Introduction to machine learning

Masters M2MO & MIDS

Stéphane Gaïffas

Université de Paris

Who I am ?

- Pr. Stéphane Gaïffas
- LPSM, Université de Paris
- DMA, Ecole normale supérieure
- `https://stephanegaiffas.github.io/`
- `stephane.gaiffas@math.univ-paris-diderot.fr`

Teasers: data science or statistics?

Jeremy Jarvis
@jeremyjarvis

Follow

"A data scientist is a statistician who lives in San Fransisco" #monkigras pic.twitter.com/HypLL3Cnye

12:13 PM - 30 Jan 2014

1,475 841

Big Data Borat
@BigDataBorat

Follow

Data Science is statistics on a Mac.

3:32 PM - 27 Aug 2013

611 273

Josh Wills
@josh_wills

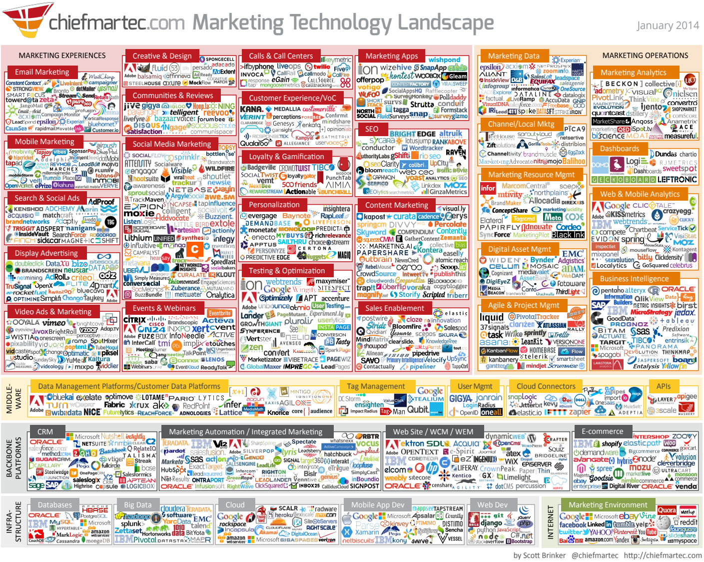
Follow

Data Scientist (n.): Person who is better at statistics than any software engineer and better at software engineering than any statistician.

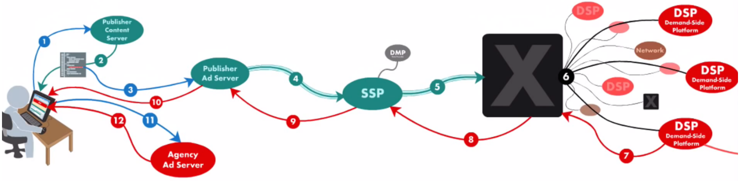
6:55 PM - 3 May 2012

1,360 821

Teasers: an application in marketing



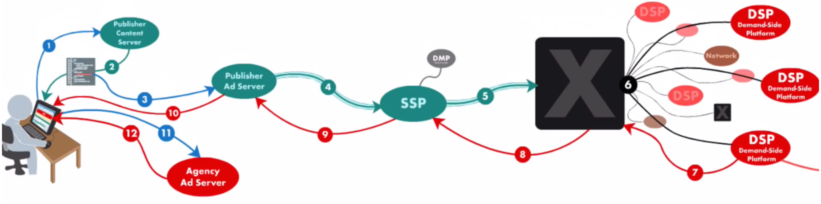
Teasers: an application in marketing (Real Time Bidding)



- A **customer** visits a webpage with his browser: a complex process of content selection and delivery begins.
- An **advertiser** might want to display an ad on the webpage where the user is going. The webpage belongs to a **publisher**.
- The publisher sells ad space to advertisers who want to reach customers

In some cases, an auction starts: **RTB** (Real Time Bidding)

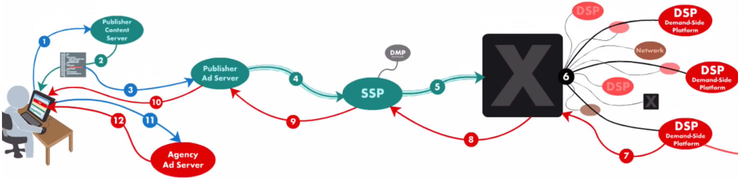
Teasers: an application in marketing (Real Time Bidding)



- Advertisers have **10ms** (!) to give a price: they need to assess quickly how willing they are to display the ad to this customer
- Machine learning is used here to **predict the probability of click on the ad**. Time constraint: few model parameters to answer quickly
- Feature selection / dimension reduction is crucial here

Full process takes < 100ms

Teasers: an application in marketing (Real Time Bidding)



Some figures:

- 10 million prediction of click probability per second
- answers within 10ms
- stores 20Terabytes of data daily

Aim

- Based on past data, you want to find users that will click on some ads

This problem can be formulated as a **binary classification problem**

Classification = **supervised learning** with a binary label

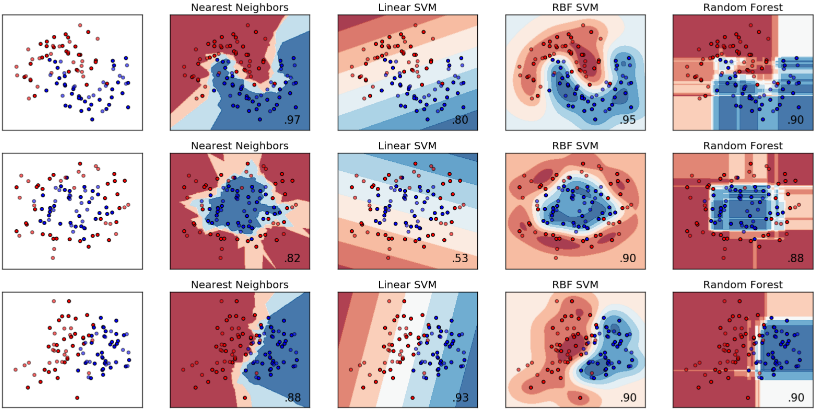
Setting

- You have past/historical data, containing data about individuals $i = 1, \ldots, n$
- You have a **features** vector $x_i \in \mathbb{R}^d$ for each individual i
- For each i, you know if he/she clicked $(y_i = 1)$ or not $(y_i = -1)$
- We call $y_i \in \{-1, 1\}$ the **label** of i
- (x_i, y_i) are i.i.d realizations of (X, Y)

Aim

- Given a features vector x (with no corresponding label), predict a label $\hat{y} \in \{-1, 1\}$
- Use data $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ to construct a **classifier**

Many ways to separate points!



Today: model-based classification

- Naive Bayes
- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA)
- Logistic regression
- Penalization
- Cross-validation

Probabilistic / statistical approach

- Model the distribution of $Y|X$
- Construct estimators $\hat{p}_1(x)$ and $\hat{p}_{-1}(x)$ of

$$p_1(x) = \mathbb{P}(Y = 1|X = x) \quad \text{and} \quad p_{-1}(x) = 1 - p_1(x)$$

- Given x, classify using

$$\hat{y} = \begin{cases} 1 & \text{if } \hat{p}_1(x) \geq t \\ -1 & \text{otherwise} \end{cases}$$

for some threshold $t \in (0, 1)$

Bayes formula. We know that

$$p_y(x) = \mathbb{P}(Y = y | X = x) = \frac{\mathbb{P}(X = x | Y = y)\mathbb{P}(Y = y)}{\mathbb{P}(X = x)}$$
$$= \frac{\mathbb{P}(X = x | Y = y)\mathbb{P}(Y = y)}{\sum_{y'=-1,1} \mathbb{P}(X = x | Y = y')\mathbb{P}(Y = y')}$$

If we know the distribution of $X|Y$ and the distribution of Y, we know the distribution of $Y|X$

Bayes classifier. Classify using Bayes formula, given that:

- We model $\mathbb{P}(X|Y)$
- We are able to estimate $\mathbb{P}(X|Y)$ based on data

Maximum a posteriori. Classify using the *discriminant* functions

$$\delta_y(x) = \log \mathbb{P}(X = x | Y = y) + \log \mathbb{P}(Y = y)$$

for $y = 1, -1$ and decide (largest, beyond a threshold, etc.)

Remark.

- Different models on the distribution of $X|Y$ leads to different classifiers
- The simplest one is the Naive Bayes
- Then, the most standard are Linear Discriminant Analysis (LDA) and Quadratic discriminant Analysis (QDA)

Naive Bayes. A crude modeling for $\mathbb{P}(X|Y)$: assume features X^j are independent conditionally on Y:

$$\mathbb{P}(X = x|Y = y) = \prod_{j=1}^{d} \mathbb{P}(X^j = x^j|Y = y)$$

Model the univariate distribution $X^j|Y$: for instance, assume that

$$\mathbb{P}(X^j|Y = y) = \mathrm{Normal}(\mu_{j,y}, \sigma^2_{j,y}),$$

parameters $\mu_{j,y}$ and $\sigma^2_{j,y}$ easily estimated by MLE

- If the feature X^j is discrete, use a Bernoulli or multinomial distribution
- Leads to a classifier which is very easy to compute
- Requires only the computation of some averages (MLE)

Discriminant Analysis. Assume that

$$\mathbb{P}(X|Y=y) = \text{Normal}(\mu_y, \Sigma_y),$$

where we recall that the density of Normal(μ, Σ) is given by

$$f(x) = \frac{1}{(2\pi)^{d/2}\sqrt{\det \Sigma}} \exp\Big(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\Big)$$

In this case, discriminant functions are

$$\begin{aligned}
\delta_y(x) &= \log \mathbb{P}(X=x|Y=y) + \log \mathbb{P}(Y=y) \\
&= -\frac{1}{2}(x-\mu_y)^\top \Sigma_y^{-1}(x-\mu_y) - \frac{d}{2}\ln(2\pi) \\
&\quad -\frac{1}{2}\log\det\Sigma_y + \log \mathbb{P}(Y=y)
\end{aligned}$$

Estimation. Use "natural" estimators, obtained by maximum likelihood estimation. Define for $y \in \{-1, 1\}$

$$I_y = \{i = 1, \ldots, n \ : \ y_i = y\} \quad \text{and} \quad n_y = |I_y|$$

MLE estimators are given by

$$\hat{\mathbb{P}}(Y = y) = \frac{n_y}{n}, \quad \hat{\mu}_y = \frac{1}{n_y} \sum_{i \in I_y} x_i,$$

$$\hat{\Sigma}_y = \frac{1}{n_y} \sum_{i \in I_y} (x_i - \hat{\mu}_y)(x_i - \hat{\mu}_y)^\top$$

for $y \in \{-1, 1\}$. These are simply the proportion, sample mean and sample covariance within each group of labels

Linear Discriminant Analysis (LDA)

- Assumes that $\Sigma = \Sigma_1 = \Sigma_{-1}$
- All groups have the same correlation structure
- In this case decision function is linear $\langle x, w \rangle \geq c$ with

$$
\begin{aligned}
w &= \Sigma^{-1}(\mu_1 - \mu_{-1}) \\
c &= \frac{1}{2}(\langle \mu_1, \Sigma^{-1}\mu_1 \rangle - \langle \mu_{-1}, \Sigma^{-1}\mu_{-1} \rangle) \\
&\quad + \log\left(\frac{\mathbb{P}(Y = 1 | X = x)}{\mathbb{P}(Y = -1 | X = x)}\right)
\end{aligned}
$$

Quadratic Discriminant Analysis (QDA)

- Assumes that $\Sigma_1 \neq \Sigma_{-1}$
- Decision function is quadratic

cf. Exercice 1 from `exos1.pdf`

Logistic regression

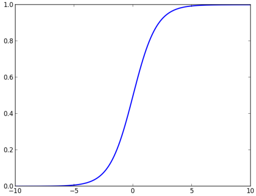
- By far the most widely used classification algorithm
- We want to explain the label y based on x, we want to "regress" y on x
- Models the distribution of $Y|X$

For $y \in \{-1, 1\}$, we consider the model

$$\mathbb{P}(Y = 1|X = x) = \sigma(x^\top w + b)$$

where $w \in \mathbb{R}^d$ is a vector of model **weights** and $b \in \mathbb{R}$ is the **intercept**, and where σ is the **sigmoid** function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



- The sigmoid choice really is a **choice**. It is a **modelling choice**.
- It's a way to map $\mathbb{R} \to [0, 1]$ (we want to model a probability)
- We could also consider

$$\mathbb{P}(Y = 1 | X = x) = F(\langle x, w \rangle + b)$$

for any distribution function F. Another popular choice is the Gaussian distribution

$$F(z) = \mathbb{P}(N(0, 1) \leq z),$$

which leads to another loss called **probit**

- However, the sigmoid choice has the following nice interpretation: an easy computation leads to

$$\log \left(\frac{\mathbb{P}(Y = 1 | X = x)}{\mathbb{P}(Y = -1 | X = x)} \right) = \langle x, w \rangle + b$$

  This quantity is called the **log-odd ratio**
- Note that

$$\mathbb{P}(Y = 1 | X = x) \geq \mathbb{P}(Y = -1 | X = x)$$

  iff

$$\langle x, w \rangle + b \geq 0.$$

- This is a **linear classification** rule
- Linear with respect to the considered features x
- But, **you** choose the features: **features engineering** (more on that later)

Estimation of w and b

- We have a model for $Y|X$
- Data (x_i, y_i) is assumed i.i.d with the same distribution as (X, Y)
- Compute estimators $\hat{w}$ and $\hat{b}$ by **maximum likelihood estimation**
- Or equivalently, minimize the minus log-likelihood
- More generally, when a model is used

$$\text{Goodness-of-fit} = -\text{log likelihood}$$

- log is used mainly since averages are easier to study (and compute) than products

Likelihood is given by

$$\begin{aligned}
&\prod_{i=1}^{n} \mathbb{P}(Y = y_i | X = x_i) \\
&= \prod_{i=1}^{n} \sigma(\langle x_i, w \rangle + b)^{\frac{1+y_i}{2}} \left(1 - \sigma(\langle x_i, w \rangle + b)\right)^{\frac{1-y_i}{2}} \\
&= \prod_{i=1}^{n} \sigma(\langle x_i, w \rangle + b)^{\frac{1+y_i}{2}} \sigma(-\langle x_i, w \rangle - b)^{\frac{1-y_i}{2}}
\end{aligned}$$

and the minus log-likelihood is given by

$$\sum_{i=1}^{n} \log(1 + e^{-y_i(\langle x_i, w \rangle + b)})$$

Compute $\hat{w}$ and $\hat{b}$ as follows:

$$(\hat{w}, \hat{b}) \in \operatorname*{argmin}_{w \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} \log(1 + e^{-y_i(\langle x_i, w\rangle + b)})$$

- It is a convex and smooth problem
- Many ways to find an approximate minimizer
- Convex optimization algorithms (more on that later)

If we introduce the **logistic loss** function

$$\ell(y, y') = \log(1 + e^{-yy'})$$

then

$$(\hat{w}, \hat{b}) \in \operatorname*{argmin}_{w \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle x_i, w\rangle + b)$$

A goodness-of-fit

$$(\hat{w}, \hat{b}) \in \underset{w \in \mathbb{R}^d, b \in \mathbb{R}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle x_i, w \rangle + b)$$

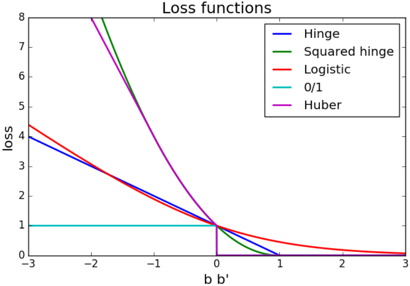
is natural: it is an **average of losses**, one for each sample point

Note that

- $\ell(y, y') = \log(1 + e^{-yy'})$ for logistic regression
- $\ell(y, y') = \frac{1}{2}(y - y')^2$ for least-squares linear regression

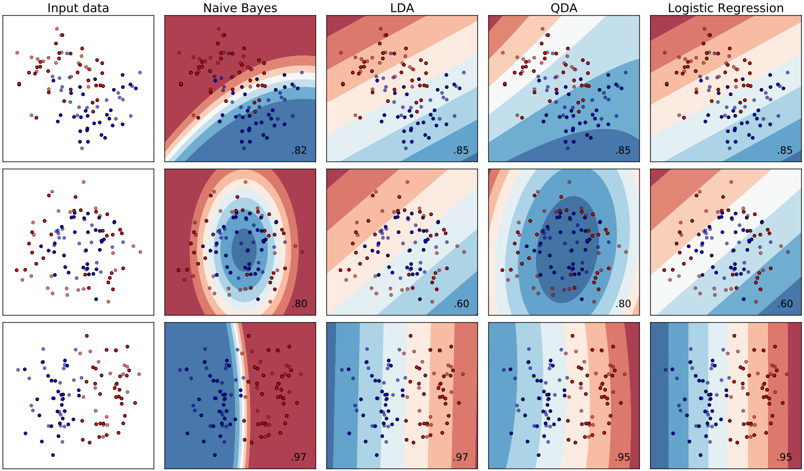
Other classical loss functions for binary classication

- Hinge loss (SVM), $\ell(y, y') = (1 - yy')_+$
- Quadratic hinge loss (SVM), $\ell(y, y') = \frac{1}{2}(1 - yy')_+^2$
- Huber loss $\ell(y, y') = -4yy'\mathbf{1}_{yy'<-1} + (1 - yy')_+^2 \mathbf{1}_{yy' \geq -1}$



- These losses can be understood as a convex approximation of the 0/1 loss $\ell(y, y') = \mathbf{1}_{yy' \leq 0}$

A comparison of classifiers on toy datasets



[the `jupyter notebook` for this figure will be on the webpage]

Standard error metrics in classification

- Precision, Recall, F-Score, AUC

For each sample i we have

- an actual label y_i
- a predicted label $\hat{y}_i$

We can construct the **confusion matrix**

		Predicted Class	
		Yes	No
Actual Class	Yes	TP	FN
	No	FP	TN

with

$$
\begin{aligned}
\text{TP} &= \sum_{i=1}^{n} \mathbf{1}_{y_i=1,\hat{y}_i=1} \\
\text{TN} &= \sum_{i=1}^{n} \mathbf{1}_{y_i=-1,\hat{y}_i=-1} \\
\text{FN} &= \sum_{i=1}^{n} \mathbf{1}_{y_i=1,\hat{y}_i=-1} \\
\text{FP} &= \sum_{i=1}^{n} \mathbf{1}_{y_i=-1,\hat{y}_i=1}
\end{aligned}
$$

with yes $= 1$ and no $= -1$

Standard error metrics in classification

$$\text{Precision} = \frac{\text{TP}}{\#(\text{predicted P})} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\#(\text{real P})} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

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

$$\text{F-Score} = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Some vocabulary

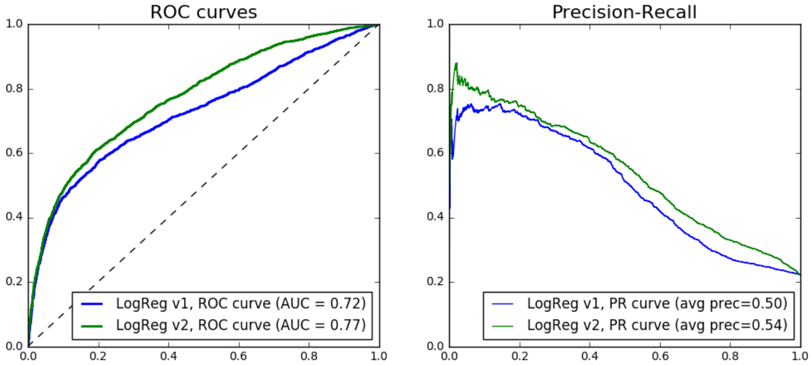
- Recall = Sensitivity
- False-Discovery Rate FDR $= 1 -$ Precision

ROC Curve (Receiver Operating Characteristic)

- Based on the estimated probabilities $\hat{p}_{i,1} = \hat{\mathbb{P}}(Y = 1|X = x_i)$
- Each point A_t of the curve has coordinates $(\mathrm{FPR}_t, \mathrm{TPR}_t)$, where FPR_t and TPR_t are FPR and TPR of the confusion matrix obtained by the classification rule

  $$\hat{y}_i = \begin{cases} 1 & \text{if } \hat{p}_{i,1} \geq t \\ -1 & \text{otherwise} \end{cases}$$

  for a threshold t varying in $[0, 1]$
- AUC score is the Area Under the ROC Curve


ROC curves
LogReg v1, ROC curve (AUC = 0.72)
LogReg v2, ROC curve (AUC = 0.77)
Precision-Recall
LogReg v1, PR curve (avg prec=0.50)
LogReg v2, PR curve (avg prec=0.54)
0.0
0.2
0.4
0.6
0.8
1.0

Penalization to avoid overfitting

Computing

$$\hat{w}, \hat{b} \in \operatorname*{argmin}_{w \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle x_i, w \rangle + b)$$

generally leads to a bad classifier. Minimize instead

$$\hat{w}, \hat{b} \in \operatorname*{argmin}_{w \in \mathbb{R}^d, b \in \mathbb{R}} \Big\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle x_i, w \rangle + b) + \frac{1}{C} \operatorname{pen}(w) \Big\}$$

where

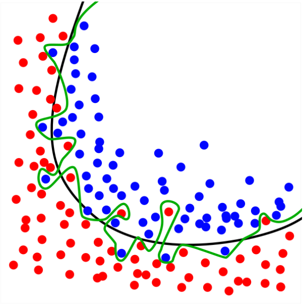
- pen is a **penalization** function, it forbids w to be "too complex"
- $C > 0$ is a **tuning** or **smoothing** parameter, that **balances** goodness-of-fit and penalization

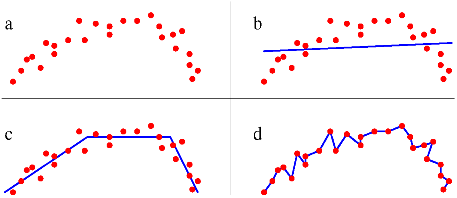
Penalization to avoid overfitting

In the problem

$$\hat{w}, \hat{b} \in \underset{w \in \mathbb{R}^d, b \in \mathbb{R}}{\text{argmin}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle x_i, w \rangle + b) + \frac{1}{C} \text{pen}(w) \right\},$$

a well-chosen $C > 0$, allows to avoid **overfitting**





Overfitting is what you want to avoid

Which penalization? The **ridge** penalization considers

$$\text{pen}(w) = \frac{1}{2}\|w\|_2^2 = \frac{1}{2}\sum_{j=1}^{d} w_j^2$$

It penalizes the "size" of w

In the case of the SVM (hinge loss) it has a nice interpretation: **corresponds to the margin** (more on that later)

This is the most widely used penalization

- It's nice and easy
- It allows to "deal" with correlated features (more on that later)
- It actually helps training! With a ridge penalization, the optimization problem is easier (more on that later)

There is another desirable property on $\hat{w}$

If $\hat{w}_j = 0$, then feature j has no impact on the prediction:

$$\hat{y} = \text{sign}(\langle x, \hat{w} \rangle + \hat{b})$$

If we have many features (d is large), it would be nice if $\hat{w}$ contained **zeros**, and many of them

- Means that only **few** features are statistically relevant.
- Means that only **few** features are useful to predict the label

Leads to a simpler model, with a "reduced" dimension

How to do it ?

Tempting to use

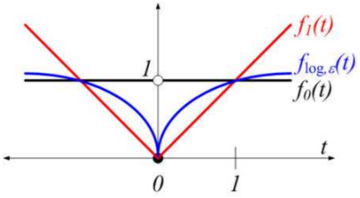
$$\hat{w}, \hat{b} \in \operatorname*{argmin}_{w \in \mathbb{R}^d, b \in \mathbb{R}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle x_i, w \rangle + b) + \frac{1}{C} \|w\|_0 \right\},$$

where

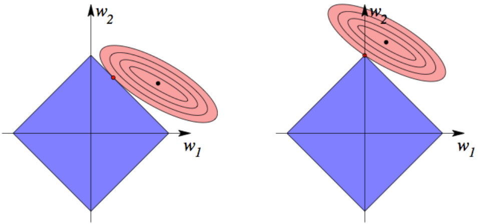
$$\|w\|_0 = \#\{j \in \{1, \dots, d\} : w_j \neq 0\}.$$

To solve this, explore **all** possible supports of w. Too long! (NP-hard)

Find a convex proxy of $\|\cdot\|_0$: the ℓ_1-**norm** $\|w\|_1 = \sum_{j=1}^d |w_j|$



Why does it induce sparsity?



Why ℓ_2 (ridge) does not induce sparsity?

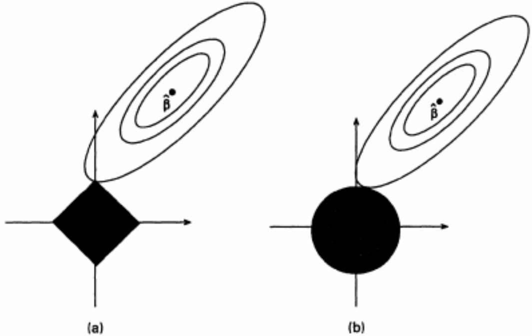


Fig. 2. Estimation picture for (a) the lasso and (b) ridge regression

A direct computation

Consider the minimization problem

$$\min_{z' \in \mathbb{R}} \frac{1}{2}(z' - z)^2 + \lambda |z'|$$

for $\lambda > 0$ and $z \in \mathbb{R}$

- Derivative at 0_+: $d_+ = \lambda - z$
- Derivative at 0_-: $d_- = -\lambda - z$

Let z_* be the solution

- $z_* = 0$ iff $d_+ \geq 0$ and $d_- \leq 0$, namely $|z| \leq \lambda$
- $z_* \geq 0$ iff $d_+ \leq 0$, namely $z \geq \lambda$ and $z_* = z - \lambda$
- $z_* \leq 0$ iff $d_- \geq 0$, namely $z \leq -\lambda$ and $z_* = z + \lambda$

Hence

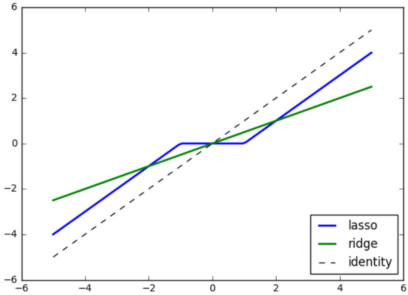
$$z_* = \operatorname{sign}(z)(|z| - \lambda)_+.$$

$$\operatorname*{argmin}_{z' \in \mathbb{R}} \frac{1}{2}(z' - z)^2 + \frac{1}{C}|z'| = \operatorname{sign}(z)\left(|z| - \frac{1}{C}\right)_+$$

so that

$$\operatorname*{argmin}_{w' \in \mathbb{R}^d} \frac{1}{2}\|w' - w\|_2^2 + \frac{1}{C}\|w'\|_1 = \operatorname{sign}(w) \odot \left(|w| - \frac{1}{C}\right)_+ .$$

Example with $C = 1$



Particular instances of problem

$$\hat{w}, \hat{b} \in \underset{w \in \mathbb{R}^d, b \in \mathbb{R}}{\mathrm{argmin}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle x_i, w \rangle + b) + \frac{1}{C} \mathrm{pen}(w) \right\},$$

For $\ell(y, y') = \frac{1}{2}(y - y')^2$ and $\mathrm{pen}(w) = \frac{1}{2}\|w\|_2^2$, the problem is called **ridge regression**

For $\ell(y, y') = \frac{1}{2}(y - y')^2$ and $\mathrm{pen}(w) = \|w\|_1$, the problem is called **Lasso** (Least absolute shrinkage and selection operator)

For $\ell(y, y') = \log(1 + e^{-yu'})$ and $\mathrm{pen}(w) = \|w\|_1$, the problem is called ℓ_1-**penalized logistic regression**

Many combinations possible...

The combinations

$$\text{(linear regression or logistic)} + \text{(ridge or } \ell_1\text{)}$$

are the most wildely used

Another penalization is

$$\text{pen}(w) = \frac{1}{2}\|w\|_2^2 + \alpha\|w\|_1$$

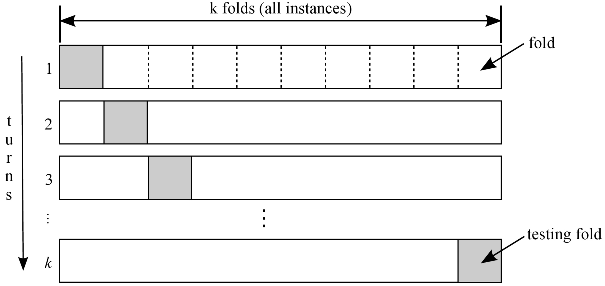
called **elastic-net**, benefits from both the advantages of ridge and ℓ_1 penalization (where $\alpha \geq 0$ balances the two)

Cross-validation

- **Generalization** is the goal of supervised learning
- A trained classifier has to be **generalizable**. It must be able to work on other data than the training dataset
- Generalizable means "works without **overfitting**"
- This can be achieved using **cross-validation**
- There is **no machine learning without cross-validation** at some point!
- In the case of penalization, we need to choose a penalization parameter C that generalizes

V-Fold cross-validation

- Most standard cross-validation technique
- Take $V = 5$ or $V = 10$. Pick a random partition $I_1, \ldots, I_V$ of $\{1, \ldots, n\}$, where $|I_v| \approx \frac{n}{V}$ for any $v = 1, \ldots, V$



Consider a set

$$\mathcal{C} = \{C_1, \dots C_K\}$$

of possible values for C. For each $v = 1, \dots, V$

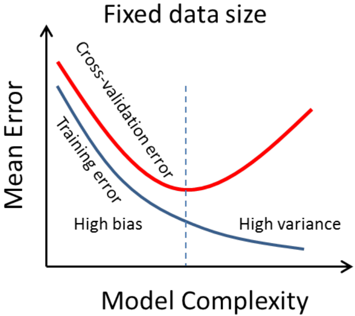
- Put $I_{v,\mathtt{train}} = \cup_{v' \neq v} I_{v'}$ and $I_{v,\mathtt{test}} = I_v$
- For each $C \in \mathcal{C}$, find

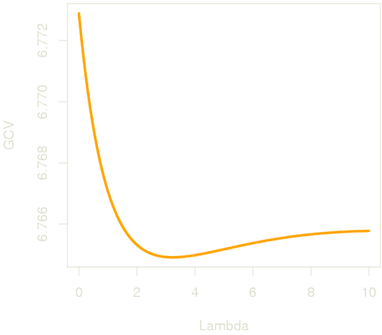
$$\hat{w}_{v,C} \in \operatorname*{argmin}_{w} \Big\{ \frac{1}{|I_{v,\mathtt{train}}|} \sum_{i \in I_{v,\mathtt{train}}} \ell(y_i, \langle x_i, w \rangle) + \frac{1}{C} \operatorname{pen}(w) \Big\}$$

Take

$$\hat{C} \in \operatorname*{argmin}_{C \in \mathcal{C}} \sum_{v=1}^{V} \sum_{i \in I_{v,\mathtt{test}}} \ell(y_i, \langle x_i, \hat{w}_{v,C} \rangle)$$

Remark: depending on the problem, we might use a different loss (or score) for choosing $\hat{C}$





- Training error:

$$C \mapsto \sum_{v=1}^{V} \sum_{i \in I_{v,\mathtt{train}}} \ell(y_i, \langle x_i, \hat{w}_{v,C} \rangle)$$

- Testing, validation or cross-validation error:

$$C \mapsto \sum_{v=1}^{V} \sum_{i \in I_{v,\mathtt{test}}} \ell(y_i, \langle x_i, \hat{w}_{v,C} \rangle)$$

Next week

- The linear SVM: the hinge loss
- Kernels methods
- And some jokes too...

Thank you!