25: Adversarial example

An **adversarial example** $\boldsymbol{x}'$ is an input to a model that is close to an original input $\boldsymbol{x}$, but causes the model to different predictions, i.e., given a model $m : \mathcal{X} \to \mathcal{Y}$, a distance metric d on $\mathcal{Y}$ and a norm $\|\cdot\|$ on $\mathcal{X}$, and thresholds $\varepsilon, \alpha > 0$, an input $\boldsymbol{x}' \in \mathcal{X}$ is called an **adversarial example** for an original input $\in \mathcal{X}$ if:

$$\begin{cases} \|\boldsymbol{x} - \boldsymbol{x}'\| < \varepsilon & \text{small input perturbation} \\ d(m(\boldsymbol{x}), m(\boldsymbol{x}')) > \alpha & \text{large model output change} \end{cases}$$

Adversarial examples are typically generated by small perturbations.

These perturbations are often imperceptible to humans but lead to model errors.

Example: Slightly altering pixels in an image to mislead an image classifier.

Adversarial learning

Types of Attacks

Adversarial attacks are methods used to generate adversarial examples. Common attacks include:

Fast Gradient Sign Method (FGSM): Perturbation is calculated by:

$$\boldsymbol{x}' = \boldsymbol{x} + \varepsilon \cdot \text{sign}(\nabla_{\boldsymbol{x}} \ell(m_\theta(\boldsymbol{x}), y))$$

where $\ell(m_\theta(\boldsymbol{x}), y))$ is the loss function, and ε is the perturbation magnitude.

Projected Gradient Descent (PGD): Iterative version of FGSM, where the adversarial example is refined over multiple steps.

These attacks are designed to optimize the perturbation such that the model misbehaves.

The goal is to maximize the loss function with respect to the input x.

Adversarial learning

Definition

Adversarial training is a technique to defend against adversarial attacks by including adversarial examples in the training process.

The model is trained on both original and adversarial examples.

This forces the model to learn to be robust to perturbations:

$$\mathcal{L}_{adv}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{x'\in\mathcal{A}} \ell(m_\theta(\boldsymbol{x}'), y))\right]$$

where $\mathcal{A}$ represents the adversarial set.

Example: Adding adversarial examples generated by FGSM to the training data.

The model learns to be less sensitive to small perturbations.

Adversarial Training

The adversarial training process includes adversarial examples in the training data to improve the model's robustness.

Adversarial learning

During adversarial training, adversarial examples are generated and used to update the model.

The model learns to be more robust to these perturbations.

Adversarial learning addresses the vulnerability of machine learning models to small, carefully crafted perturbations.

Adversarial examples can be used to improve model robustness through adversarial training.

Applications:

Security: Protecting AI systems from adversarial attacks (e.g., image classification, speech recognition).
Robustness testing: Evaluating the stability of models in real-world environments.
Generative models: Using adversarial learning principles in GANs to train robust models.

Adversarial learning

Adversarial attacks generate input perturbations that exploit the model's weaknesses.

These attacks can be used to fool models into making wrong predictions.

Adversarial learning

Adversarial learning has to do with robustness of learning algorithm, Szegedy et al. (2013), "are neural network stables?".

"Adversarial examples are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake," Bengio et al. (2017)

Adversarial learning

Adversarial learning deals with the problem that the distributions we obtain IRL are not the ones we train the model on... and we try to quantify what can go wrong

Popular in pictures (what happens if we rotate an object, add glasses to people, etc). Brittleness of ML algorithms...

Problem of data pollution (add outliers) and problems of adversarial examples.

Machine learning perspective

$$\min_{\boldsymbol{\theta}} \left\{ \mathbb{E}_{(\boldsymbol{X},Y)\sim\mathbb{P}} \left[\ell(m_{\boldsymbol{\theta}}(\boldsymbol{X}), Y) \right] \right\}$$

Adversarial perspective

$$\max_{\boldsymbol{\varepsilon}\in\mathcal{E}} \left\{ \mathbb{E}_{(\boldsymbol{X},Y)\sim\mathbb{P}} \left[\ell(m_{\boldsymbol{\theta}}(\boldsymbol{X}+\boldsymbol{\varepsilon}), Y) \right] \right\}$$

leads to robust learning...

Adversarial learning

$$\overbrace{\min_{\boldsymbol{\theta}} \Big\{ \underbrace{\max_{\varepsilon \in \mathcal{E}} \big\{ \mathbb{E}_{(\boldsymbol{X},Y)\sim\mathbb{P}} \big[\ell(m_{\boldsymbol{\theta}}(\boldsymbol{X}+\varepsilon), Y) \big] \big\}}_{\text{creating an adversarial example}} \Big\}}^{\text{training a robust classifier}}$$

Approaches based on robust optimization, Ben-Tal et al. (2009), e.g., Danskin's Theorem, Danskin (1967),

$$\nabla_{\boldsymbol{\theta}} \max_{\varepsilon \in \mathcal{E}} \big\{ \ell(m_{\boldsymbol{\theta}}(\boldsymbol{X}+\varepsilon), Y) \big\} = \nabla_{\boldsymbol{\theta}} \ell(m_{\boldsymbol{\theta}}(\boldsymbol{X}+\varepsilon^\star), Y)$$

where $\varepsilon^\star = \underset{\varepsilon \in \mathcal{E}}{\text{argmax}} \big\{ \ell(m_{\boldsymbol{\theta}}(\boldsymbol{X}+\varepsilon), Y) \big\}$.

Recall the minimax theorem from von Neumann (1928)

Adversarial learning

Proposition 2.7: Nash equilibrium and Minimax

Let A be some $m \times n$ real-valued matrix, there is a Nash equilibrium $(\boldsymbol{x}_\star, \boldsymbol{y}_\star)$ associated with A if

$$\boldsymbol{y}_\star^\top A \boldsymbol{x}_\star = \max_{\boldsymbol{x} \in \mathcal{S}_m} \min_{\boldsymbol{y} \in \mathcal{S}_n} \left\{ \boldsymbol{y}^\top A \boldsymbol{x} \right\} = \min_{\boldsymbol{y} \in \mathcal{S}_n} \max_{\boldsymbol{x} \in \mathcal{S}_m} \left\{ \boldsymbol{y}^\top A \boldsymbol{x} \right\}.$$

Adversarial learning

Consider a Minimax games: given that the discriminator will try to do the best job it can, the generator is set to make the discriminator as wrong as possible

$$\min_{\boldsymbol{\theta}_g} \max_{\boldsymbol{\theta}_d} \{\mathbb{E}_{\boldsymbol{X}\sim\mathbb{P}}[\log(m_{\boldsymbol{\theta}_d}(\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{Z}\sim\mathbb{Q}}[\log(1 - m_{\boldsymbol{\theta}_d}(G_{\boldsymbol{\theta}_d}(\boldsymbol{z}))]\}$$

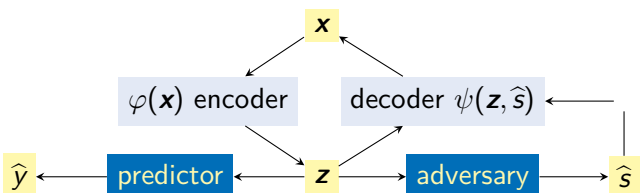
where $\boldsymbol{X} \sim \mathbb{P}$ denotes data sampled from the training data, while $\boldsymbol{Z} \sim \mathbb{Q}$ are sampled by the opponent

See Wadsworth et al. (2018), Xu et al. (2021), Lima et al. (2022) for achieving fairness through adversarial learning

Adversarial learning

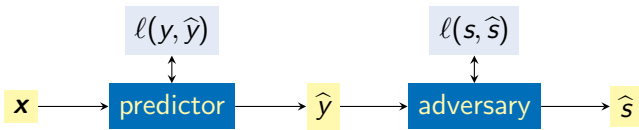
FairGAN, Xu et al. (2018)
Pre-processing approach actually, with demographic parity (DP)

Other algorithms are in-processing approaches, with demographic parity (DP) and equalized odds (EO)
Learning adversarially fair and transferable representations, Madras et al. (2018)



Adversarially learning fair representations, Beutel et al. (2017)
Fair Adversarial Debiasing Approach, Zhang et al. (2018)

Adversarial learning



Following Zhang et al. (2018)
the predictor predicts y given $\boldsymbol{x}$,
the adversary tries to predict s bases on the output of the predictor

the predictor targets to increase its prediction accuracy
and tries to increase the adversary's loss

Sequential learning

Classical supervised machine learning, the learner

- observes training data with labels,
- builds a program to minimize the training error
- controls the error of new data if they are similar to the training data

but

- the environment may evolve over time
- the data may be available sequentially

Sequential learning

At each time step $t = 1, \cdots, T$

- the player observes a context $x_t \in \mathcal{X}$ (optional step)
- the player chooses an action $\theta_t \in \Theta$ (compact decision/parameter set);
- the environment chooses a loss function $\ell_t : \Theta \to [0, 1]$;
- the player suffers loss $\ell_t(\theta_t)$ and observes
  - the losses of every actions: $\ell_t(\theta)$, for all $\theta \in \Theta$: full-information feedback
  - the loss of the chosen action only: $\ell_t(\theta_t)$: bandit feedback.

Goal. Minimize the cumulative loss $\sum_{t=1}^{T} \ell_t()$

Multi-armed bandit

Multi-Armed Bandit

In probability theory and machine learning, the multi-armed bandit problem (sometimes called the K-armed bandit problem) is a problem in which a decision maker iteratively selects one of multiple fixed choices (i.e., arms or actions) when the properties of each choice are only partially known at the time of allocation, and may become better understood as time passes. A fundamental aspect of bandit problems is that choosing an arm does not affect the properties of the arm or other arms. The multi-armed bandit problem is a classic reinforcement learning problem that exemplifies the exploration–exploitation tradeoff dilemma. W

Multi-armed bandit

Exploration–exploitation dilemma

The exploration–exploitation dilemma, also known as the explore–exploit tradeoff, is a fundamental concept in decision-making. It is depicted as the balancing act between two opposing strategies. Exploitation involves choosing the best option based on current knowledge of the system (which may be incomplete or misleading), while exploration involves trying out new options that may lead to better outcomes in the future at the expense of an exploitation opportunity. Finding the optimal balance between these two strategies is a crucial challenge in many decision-making problems whose goal is to maximize long-term benefits W

Multi-armed bandit

Consider K possible actions (arms)
Each arm k is associated an unknown probability distribution with mean μ_k
sequentially pick an arm k_t and get reward $X_{k,t}$, with mean $\mu_{k,t}$

Goal: maximize the expected cumulative reward $\mathbb{E}\left[\sum_{t=1}^{T} X_{k,t}\right]$

Exploration vs Exploitation trade-off.
Historical motivation, Thompson (1933): clinical trials, for each patient t in a clinical study

- choose a treatment k_t
- observe response to the treatment X_{kt} Goal: maximize the number of patient healed (or find the best treatment)

Successful because of many applications coming from Internet: recommender systems, online advertisements,

Multi-armed bandit

At each time step $t = 1, \cdots, T$

- the player observes a context $x_t \in \mathcal{X}$ (optional step)
- the player chooses an action $\theta_t = k_t \in \Theta = \{1, \cdots, K\}$ (compact decision/parameter set);
- the environment chooses a loss function (by sampling the arms);
- the player suffers loss and observes
  - the losses of every actions: full-information feedback
  - the loss of the chosen action only: bandit feedback.

The goal of the player is to minimize his cumulative loss:

$$r_T := \sum_{t=1}^{T} \ell_t(\theta_t)$$

REPRENDRE

Multi-armed bandit

Definition 2.26: Regret

The regret of the player with respect to a fixed parameter $\theta^\star \in \Theta$ after T time steps is

$$r_T(\theta^\star) := \sum_{t=1}^{T} \ell_t(\theta_t) - \sum_{t=1}^{T} \ell_t(\theta^\star)$$

The regret (or uniform regret) is defined as $r_T^\star = \sup_{\theta^\star \in \Theta} \{r_T(\theta^\star)\}$

Multi-armed bandit

Instead of a deterministic choice, one can consider a probability distribution,

$$\theta_t \in \mathcal{S}_K = \left\{ \boldsymbol{p} \in [0,1]^K : \sum_{k=1}^{K} p_k = 1 \right\}$$

How to choose the weights At round t the player needs to choose a weight vector $\theta_t \in \mathcal{S}_K$. How to choose the weights? The player should

- give more weight to actions that performed well in the past.
- not give all the weight to the current best action, otherwise it would not work

The exponentially weighted average forecaster (EWA) also called Hedge performs this trade-off by choosing a weight that decreases exponentially fast with the past errors.

Multi-armed bandit

Algorithm 5: Exponentially Weighted Average forecaster

initialization : $\eta > 0$ and $\boldsymbol{p}_1 \propto (1, 1, \cdots, 1)$;

for *t=1,2,...,T* **do**

  select $\boldsymbol{p}_1$; incur loss $\ell_t(\boldsymbol{p}_t) = \boldsymbol{p}_t^\top \boldsymbol{g}_t$ and observe $\boldsymbol{g}_t \in [-1, +1]^K$; update for all k;

  $$p_{t+1}(k) \leftarrow \frac{\exp\left[-\eta \sum_{i=1}^t g_i(k)\right]}{\sum_{j=1}^K \exp\left[-\eta \sum_{i=1}^t g_i(j)\right]}$$

In that case, it is possible to get an upper bound for regret

Multi-armed bandit

Proposition 2.8: Regret bound for EWA

Given T, for all sequences of loss vectors $\boldsymbol{g}_1, \cdots, \boldsymbol{g}_T \in [1, +1]^K$, EWA achieves the bound

$$r_T := \sum_{t=1}^{T} \ell_t(\boldsymbol{p}_t) - \min_{\boldsymbol{p} \in \mathcal{S}_K} \sum_{t=1}^{T} \ell_t(\boldsymbol{p}) \leq \eta \sum_{t=1}^{T} \sum_{k=1}^{K} p_t(k) g_t(k)^2 + \frac{\log(K)}{\eta}$$

since $\ell_t(\boldsymbol{p}) = \sum_{k=1}^{K} p(k) g_t(k) = \boldsymbol{p}^\top \boldsymbol{g}_t$. If $\eta = \sqrt{\frac{\log(K)}{T}}$, EWA satisfies

$$r_T \leq 2\sqrt{T \log(K)}$$

This regret bound is optimal, see Cesa-Bianchi and Lugosi (2006)

Multi-armed bandit

Reinforcement Learning

Reinforcement learning (RL) is an interdisciplinary area of machine learning and optimal control concerned with how an intelligent agent should take actions in a dynamic environment in order to maximize a reward signal. Reinforcement learning is one of the three basic machine learning paradigms, alongside supervised learning and unsupervised learning. W

Bayesianism as a learning process

Markov Decision Process

Markov decision process (MDP), also called a stochastic dynamic program or stochastic control problem, is a model for sequential decision making when outcomes are uncertain. W

Reinforcement learning

Reinforcement learning is an interdisciplinary area of machine learning and optimal control that has, as main objective, finding an approximately optimal policy for MDPs where transition probabilities and rewards are unknown. W

Bayesianism as a learning process

Thompson sampling (or posterior sampling and probability matching), by Thompson (1933, 1935), and Beta-Bernoulli bandits.

> **Thompson sampling**
> Thompson sampling is a heuristic for choosing actions that address the exploration–exploitation dilemma in the multi-armed bandit problem. It consists of choosing the action that maximizes the expected reward with respect to a randomly drawn belief. W

See Russo et al. (2018)

Bayesianism as a learning process

Thompson sampling (or posterior sampling and probability matching), by Thompson (1933, 1935), and Beta-Bernoulli bandits.

We have to choose among K alternatives, that yield $\boldsymbol{X} = (X_1, \cdots, X_K)$, $X_k \sim \mathcal{B}(\theta_k)$. Assume (prior) $\theta_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$. At time t, draw K Beta variables (independents) $B_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$, and select $k^\star = \underset{k=1,\cdots,K}{\text{argmin}} \{B_k\}$.

Consider updating $(\alpha_{k^*}, \beta_{k^*}) \leftarrow (\alpha_{k^*} + x_{k^*}, \beta_{k^*} + (1 - x_{k^*}))$,

- simulated data, i.i.d., $X_1 \sim \mathcal{B}(72\%)$
- simulated data, i.i.d., $X_2 \sim \mathcal{B}(24\%)$

Bayesianism as a learning process

Thompson sampling (or posterior sampling and probability matching), by Thompson (1933, 1935), and Beta-Bernoulli bandits.

We have to choose among K alternatives, that yield $\boldsymbol{X} = (X_1, \cdots, X_K)$, $X_k \sim \mathcal{B}(\theta_k)$. Assume (prior) $\theta_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$. At time t, draw K Beta variables (independents) $B_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$, and select $k^\star = \underset{k=1,\cdots,K}{\text{argmin}} \{B_k\}$.

Consider updating $(\alpha_{k^*}, \beta_{k^*}) \leftarrow (\alpha_{k^*} + x_{k^*}, \beta_{k^*} + (1 - x_{k^*}))$,

- simulated data, i.i.d., $X_1 \sim \mathcal{B}(72\%)$
- simulated data, i.i.d., $X_2 \sim \mathcal{B}(24\%)$

Bayesianism as a learning process

Thompson sampling (or posterior sampling and probability matching), by Thompson (1933, 1935), and Beta-Bernoulli bandits.

We have to choose among K alternatives, that yield $\boldsymbol{X} = (X_1, \cdots, X_K)$, $X_k \sim \mathcal{B}(\theta_k)$. Assume (prior) $\theta_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$. At time t, draw K Beta variables (independents) $B_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$, and select $k^\star = \underset{k=1,\cdots,K}{\operatorname{argmin}} \{B_k\}$.

Consider updating $(\alpha_{k^*}, \beta_{k^*}) \leftarrow (\alpha_{k^*} + x_{k^*}, \beta_{k^*} + (1 - x_{k^*}))$,

- simulated data, i.i.d., $X_1 \sim \mathcal{B}(72\%)$
- simulated data, i.i.d., $X_2 \sim \mathcal{B}(24\%)$

Bayesianism as a learning process

Thompson sampling (or posterior sampling and probability matching), by Thompson (1933, 1935), and Beta-Bernoulli bandits.

We have to choose among K alternatives, that yield $\boldsymbol{X} = (X_1, \cdots, X_K)$, $X_k \sim \mathcal{B}(\theta_k)$. Assume (prior) $\theta_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$. At time t, draw K Beta variables (independents) $B_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$, and select $k^\star = \underset{k=1,\cdots,K}{\text{argmin}} \{B_k\}$.

Consider updating $(\alpha_{k^*}, \beta_{k^*}) \leftarrow (\alpha_{k^*} + x_{k^*}, \beta_{k^*} + (1 - x_{k^*}))$,

- simulated data, i.i.d., $X_1 \sim \mathcal{B}(72\%)$
- simulated data, i.i.d., $X_2 \sim \mathcal{B}(24\%)$

Bayesianism as a learning process

Thompson sampling (or posterior sampling and probability matching), by Thompson (1933, 1935), and Beta-Bernoulli bandits.

We have to choose among K alternatives, that yield $\boldsymbol{X} = (X_1, \cdots, X_K)$, $X_k \sim \mathcal{B}(\theta_k)$. Assume (prior) $\theta_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$. At time t, draw K Beta variables (independents) $B_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$, and select $k^\star = \underset{k=1,\cdots,K}{\operatorname{argmin}} \{B_k\}$.

Consider updating $(\alpha_{k^*}, \beta_{k^*}) \leftarrow (\alpha_{k^*} + x_{k^*}, \beta_{k^*} + (1 - x_{k^*}))$,

- simulated data, i.i.d., $X_1 \sim \mathcal{B}(72\%)$
- simulated data, i.i.d., $X_2 \sim \mathcal{B}(24\%)$

Bayesianism as a learning process

Thompson sampling (or posterior sampling and probability matching), by Thompson (1933, 1935), and Beta-Bernoulli bandits.

We have to choose among K alternatives, that yield $\boldsymbol{X} = (X_1, \cdots, X_K)$, $X_k \sim \mathcal{B}(\theta_k)$. Assume (prior) $\theta_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$. At time t, draw K Beta variables (independents) $B_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$, and select $k^\star = \underset{k=1,\cdots,K}{\operatorname{argmin}} \{B_k\}$.

Consider updating $(\alpha_{k^*}, \beta_{k^*}) \leftarrow (\alpha_{k^*} + x_{k^*}, \beta_{k^*} + (1 - x_{k^*}))$,

- simulated data, i.i.d., $X_1 \sim \mathcal{B}(72\%)$
- simulated data, i.i.d., $X_2 \sim \mathcal{B}(24\%)$

Bayesianism as a learning process

Thompson sampling (or posterior sampling and probability matching), by Thompson (1933, 1935), and Beta-Bernoulli bandits.

We have to choose among K alternatives, that yield $\boldsymbol{X} = (X_1, \cdots, X_K)$, $X_k \sim \mathcal{B}(\theta_k)$. Assume (prior) $\theta_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$. At time t, draw K Beta variables (independents) $B_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$, and select $k^\star = \underset{k=1,\cdots,K}{\text{argmin}} \{B_k\}$.

Consider updating $(\alpha_{k^*}, \beta_{k^*}) \leftarrow (\alpha_{k^*} + x_{k^*}, \beta_{k^*} + (1 - x_{k^*}))$,

- simulated data, i.i.d., $X_1 \sim \mathcal{B}(72\%)$
- simulated data, i.i.d., $X_2 \sim \mathcal{B}(24\%)$

Bayesianism as a learning process

Thompson sampling (or posterior sampling and probability matching), by Thompson (1933, 1935), and Beta-Bernoulli bandits.

We have to choose among K alternatives, that yield $\boldsymbol{X} = (X_1, \cdots, X_K)$, $X_k \sim \mathcal{B}(\theta_k)$. Assume (prior) $\theta_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$. At time t, draw K Beta variables (independents) $B_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$, and select $k^\star = \underset{k=1,\cdots,K}{\operatorname{argmin}} \{B_k\}$.

Consider updating $(\alpha_{k^*}, \beta_{k^*}) \leftarrow (\alpha_{k^*} + x_{k^*}, \beta_{k^*} + (1 - x_{k^*}))$,

- simulated data, i.i.d., $X_1 \sim \mathcal{B}(72\%)$
- simulated data, i.i.d., $X_2 \sim \mathcal{B}(24\%)$

Bayesianism as a learning process

Thompson sampling (or posterior sampling and probability matching), by Thompson (1933, 1935), and Beta-Bernoulli bandits.

We have to choose among K alternatives, that yield $\boldsymbol{X} = (X_1, \cdots, X_K)$, $X_k \sim \mathcal{B}(\theta_k)$. Assume (prior) $\theta_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$. At time t, draw K Beta variables (independents) $B_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$, and select $k^\star = \underset{k=1,\cdots,K}{\operatorname{argmin}} \{B_k\}$.

Consider updating $(\alpha_{k^*}, \beta_{k^*}) \leftarrow (\alpha_{k^*} + x_{k^*}, \beta_{k^*} + (1 - x_{k^*}))$,

- simulated data, i.i.d., $X_1 \sim \mathcal{B}(72\%)$
- simulated data, i.i.d., $X_2 \sim \mathcal{B}(24\%)$

Bayesianism as a learning process

Thompson sampling (or posterior sampling and probability matching), by Thompson (1933, 1935), and Beta-Bernoulli bandits.

We have to choose among K alternatives, that yield $\boldsymbol{X} = (X_1, \cdots, X_K)$, $X_k \sim \mathcal{B}(\theta_k)$. Assume (prior) $\theta_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$. At time t, draw K Beta variables (independents) $B_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$, and select $k^\star = \underset{k=1,\cdots,K}{\text{argmin}} \{B_k\}$.

Consider updating $(\alpha_{k^*}, \beta_{k^*}) \leftarrow (\alpha_{k^*} + x_{k^*}, \beta_{k^*} + (1 - x_{k^*}))$,

- simulated data, i.i.d., $X_1 \sim \mathcal{B}(72\%)$
- simulated data, i.i.d., $X_2 \sim \mathcal{B}(24\%)$

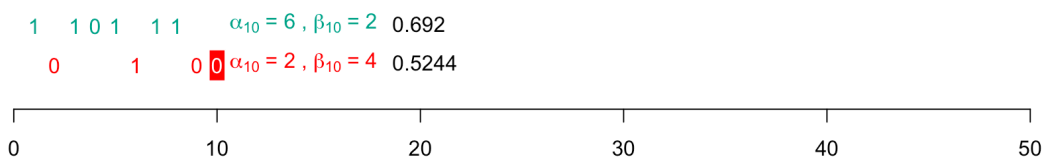
Bayesianism as a learning process

Thompson sampling (or posterior sampling and probability matching), by Thompson (1933, 1935), and Beta-Bernoulli bandits.

We have to choose among K alternatives, that yield $\boldsymbol{X} = (X_1, \cdots, X_K)$, $X_k \sim \mathcal{B}(\theta_k)$. Assume (prior) $\theta_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$. At time t, draw K Beta variables (independents) $B_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$, and select $k^\star = \underset{k=1,\cdots,K}{\operatorname{argmin}} \{B_k\}$.

Consider updating $(\alpha_{k^*}, \beta_{k^*}) \leftarrow (\alpha_{k^*} + x_{k^*}, \beta_{k^*} + (1 - x_{k^*}))$,

- simulated data, i.i.d., $X_1 \sim \mathcal{B}(72\%)$
- simulated data, i.i.d., $X_2 \sim \mathcal{B}(24\%)$



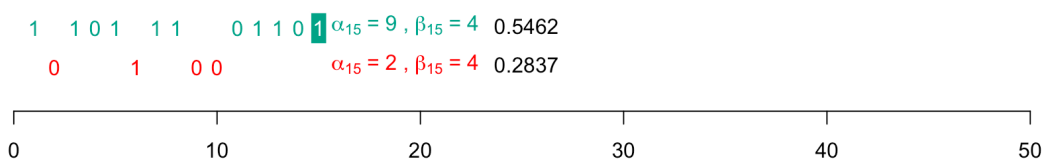
Bayesianism as a learning process

Thompson sampling (or posterior sampling and probability matching), by Thompson (1933, 1935), and Beta-Bernoulli bandits.

We have to choose among K alternatives, that yield $\boldsymbol{X} = (X_1, \cdots, X_K)$, $X_k \sim \mathcal{B}(\theta_k)$. Assume (prior) $\theta_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$. At time t, draw K Beta variables (independents) $B_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$, and select $k^\star = \underset{k=1,\cdots,K}{\operatorname{argmin}} \{B_k\}$.

Consider updating $(\alpha_{k^*}, \beta_{k^*}) \leftarrow (\alpha_{k^*} + x_{k^*}, \beta_{k^*} + (1 - x_{k^*}))$,

- simulated data, i.i.d., $X_1 \sim \mathcal{B}(72\%)$
- simulated data, i.i.d., $X_2 \sim \mathcal{B}(24\%)$



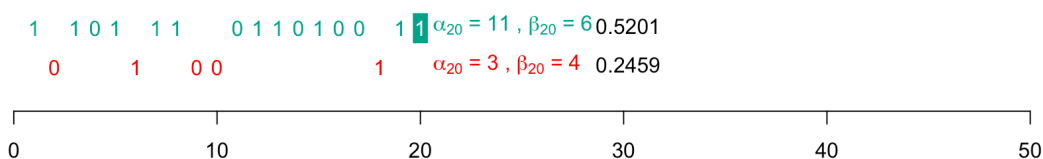
Bayesianism as a learning process

Thompson sampling (or posterior sampling and probability matching), by Thompson (1933, 1935), and Beta-Bernoulli bandits.

We have to choose among K alternatives, that yield $\boldsymbol{X} = (X_1, \cdots, X_K)$, $X_k \sim \mathcal{B}(\theta_k)$. Assume (prior) $\theta_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$. At time t, draw K Beta variables (independents) $B_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$, and select $k^\star = \underset{k=1,\cdots,K}{\text{argmin}} \{B_k\}$.

Consider updating $(\alpha_{k^*}, \beta_{k^*}) \leftarrow (\alpha_{k^*} + x_{k^*}, \beta_{k^*} + (1 - x_{k^*}))$,

- simulated data, i.i.d., $X_1 \sim \mathcal{B}(72\%)$
- simulated data, i.i.d., $X_2 \sim \mathcal{B}(24\%)$



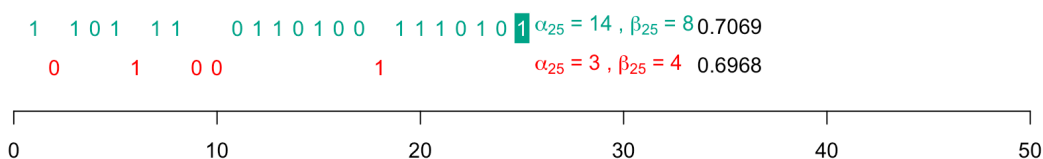
Bayesianism as a learning process

Thompson sampling (or posterior sampling and probability matching), by Thompson (1933, 1935), and Beta-Bernoulli bandits.

We have to choose among K alternatives, that yield $\boldsymbol{X} = (X_1, \cdots, X_K)$, $X_k \sim \mathcal{B}(\theta_k)$. Assume (prior) $\theta_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$. At time t, draw K Beta variables (independents) $B_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$, and select $k^\star = \underset{k=1,\cdots,K}{\operatorname{argmin}} \{B_k\}$.

Consider updating $(\alpha_{k^*}, \beta_{k^*}) \leftarrow (\alpha_{k^*} + x_{k^*}, \beta_{k^*} + (1 - x_{k^*}))$,

- simulated data, i.i.d., $X_1 \sim \mathcal{B}(72\%)$
- simulated data, i.i.d., $X_2 \sim \mathcal{B}(24\%)$



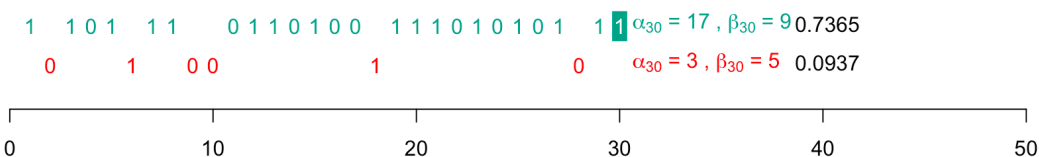
Bayesianism as a learning process

Thompson sampling (or posterior sampling and probability matching), by Thompson (1933, 1935), and Beta-Bernoulli bandits.

We have to choose among K alternatives, that yield $\boldsymbol{X} = (X_1, \cdots, X_K)$, $X_k \sim \mathcal{B}(\theta_k)$. Assume (prior) $\theta_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$. At time t, draw K Beta variables (independents) $B_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$, and select $k^\star = \underset{k=1,\cdots,K}{\operatorname{argmin}} \{B_k\}$.

Consider updating $(\alpha_{k^*}, \beta_{k^*}) \leftarrow (\alpha_{k^*} + x_{k^*}, \beta_{k^*} + (1 - x_{k^*}))$,

- simulated data, i.i.d., $X_1 \sim \mathcal{B}(72\%)$
- simulated data, i.i.d., $X_2 \sim \mathcal{B}(24\%)$



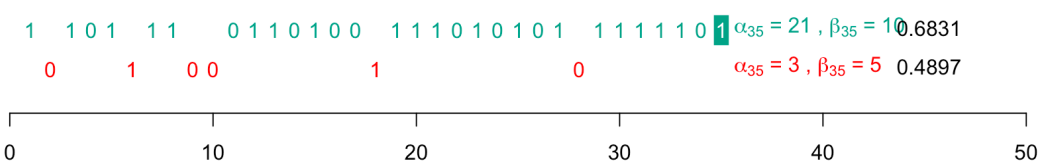
Bayesianism as a learning process

Thompson sampling (or posterior sampling and probability matching), by Thompson (1933, 1935), and Beta-Bernoulli bandits.

We have to choose among K alternatives, that yield $\boldsymbol{X} = (X_1, \cdots, X_K)$, $X_k \sim \mathcal{B}(\theta_k)$. Assume (prior) $\theta_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$. At time t, draw K Beta variables (independents) $B_k \sim \mathcal{B}eta(\alpha_k, \beta_k)$, and select $k^\star = \underset{k=1,\cdots,K}{\operatorname{argmin}} \{B_k\}$.

Consider updating $(\alpha_{k^*}, \beta_{k^*}) \leftarrow (\alpha_{k^*} + x_{k^*}, \beta_{k^*} + (1 - x_{k^*}))$,

- simulated data, i.i.d., $X_1 \sim \mathcal{B}(72\%)$
- simulated data, i.i.d., $X_2 \sim \mathcal{B}(24\%)$



Bayesianism as a learning process

We can use that approach in the context of Monty Hall

- strategy 1 : always switch the door
- strategy 2 : never switch the door

Bayesianism as a learning process

We can use that approach in the context of Monty Hall

- strategy 1 : always switch the door
- strategy 2 : never switch the door

Bayesianism as a learning process

We can use that approach in the context of Monty Hall

- strategy 1 : always switch the door
- strategy 2 : never switch the door

Bayesianism as a learning process

We can use that approach in the context of Monty Hall

- strategy 1 : always switch the door
- strategy 2 : never switch the door

Bayesianism as a learning process

We can use that approach in the context of Monty Hall

- strategy 1 : always switch the door
- strategy 2 : never switch the door

Bayesianism as a learning process

We can use that approach in the context of Monty Hall

- strategy 1 : always switch the door
- strategy 2 : never switch the door

Bayesianism as a learning process

We can use that approach in the context of Monty Hall

- strategy 1 : always switch the door
- strategy 2 : never switch the door

Bayesianism as a learning process

We can use that approach in the context of Monty Hall

- strategy 1 : always switch the door
- strategy 2 : never switch the door

Bayesianism as a learning process

We can use that approach in the context of Monty Hall

- strategy 1 : always switch the door
- strategy 2 : never switch the door

Bayesianism as a learning process

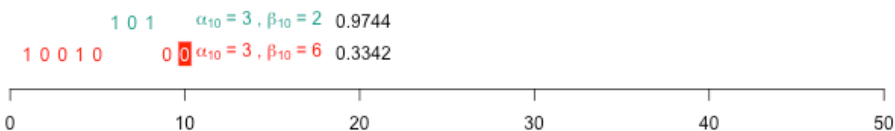
We can use that approach in the context of Monty Hall

- strategy 1 : always switch the door
- strategy 2 : never switch the door

Bayesianism as a learning process

We can use that approach in the context of Monty Hall

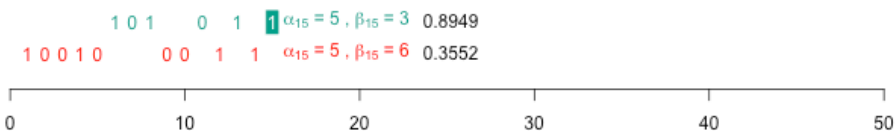
- strategy 1 : always switch the door
- strategy 2 : never switch the door



Bayesianism as a learning process

We can use that approach in the context of Monty Hall

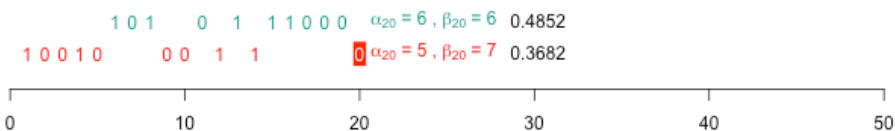
- strategy 1 : always switch the door
- strategy 2 : never switch the door



Bayesianism as a learning process

We can use that approach in the context of Monty Hall

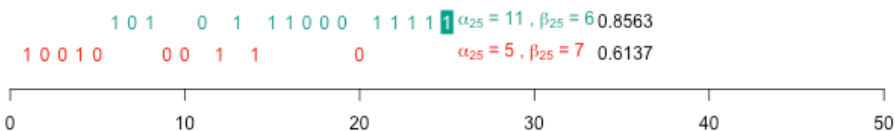
- strategy 1 : always switch the door
- strategy 2 : never switch the door



Bayesianism as a learning process

We can use that approach in the context of Monty Hall

- strategy 1 : always switch the door
- strategy 2 : never switch the door



Bayesianism as a learning process

We can use that approach in the context of Monty Hall

- strategy 1 : always switch the door
- strategy 2 : never switch the door

References

Avraham, R. (2017). Discrimination and insurance. In Lippert-Rasmussen, K., editor, *Handbook of the Ethics of Discrimination*, pages 335–347. Routledge.

Bailey, R. A. and Simon, L. J. (1960). Two studies in automobile insurance ratemaking. *ASTIN Bulletin: The Journal of the IAA*, 1(4):192–217.

Ben-Tal, A., El Ghaoui, L., and Nemirovski, A. (2009). *Robust optimization*, volume 28. Princeton university press.

Bengio, Y., Goodfellow, I., and Courville, A. (2017). *Deep learning*, volume 1. MIT press.

Beutel, A., Chen, J., Zhao, Z., and Chi, E. H. (2017). Data decisions and theoretical implications when adversarially learning fair representations. *arXiv*, 1707.00075.

Blanpain, N. (2018). L'espérance de vie par niveau de vie-méthode et principaux résultats. *INSEE Document de Travail*, F1801.

Box, G. E., Luceño, A., and del Carmen Paniagua-Quinones, M. (2011). *Statistical control by monitoring and adjustment*, volume 700. John Wiley & Sons.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and regression trees*. Taylor & Francis.

References

Brilmayer, L., Hekeler, R. W., Laycock, D., and Sullivan, T. A. (1979). Sex discrimination in employer-sponsored insurance plans: A legal and demographic analysis. *University of Chicago Law Review*, 47:505.

Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge university press.

Charpentier, A. (2014). Mesures de risque. In Droesbeke, J.-J. and Saporta, G., editors, *Approches statistiques du risque*. Éditions Technip.

Condorcet, J. A. N. (1785). *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Imprimerie royale.

Danskin, J. M. (1967). *The theory of max-min and its application to weapons allocation problems*. Springer.

De Leeuw, J., Hornik, K., and Mair, P. (2010). Isotone optimization in r: pool-adjacent-violators algorithm (pava) and active set methods. *Journal of statistical software*, 32:1–24.

De Pril, N. and Dhaene, J. (1996). Segmentering in verzekeringen. *DTEW Research Report 9648*, pages 1–56.

De Wit, G. and Van Eeghen, J. (1984). Rate making and society's sense of fairness. *ASTIN Bulletin: The Journal of the IAA*, 14(2):151–163.

References

Denuit, M. and Charpentier, A. (2004). *Mathématiques de l'assurance non-vie: Tome I Principes fondamentaux de théorie du risque*. Economica.

Eckart, C. and Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218.

Feeley, M. M. and Simon, J. (1992). The new penology: Notes on the emerging strategy of corrections and its implications. *Criminology*, 30(4):449–474.

Friedman, J., Hastie, T., Tibshirani, R., et al. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York.

Galton, F. (1907). Vox populi (the wisdom of crowds). *Nature*, 75(7):450–451.

Gandy, O. H. (2016). *Coming to terms with chance: Engaging rational discrimination and cumulative disadvantage*. Routledge.

Glenn, B. J. (2003). Postmodernism: the basis of insurance. *Risk Management and Insurance Review*, 6(2):131–143.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*. MIT Press.

Harcourt, B. E. (2015). Risk as a proxy for race: The dangers of risk assessment. *Federal Sentencing Reporter*, 27(4):237–243.

References

Hoerl, A. E. and Kennard, R. W. (1970a). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55.

Hoerl, A. E. and Kennard, R. W. (1970b). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Hoffman, F. L. (1896). *Race traits and tendencies of the American Negro*, volume 11. American Economic Association.

Hoffman, F. L. (1918). *Mortality from respiratory diseases in dusty trades (inorganic dusts)*. Number 231. US Government Printing Office.

Hoffman, F. L. (1931). Cancer and smoking habits. *Annals of surgery*, 93(1):50.

James, G., Witten, D., Hastie, T., Tibshirani, R., et al. (2013). *An introduction to statistical learning*, volume 112. Springer.

Kaufman, L. and Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons.

Kearns, M. (1988). Thoughts on hypothesis boosting. *Unpublished manuscript*, 45:105.

Kearns, M. and Valiant, L. (1989). Cryptographic limitations on learning boolean formulae and finite automata. *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, 41(1):67–95.

References

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv*, 1312.6114.

Lima, L. F. F. P. d., Ricarte, D. R. D., and Siebra, C. d. A. (2022). An overview on the use of adversarial learning strategies to ensure fairness in machine learning models. In *XVIII Brazilian Symposium on Information Systems*, pages 1–8.

Madras, D., Creager, E., Pitassi, T., and Zemel, R. (2018). Learning adversarially fair and transferable representations. In *International Conference on Machine Learning*, pages 3384–3393. PMLR.

McCullagh, P. and Nelder, J. (1989). *Generalized linear models*. Chapman & Hall.

Mienye, I. D. and Sun, Y. (2022). A survey of ensemble learning: Concepts, algorithms, applications, and prospects. *Ieee Access*, 10:99129–99149.

Mirsky, L. (1960). Symmetric gauge functions and unitarily invariant norms. *The quarterly journal of mathematics*, 11(1):50–59.

Mowbray, A. (1921). Classification of risks as the basis of insurance rate making with special reference to workmen's compensation. *Proceedings of the Casualty Actuarial Society*.

Neumann, J. v. and Morgenstern, O. (1947). *Theory of games and economic behavior*. Princeton University Press.

Platt, J. et al. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

References

Russo, D. J., Van Roy, B., Kazerouni, A., Osband, I., Wen, Z., et al. (2018). A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96.

Schervish, M. J. and DeGroot, M. H. (2014). *Probability and statistics*, volume 563. Pearson Education London, UK:.

Simon, J. (1987). The emergence of a risk society-insurance, law, and the state. *Socialist Review*, (95):60–89.

Simon, J. (1988). The ideological effects of actuarial practices. *Law & Society Review*, 22:771.

Struyck, N. (1912). *Les oeuvres de Nicolas Struyck (1687-1769): qui se rapportent au calcul des chances, à la statistique général, ż la statistique des décès et aux rentes viagèter*. Société générale néerlandaise d'assurances sur la vie et de rentes viagères.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv*, 1312.6199.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294.

Thompson, W. R. (1935). On the theory of apportionment. *American Journal of Mathematics*, 57(2):450–456.

References

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Tukey, J. W. (1961). Curves as parameters, and touch estimation. In Neyman, J., editor, *Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 681–694. University of California Press, University of California Press.

Tweedie, M. C. K. (1984). An index which distinguishes between some important exponential families. *Statistics: applications and new directions (Calcutta, 1981)*, pages 579–604.

Vapnik, V. N., Vapnik, V., et al. (1998). Statistical learning theory.

Vito, E. D., Rosasco, L., Caponnetto, A., Piana, M., and Verri, A. (2004). Some properties of regularized kernel methods. *Journal of Machine Learning Research*, 5(Oct):1363–1390.

von Neumann, J. (1928). Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1):295–320.

Wadsworth, C., Vera, F., and Piech, C. (2018). Achieving fairness through adversarial learning: an application to recidivism prediction. *arXiv*, 1807.00199.

Wahba, G. (1999). Support vector machines, reproducing kernel hilbert spaces and the randomized gacv. *Advances in Kernel Methods-Support Vector Learning*, 6:69–87.

Wahba, G., Lin, X., Gao, F., Xiang, D., Klein, R., and Klein, B. (1998). The bias-variance tradeoff and the randomized gacv. *Advances in Neural Information Processing Systems*, 11.

References

Wortham, L. (1986). The economics of insurance classification: The sound of one invisible hand clapping. *Ohio State Law Journal*, 47:835.

Xu, D., Yuan, S., Zhang, L., and Wu, X. (2018). Fairgan: Fairness-aware generative adversarial networks. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 570–575. IEEE.

Xu, H., Liu, X., Li, Y., Jain, A., and Tang, J. (2021). To be robust or to be fair: Towards fairness in adversarial training. In *International conference on machine learning*, pages 11492–11501. PMLR.

Zhang, B. H., Lemoine, B., and Mitchell, M. (2018). Mitigating unwanted biases with adversarial learning. *arXiv*, 1801.07593:335–340.