ESPCI PARIS

EDUCATION SCIENCE INNOVATION

# Introduction to Big Data

## Statistical & Machine Learning

*Julien Barrier — class of 2018*

# *Introduction*

I have started writing this handout from the notes I have taken from Olivier Rivoire's course on Big Data and Statistical learning at ESPCI Paris from March to April 2018. Please be considerate if some mistakes crop up in this work.

*Julien*

Some book reading is advised during the course, partialicularly:

- *The Elements of Statistical Learning*, T. Hastie, R. Tibshirani and J. Friedman, Springer Series in Statistics, 2008;
- *Information Theory, Inference, and Learning Algorithms*, D.J.C. MacKay, Cambridge University Press, 2003.

**Dr Olivier Rivoire**
Center for Interdisciplinary Research in Biology (CIRB)
Collège de France
olivier.rivoire@college-de-france.fr

## *Applications*

There are plenty of applications for Big Data problems. A few examples may be given:

*Post* learn + identify digits on enveloppes

*Biology* DNA sequencing

*IT* Face recognition

*etc.*

Big Data is an issue of growing importance. As engineers, we may be familiar with such concepts.

## *Idea of marchine learning*

The main idea of machine learning is to find models to give prediction of input data. In facts, Big Data models are deduced from a

training batch of N input-output data, on which programs train to generalise models. The deduced model *input* $i \rightarrow$ *output* $i$ can then be generalised to give prediction from a random input, as long as it relates to the training batch.

Analytically, let's start with a collection of $x$ and $y$ data, where $x$ stands for the input data and $y$ is the vector of the output data. Each sample is going to have multiple dimensions, therefore we may use an algebraic model. Let $N$ be the number of samples used and p the dimension of each $x$ data. We may write x as an $N, p$ matrix and $y$ as a vector of $p$ dimensions.

We now N samples of p dimensions $x_{ij}$ associated with the N output data $y_i$.

From now on there are two possible cases: $y_i$ can be known or unknown. In the fist case ($y_i$ known), the problem is said to be *supervised*. Hence we may work with a finite discrete set of data: $y_i = 1, \cdots, K$. This problem is called categorical, and we can solve it with *classification*. We may also work with an infinite set of numbers: $y_i \in \mathbb{R}$. This problem is called quantitative, and we can solve it with *regression*.

The second case ($y_i$ unknown) is said to be *unsupervised* and can be solved via *clustering* or *dimension reduction* methods.

## *Deep learning*

In the past few years, there have been huge progress in the *deep learning* approach. It is based on so-called neural networks, that are models inspired by the brain operation.

People are trying to understand how to train these networks. It has had remarkable outcomes in image recognition, social network filtering, medical diagnoses, etc.

Deep learning is based on hidden layers, placed inbetween input and output layers, that are trained to find correlations and mathematical models.

The goal of this course is to explain whate these objects are, how do they work, and put it in relation with state of the art research.

What kind of open problems are there? How do neural networks operate? What are their unsuperised learning behaviour?

# *Contents*

*Least square regression, from small to big data* 7

*General principles* 13

*Supervised learning: classificaton, regression, nearest neighbours* 25

*Unsupervised learning: dimensionality reduction, PCA, SVD* 33

*Unsupervised learning: clustering, K-means, hierarchical* 35

*Neural networks, from single neuron to multilayer networks* 37

*Physics of machine learning, statistical mechanics of machine learning, applications* 39

# *Least square regression, from small to big data*

## *Linear Regression at One Dimension*

Let us work at one dimension: $p = 1$. If we work with N points, then $i = 1, \cdots, N$, and we work with a set of data $(x_i, y_i)$.

The goal here is to make a prediction of what the $y$ data should be when $x$ is given.

The simplest possible model is the linear regression given by the equation 1.

$$y = \alpha + \beta x. \tag{1}$$

Here, the main issue is to get the best $\alpha$ and $\beta$ for a partialicular set of data. To know what the best choice is, we may define a cost function, that returns a number representing how well the regression permorms. In neural network problems, the cost fuction return number is associated with how well the neural network performs in mapping training examples to the correct output.

There are several choices that can be made to define the cost function. At one dimension, the simplest choice is the sum of squared residuals, defined in equation 2, usually shortened as SSR.

$$l(\alpha, \beta) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \alpha - \beta x_i)^2 \tag{2}$$

Figure 1 illustrate a simple geometrical interpretation of what the SSR is. Actually, the lower $\epsilon_i$, the better the fit.

**TODO!**
Figure 1: geometrical intepretation of the SSR, where $\epsilon_i$ is given by the relation: $\epsilon_i^2 = (y_i - \alpha - \beta x_i)^2$

For all we have done up to now, we never have never worked with big data. We need $p$ large enough to consider this as a real big data issue.

If we're looking at a hundreds or thousands pixels picture, $p$ will be large in comparison with $N$. That is a full statistics problem.

Currently, $p = 1$ is small data, but all we did there has been a correct introduction to clearly understand big data problems.

Striking a good fit necessitates finding the best $\alpha$ and the best $\beta$. For this, we may look at the optimum, defined as the points where the derivative of $l$ versus $\alpha$ and $\beta$ vanishes. This is given by equations 3 and 4.

$$\frac{\partial l}{\partial \alpha} = -\frac{1}{N}\sum_{i=1}^{N}(y_i - \alpha - \beta x_i) = 0 \tag{3}$$

$$\frac{\partial l}{\partial \beta} = -\frac{1}{N}\sum_{i=1}^{N} x_i(y_i - \alpha - \beta x_i) = 0 \tag{4}$$

To solve this set of equations, we require a substitution for $x$ and $y$.

Let's define the mean values:

We must keep in mind that $\overline{x^2} \neq \bar{x}^2$.

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{5}$$

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{6}$$

$$\overline{xy} = \frac{1}{N}\sum_{i=1}^{N} x_i y_i \tag{7}$$

$$\overline{x^2} = \frac{1}{N}\sum_{i=1}^{N} x_i^2 \tag{8}$$

Thus, equations 3 and 4 can be reduced as:

$$\frac{\partial l}{\partial \alpha} = -(\bar{y} - \alpha - \beta\bar{x}) \tag{9}$$

$$\frac{\partial l}{\partial \beta} = -(\overline{xy} - \alpha\bar{x} - \beta\overline{x^2}) \tag{10}$$

This yields to:

$$\alpha = \bar{y} + \beta\bar{x} \tag{11}$$

$$\overline{xy} = -\bar{y}\bar{x} - \beta\bar{x}^2 - \beta\overline{x^2} \tag{12}$$

We use the given notations:

$$\begin{aligned} \mathrm{Cov}(x,y) &= \bar{xy} - \bar{x}\bar{y} \\ &= \overline{(x-\bar{x})}(y-\bar{y}) \\ \mathrm{Var}(x) &= \mathrm{Cov}(x,x) \\ \sigma(x) &= \sqrt{\mathrm{Var}(x)} \end{aligned}$$

There we may substitute $\alpha$ and $\beta$:

$$\hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}$$

$$\begin{aligned} \hat{\beta} &= \frac{\overline{xy} - \bar{x}\bar{y}}{\overline{x^2} - \bar{x}^2} \\ &= \frac{\mathrm{Cov}(x,y)}{\mathrm{Cov}(x,x)} \\ &= \frac{\mathrm{Cov}(x,y)}{\mathrm{Var}(x)} \end{aligned}$$

Let us define the Pearson coefficient $\mathcal{R}$ by the relation 13.

$$\mathcal{R} = \frac{\mathrm{Cov}(x,y)}{\sigma(x)\sigma(y)} \tag{13}$$

The Pearson coefficient $\mathcal{R}$ is always comprised between 0 and 1. Thus, we can define the quantity $\mathcal{R}^2$, that relates to the quality of the fit:

$$\mathcal{R}^2 = 1 - \frac{\hat{l}}{\mathrm{Var}(y)} \tag{14}$$

In the general case, we look at models where $\beta = 0$. In this case, the cost function $l$ would be the sum of the square distance to the line. Figure 2 depicts two linear regressions with different parameters. The right figure shows a much better linear regression, with much lower square distances between the points and the line.

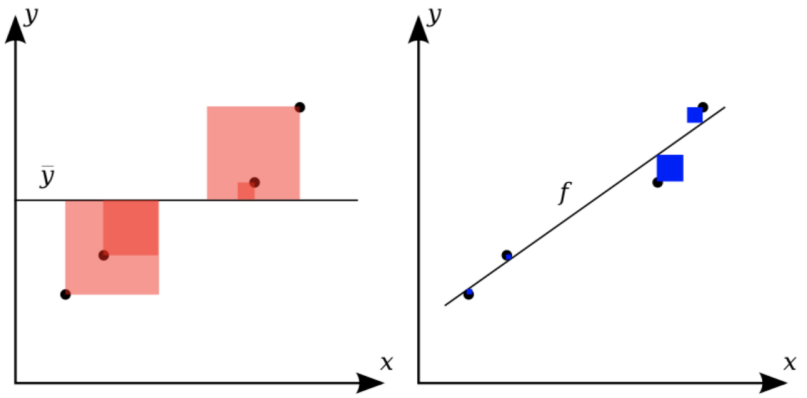



Figure 2: Two linear regression taken from two datasets. The left one shows higher square distances than the right one.

We may note that in some cases, it would be better to rescale the axis to fit the data with a linear regression. Logarithm axis is the most popular way of rescaling an axis to have a correct assumption. It is usually more valuable to rescale the axis and perform a linear regression, than trying to find an higher order fit.

### *Linear Regression at Higher Dimensions*

We are now considering higher dimensions data ($p > 1$), that are full, meaning that $p << N$. It means that, for an example of data, we might add different parameters. If we take the example (given in class) of correlations between the velocity of people versus the size of towns, we might add other relevant parameters, like the average heigh of people, their ages, etc. We might then examine many potential predictors. Thus we need to generalise the same things, where each input now becomes a vector of p dimensions, as presented in equation 15

$$(x_{i,1}, \cdots, x_{i,p}), \qquad \forall i \tag{15}$$

Let now $x_{i,j}$ be the matrix of the input data, where $i$ is the number of samples, varying from 1 to $N$, and $j$ is the dimension, ranged between 1 to $p$.

We may generalise the relation $\hat{y}_i = \hat{\alpha} + \hat{\beta} x_i$ in the new 16 equation:

$$\hat{y}_i = \hat{\alpha} + \sum_{j=1}^{p} \hat{\beta}_j x_{ij} \tag{16}$$

If we have this key figure, we can always add 1 in the x vector, as the $p+1$ coordinate. We can thus assume that $\alpha$ vanishes. In fact, we

can always redefine the data so that $\alpha$ vanishes. We can also rescale the variable, by removing the mean:

$$x' = x - \bar{x} \tag{17}$$
$$y' = y - \bar{y} \tag{18}$$

Therefore, the output coordinate $\hat{y}_i$ can be written as the product $\hat{y}_i = X\hat{\beta}$, that is a much more convenient way to write it.

That are just restrictions of the problems that help us to compute it.

Let $l(\beta)$ be the cross-function, define with equation 19.

$$l(\beta) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 \tag{19}$$

Let $Z$ be a vector whose components $z_i$ are defined as follows:

$$\sum_{i=1}^{N} z_i^2 = ||Z||^2 = Z^T Z \tag{20}$$

Therefore we can write the cross-function as:

$$l(\beta) = \frac{1}{N}(Y - X\beta)^T(Y - X\beta) \tag{21}$$

This form can easily be differentiated with $\beta$, and the retrieved derivative vanishes at the extremum (eq. 22).

$$\frac{\partial l}{\partial \beta} = -\frac{Z}{N}X^T(Y - X\beta) = 0 \tag{22}$$

The equation 22 can be reduced as $X^T Y = X^T X\beta$, which can be solved by introducing the matrix $C = X^T X$ (eq. 23)[1]

$$\hat{\beta} = (X^T X)^{-1}X^T Y = C^{-1}X^T Y \tag{23}$$

At higher dimension, the geometry consists in fitting with an hyperplane, as shown on figure 3.

Here, we are essentially solving a system of equations. We must consider the number of variables adapted to the number of equations that we get. When there is not enough equations (when $p$ is too small for example), the system is undetermined. We cannot reduce it and do not have a single solution.

Actually, when $p > N$, we can solve this problem with the condition $\hat{l} = 0$. This is a situation where there are more parameters than there are equaltions. It is easy to solve. The solutions consists in overfitting.

For instance if we have 100 parameters and 10 equations, we can never manage to get any result. However, we can find easy solutions, but this will overfit. At this stage, the system cannot be inverted.

$^T$ denotes the transpose matrix, defined by the relation: $[\mathbf{A}^\mathbf{T}]_{ij} = [\mathbf{A}]_{ji}$

[1] Note that

$$C_{ij} = \sum_{k=1}^{N} x_{kj}x_{ki}$$

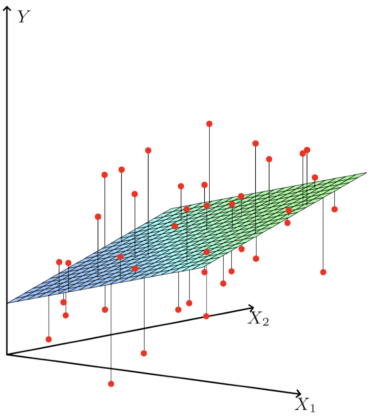



Figure 3: Linear least squares fitting with $X \in \mathbb{R}^2$. In this problem, we are looking for the linear function of X that minimises the sum of squared residuals from Y, which is an plane (hyperplane in dim 3)

When $p$ is large, even if it is of the same order of magnitude as $N$, we are working with big matrixes, that can be tricky to invert with both proper mathematical accuracy and computation performances. In such cases, we should handle the data carefuly.

Large $p$ are typical way of using statistical learning methods, by discriminating between datasets without having the aforesaid issues.

# *General principles*

- Models
- bias vs variance
- cross-validation
- maximum likelihood
- Bayes
- etc.

Generally, big data issues are composed of many parameters, with which it is very energy-consuming. We usually want to reduce the problems to one or two interesting parameters. Even if start with a lot of data, we would need to find what are those interesting parameters, and what are the important dimensions of the problem we will be working on.

There are two advantages at reducing the number of parameters:

accuracy With less parameters, we can estimate them with much more accuracy;

easiness it is always easier to estimate things on a condensed set of parameters.

Hence the issue is to find a compromise between having enough parameters to estimate properly the result of the problem, without loosing precision when having a too large batch.

Here $||\cdot||_0$ stands for the cardinal: $||\beta||_0 = \#(\beta_j \neq 0)$

Let $\tilde{l}(\beta)$ be defined as:

$$\tilde{l}(\beta) = l(\beta) + \lambda||\beta||_q \tag{24}$$

Now, the problem lays in finding

$$\min_{\beta} l(\beta) \text{ given } ||\beta||_0 \leq C \text{ with } \beta = [0, \ldots, \beta_i, 0, \ldots, 0] \tag{25}$$

It is not possible to provide this kind of problem with any good numerical solution. In fact, we would always try to find a compromise between what we are able to optimise efficiently and what is possible to optimise. We can write the problem in a new version, easier to solve:

$$\min_{\beta} l(\beta) \text{ given } ||\beta||_2 \leq C_1 \tag{26}$$

### *The two models*

There are two models that can be used to solve that kind of problem. The first is the *Ridge regression* that consists in using the constraints that can be solved efficiently numerically.

The second model is called the *Lasso regression* and consists in finding:

$$\min_{\beta} l(\beta) \text{ given } ||\beta||_1 \leq C_2 \tag{27}$$

In the ridge problem, we assume that the problem is sparse: we only need a few parameters to capture the relationship.

In fact, the problem can be reset by writing:

$$\tilde{l}(\beta) = (Y - X\beta)^T(Y - X\beta) + \lambda\beta^T\beta \tag{28}$$

$$\begin{aligned} \frac{\partial\tilde{l}(\beta)}{\partial\beta} &= 2(-X^T(Y - X\beta) + \lambda\beta) && (29)\\ &= 2(-X^TY + (X^TX + \lambda\mathbb{1})\beta && (30) \end{aligned}$$

$\mathbb{1}$ is the identity matrix.

We are here adding constraints to the problem, that will allow us to solve it numerically.

$$C_{jk}p > N \tag{31}$$

$$C_{jk} = \frac{1}{N}\sum_{i=1}^{N} x_{ij}x_{ik} \tag{32}$$

$$\bar{x} = 0; C = X^TX \tag{33}$$

$$X_{ij}Nxp \tag{34}$$

**TODO!**

$p > N => C$ is non invertible.

$N = 1, C_{jk} = x_{1j}x_{1k}$

$C = XX^T$

here, $C$ is of rank 1.

Mathematically, $rank(C) \leq N$.

remind that $Z^TZ = ||Z||^2$

$Z^Ty =< Z, y >= \vec{Z} \cdot \vec{y}$

This means that there are too many parameters for too few samples. If we try to solve this with a linear regression problem, it will give too many solutions.

Here it is easy to understand that $p > N$ cases give problems. However, because we are working with big data, the $p \sim N$ cases also gie rise to problems. We would need $p << N$ to have correct solutions.

## *Unsupervised learning*

### *Example of financial data*

Let's start with stock data, as presented on figure 4.

With these data, we want to extract information. However, y data are not labelled. For this reason, we may not use the raw data, but try to find something else that better fits with the problem.

Here, we want to get rid of the $\alpha$ parameter; for this reason, we use the $r_{ti}$ data instead of the $s_i(t)$ parameter.

Then we define $x_{ti}$, by substracting the mean and normalising with the standard deviation.

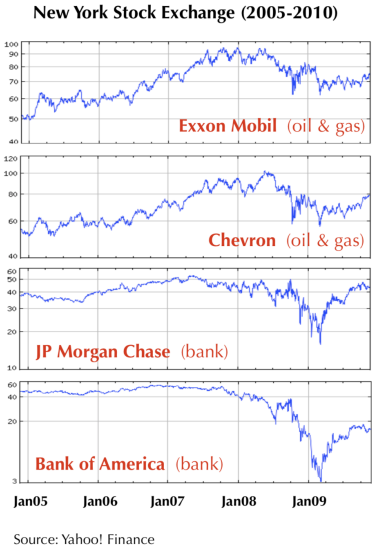
Therefore, the $x_{ti}$ value has a null mean, and a standard deviation rescaled to 1. Hence we have rescaled the problem so that each data vary within the same range.

Now, let's move to the data $C_{ij}$.

Before trying to make anycalculation, we can assume that Exxon and Chevron data are strongly correlated, as well as JP Morgan and Bank of America.

Thus, we may suppose that $C_{Ex,Ch} > C_{Ex,JP}$.

In order to analyse the data, we may compute the spectrum. Or see clearly from the definition that the matrix is symmetric, and has all the properties to be diagonalised.




Source: Yahoo! Finance

Figure 4: Stock data used as an example

NB: When we consider some data, it is usually very important to watch it before trying to compute anything else. Particularly, if something looks obvious to the eye, we need to set it clearly, before trying to interpret it with the math.

$$C_{jk}, C_{jj} = 1; C_{ij} = C_{ji} \tag{35}$$

Thus we get the eigenvalues: $\lambda_1, \ldots, \lambda_p$, and the eigenvectors $v_1, \ldots, v_p$. The dispersion of the eigenvalues is represented on figure 5.

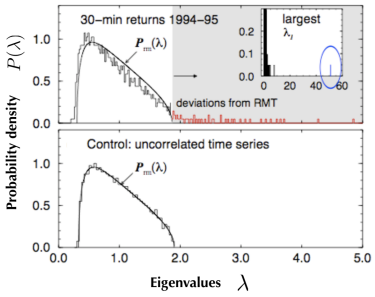



Figure 5: dispersion of the eigenvalues

The bottom part of the graph presented on figure 5 is called the control part. Actually, we see that this part comprises mostly noise, and the interesting part of the data is the upper part, that describes longer range correlations.

The control part just shuffles the data, and is due to permutation of the values . If we randomly shuffle the data, to remove all the interesting information (correlation between the different stocks), the control part stays the same. This is actually a way to see what kind of correlation we can get from just randomness, in a case where there is no correlation from the data. Thus, we can quantify the degree of noise. Because 98% of the eigenvalues are contained in the 1st part of the data (control), it means that the stock data are comprised of 98% of noise.

Let's write the matrix C as:

$$C = \sum_{j=1}^{p} \lambda_j v_j v_j^T \tag{36}$$

$$v_j^T v_k = \delta_j^k \tag{37}$$

Random metrics theory: branch of statistical physics, we can

compute analytically the shape of the control series. RMT.

$$\min_x l(x) \text{ given } g(x) \leq C \tag{38}$$

$$\min_x l(x) + \lambda g(x) \text{ with } \lambda \geq 0 \tag{39}$$

**TODO!**
Figure 6: scheme

I assume everything is differentiable.

$$\nabla_x l(x) = -\lambda \nabla_x g(x) \tag{40}$$

The conditions tells that the two gradients should be aligned, and directed in opposite directions. Generally, it would depend on the value C. Minimum value., can be a local minimum if the problem isn't perfectly shaped.

If I have $l(\beta)$, and am considering the norm of $\beta$ to be less that C value:

$$||\beta||_q \leq C_q \tag{41}$$

We can do it with $l_0$, $l_2$, sometimes also with the $l_1$ norm.

Why are we interested in that kind of things, what does it give us?

With a not too big dataset, taken from a book. The goal here is to try to predict the crime rate, and to what it is correlated.

$(x_{ij}, y_i)$ $i = 1..N = 50$ cities $j = 1 \dots 5$

The idea is to consider naively a simple problem.

Here we may find linear combination of all different problems. For a physicist, it seems we're not allowed to to so, because not homogeneous. But this helps finding correlations.

$x'_{ij} = x_{ij} - \bar{x_j}$. In this case, we have zero mean, We may also want try to divide by the standard deviation. Not the case here.

$$y = \sum_{j=1}^{p} \beta_j x_j \tag{42}$$

Therefore we can write $l(\beta) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \sum_{j=1}^{p} \beta_j x_{ij})^2$

The result of this optimismation may be given as a function.

Graph on the right, ridge regression. What is plotted is the value of the $\beta$ along the $x$ axis. This has to do with the cost ($c_q$). We can repeat for different values of the cost. If we do it for large $C$, we do not put any constraints, and therefore get the same $\beta$.

if we put a very strong cost, like o, the only solution is $\beta = 0$. Hence we're looking at different solutions, constrained, and then we relax it to a state where there's no constraint anymore.

The lasso graph is the same, but performed with $l_1$ norm.

There's a way to understand this, by giving an illustration.

Fig 2.2 slides.

$$y = F(x, \theta). \tag{43}$$

Linear models:

$$F(x, \theta) = \sum_{j=1}^{p} \theta_j x_j \tag{44}$$

The principle is to have the results of p data, and then once we get another dataset, similar to the previous one, we're able to fit it and to find the solution.

$$x_1, \cdots x_p, x_1^2, \cdots, x_p^2, \cdots cosx_1, \cdots \tag{45}$$

$$F(x, \theta) = \sum \theta_j h_j(x) \tag{46}$$

Later on, we'lll see neural networks. There, the function h people are generally using is: $h_j(x) = \frac{1}{1+\exp(-\omega_j^T x)}\omega_j$ is the weight. We'll see this later.

-> There's a relation between x and y. There's no limit to the complexity of the problem we can take.

Loss function $L(y, F(x, \theta)) = (y - F(x, \theta))^2$.

Training error: $err_{training}$ given a model and given a loss function:

$$err_{training} = \frac{1}{N} \sum_{i=1}^{N} L(y_i, F(x_i, \theta)) \tag{47}$$

This is not the only quantity we want to consider. test/generalisation error. This one would be the error we get when we are using these datapoint that have not been used in the training of the problem.

There's the training set, used to learn the parameters, and the additional data, on which we're going to apply the model, and to try to generalise the data that have been used as an input for the fit.

$$Err_{test} = \frac{1}{N'} \sum_{i=1}^{N'} L(y_i', F(x_i', \theta)) \tag{48}$$

Plot in slides: training vs test errors.

The objective is not to get the best fit, but to generalise the given data.

procedure: K-fold cross-validation. We would divide the data in 5 datasets, and then, take 1 out to use as the test, and all the other one as training test. And then we repeat for all the other combinations. Divide data in K=5 or 10, use most of the data to train, and use one set to test.

Data set is splitted in:

- training set (for the fit)
- test set (for model selection)
- Validation set (for assessment)

Typical number would be : 50%,25%,25%.

We use this to understand ho to find the best parameters.

$y = F(x)$

Training set: $\hat{\theta}$

Model $y = F(x, \theta)$

Thus we have $y = \hat{F}(x) = F(x, \theta)$

Another dataset:

$x_0$

$$\mathrm{E}[(F(x_0) - \hat{F}(x_0))^2] \tag{49}$$

Where $\hat{(}F)$ is the prediction.

This is considered over different training sets. It says how far I am from the value I want to get the prediction.

$$\mathrm{E}[] = F(x_0)^2 - 2F(x_0)\mathrm{E}(\hat{F}(x_0)) + \mathrm{E}(\hat{F}(x_0)^2) \tag{50}$$

Where $\mathrm{E}(\hat{F}(x_0)^2)$ is $\mathrm{Var}(\hat{F}(x_0) + (\mathrm{E}[\hat{F}(x_0)])^2$

i.e. $\mathrm{E}[] = (F(x_0) - \mathrm{E}[\hat{F}(x_0)]^2 + \mathrm{Var}(\hat{F}(x_0))$.

Test error = $(bias)^2$ + variance

We can generalise this when there is some noise $\epsilon$ (random variable) :

$$y = F(x) + \epsilon \tag{51}$$

There we will add another parameter : $var(y)$ that is irreducible error.

In the context of linear regression, there's the Gauss-Markov theorem. It tells us that in linear regression, all the estimators that have no bias, the best one is the one that is minimising the loss function

$$L(y, F(x)) = (y - F(x))^2 \tag{52}$$

This theorem is telling us that if we're interesting in minimising the bias in the context of linear regression we should take $min_\beta l(\beta)$

No bias + min var => $min_\beta l(\beta)$. In general, the best solution is not the solution that has no bias. I will estimate better the parameters that I have with a constrained set of data.

last time, we learnt unsupervised learning. $(x, y)_{i=1\cdots N}$

Goal : learn $x \to y$. function: theta so that when given new $x_i$

$x_i \to^{predict} \hat{y}_i \sim y_i$ function: $\hat{\theta}$

function with two components : $err_{test}$ wich is composed of the bias + variance. if large amount of data, high variance, very hard to learn, and we get very different parameters.

we may want to compromise this with a model with less parameters with a lower variance problem. not only we want to pick the best variable in the model, but also the model itself. we consider a class of problems, described with a parameter $\lambda$. we introduce $\lambda$ as a parameter of regularisation.

we are defining, for example, the error as:

$$l(\beta, \lambda) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \beta x_i)^2 - \lambda||\beta||_0$$

- training -> $\hat{\theta}$ given $\lambda$
- validation -> $\hat{\lambda}$
- test -> $Err_{test}$

At the begining, we divide the dataset into multiple datasetts.

K-fold cross validation : the idea is that we have one dataset, that we divide into K subsets. We can get it with statistics. two weeks ago, we've seen an example of this, in the context of linear regression. minimising the mean square error

planning of the day:

- Bayesian approach
- how we do the approximation in practice.
- example of CLASSIFICATION.

First, let's discuss about maximum likelihood estimation. This is not a big-data specific approach. general in statistics. In general, I would have a model of the form $y = f(x, \theta) + \epsilon$ where $\epsilon$ stands for the noise. it is a random variable. We suppose $\epsilon$ is a normal variable: $\epsilon(0, \sigma^2)$

The probability to see x with theta:

$$P(y|x, \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp - \frac{(y - f(x, \theta))^2}{2\sigma^2} \tag{53}$$

**TODO!**
Figure 7: P(y | x,theta) representation. with the standard deviation $\sigma^2$ and centered on f(x,theta)

Now, let's imagine we are given a set of values $(x_i, y_i)$. We want to find the best parameter $\hat{\theta}$. We look at the parameter that make the data the most likeable.

$$\begin{aligned}
\mathcal{L}(\theta|Z_1, \cdots, Z_n) &= \log P(Z^N|\theta) && (54)\\
&= \sum_{i=1}^{N} \log P(Z_i|\theta) && (55)\\
&= \sum_{i=1}^{N} \left[ -\frac{1}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} (y_i - f(x_i, \theta))^2 \right] && (56)\\
&= -\frac{N}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1} N(y_i - f(x_i, \theta))^2 && (57)\\
&= -\frac{N}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} l(\theta) && (58)
\end{aligned}$$

MSE

MLE: $max_\theta \mathcal{L}(\theta, Z^N) \rightarrow \hat{\theta}$ theorem: if $y = f(x, \theta_0) + \epsilon$

$$\mathrm{E}[\hat{\theta}] \rightarrow_{N \rightarrow \infty} \theta_0$$

$$\hat{\theta}(\theta_0, F(\theta_0)^{-1})$$

$F(\theta) = \mathrm{E}\mathcal{L}(\theta)$ this is called the Fisher information.

$I(\theta)_{ij} = -\frac{\partial^2 \mathcal{L}}{\partial\theta_j \partial\theta_k}$

Therefore,

$\hat{\theta} \sim (approx) N(\hat{\theta}, I(\hat{\theta})^{-1})$

**TODO!** figure parabolle inversée, maximum : courbature $-\partial\mathcal{L}/\partial\theta^2$, absisse max : $\theta_0$. ordonnée : L.

We want to find the best parameter, *i.e.* the one that is the more likelihood to be ...

L is called the log-likelihood. P is colled the likelihood. In a sense, we want to find the most likely model.

We can mention, that there are two difficulties :

- we need to find the maximum
- problem of validation: if we have a more complicated model, we would increase the log likelihood, and no way to do the validation.

Now, another approach: the Bayesian approach. Again, this is not specific to big-data. We may know that there are lot of debates in the meaning of probabilities. There are two schools: the frequentists: probabilities have a meaning only when the event is reapeating many times. Like a coin tail or head. Fundamentally, if I do it a large number of time, this is converging. If we take an event that can happen only once: the probability that there's lifeon the moon: for a frequentist, there's no meaning. the Bayesian view is different in nature: probability is not about counting, but about biliefs. This represent how I believe the event to be actually the case. Concretely, the idea of the bayesian approach is: elementary probability: when I have two random variables X, Y. The joint probability of (X,Y): P(X,Y). we can also define the marginals : $P(X)$, $P(Y)$, that are only the probability $P(X) = \sum_y P(X, Y)$. $P(X|Y)$: conditional probability. $P(X, Y) = P(X|Y)P(Y)$ therefore $P(X|Y) = \frac{P(X,Y)}{P(Y)}$.

$P(Z, \theta) = \frac{P(Z,\theta)}{P(Z)} = \frac{P(Z|\theta)P(\theta)}{P(Z)}$. This is called the Bayes formula.

$P(Z = 1) = \theta$ $P(Z = 0) = 1 - \theta$, for example, for a binomial problem.

$P(Z|\theta)$ is the likelihood I had before. In this approach, there's something new, that is $P(\theta)$ wich is called the prior. For a bayesian, we always have some *a priori* bieliefs about the distribution of the parameters. When I see the data, I have to update my beliefs. $P(\theta|Z)$ is called the posterior. $P(Z)$ is called the evidence.

Here we have to do the inference, that is the general model. Usually, when we look at probabilites, we look at $P(Z|\theta)$. reverse approach. We look at the model, that also incorporates the prior.

On the slides, one partialicular example of the prior to solve a praticular problem.

Here, there's nothing to do with big data. example taken from the book of MacKay example: Jo has a test for a disease. a = state of health. a=1 if sick, 0 otherwise. the test is giving another variable b what is known about this test is that it can be positive even if there's no disease. $P(b = 1|a = 1) = 95\%$. same for zero. it means the test gives the right result in 95% of the cases. we need to know the case when somebody of HIS AGE has the disease. this is gonna be the prior. $P(a = 1) = 1\%$. The exercise is to find what is the probability $P(a = 1|b = 1)$. In the bayesian approach, we do not have a theta, but a distribution of the theta we always have a probability to have different values, especially the maximum *a posteriori* estimate (MAP) which is given by taking the max of this: $\max_\theta P(\theta|Z)$.

It will give the same results if we are assuming that this is not depending on theta. If we have a flat prior. Then, this is equivalent to the MLE.

When I'm maximising the posterior, it means I'm maximising the likelihood. Thus we can see the likelihood as a bayesian approach, with a partialicular prior value.

Let's say we have the same model as before. This time, we as-

sume the prior is a one dimensional variable, with a gaussian distribution.

$f(\theta) = \sqrt{\frac{\lambda}{2\pi}} e^{-\lambda \frac{\theta^2}{2}}$

Then, this will be very similar to the l validation, if we take the log of this. Then, this is multiplied by lambda theta square over two.

When we take a prior on these parameters, we want to give more probability to the small values of the parameters. width of the gaussian: 1/sqrt(lambda). control the probability of the parameter to have a large value. restricting the range of the parameter that we are considering, as we saw before

The goal here is to present this partialicular approach, and recover the maximum likelihood. An interesting aspect of the bayesian approach. Again, this is very general. Let's say that, in general, we have the probability of the data, given some hypotheses:

$P(D|H_1)$ Dis the data, $H_1$ thehypothesis.

we want to compare with another hypothesis: $P(D|H_2)$

What the bayesian approach is telling us is that we have to consider the probability : $\frac{P(H_1|D)}{P(H_2|D)} = \frac{P(D|H_1)P(H_1)}{P(D|H_2)P(H_2)}$

it depends on the prior we put there. we do not here have no belief????

One example of the maximum likelihood and the general approach.

Problem of generalization. Let's say we have the data: $x_{i=1...N}$. two approaches are possible: MLE that gives $\hat{\theta}$ bayesian that gives $P(\theta|X)$.

what is the probability of a partialicular value?

$P(x_{N+1})$ =(MLE) $P(x_{N+1}|\hat{\theta})$. if we need a best value of approximation, let's replace the parameter.

= (bayesian)$P(x_{N+1}|X^N) = \sum_\theta P(x_{N+1}|\theta P(\theta|x^N)$

here, $P(\theta|x^N)$ should be used as the new prior. We are constantely updating our believes. this is the prior we get before knowing the value, depending on what we saw before. This has to be equal to: $\frac{P(x^N|\theta)P(\theta)}{P(x^N)}$

There's a famous problem that has to do with that: the rule of succession. The sun is rising every morning, what is the probability it will rise tomorrow? Laplace discussed this issue.

With different approaches, we may get different results. One of the simplest example. Let's assume there's some probability $\theta$ the sun is rising in the morning. We hae a binomial model. This is the same issue as a coin that always ends up in the same edge: tail for example. Maximum likelihood estimation: the probability would be 1.

$x_i = 0/1$.

$x^N = (x_1, \ldots, x_N) = (1, \ldots, 1)$

$P(x_{N+1} = 1)$ =? If we do the maximum likelihood approximation, this would be 1 everytime.

$X^N$ : $N_1 times 1; N - N_1 times 0$. (times : frequency it happens over

time)

$P(x_i|theta) = \theta$. One parameter model, binomial model. Obviously, the probability $P(x_i = 0|\theta) = 1 - \theta$. this example cannot be simpler than this.

theta can be anything between 0 and 1. $0 \leq \theta \leq 1$.

I'm giving the same probability for every $\theta$. $P(\theta) \sim$uniform.

$P(x_{N+1} = 1|x^N) = \int d\theta P(x_{N+1} = 1|\theta)P(\theta|x^N) = \int d\theta\theta \frac{P(x^N|theta)P(\theta)}{P(x^N)} where P(\theta) = 1 = \frac{N!}{N_1!(N-N_1)!}\theta^{N_1}(1-\theta)^{N-N_1}$

The difficulty lies in the $P(x^N)$ that does not depend on theta.

$P(\theta|x^N) = C(N, N_1)\theta^{N_1}(1-\theta)^{N-N_1}$

$\int d\theta P(\theta|x^N) = 1$

$\int_0^1 d\theta\theta^a(1-\theta)^b = \frac{a!b!}{(a+b+1)!}$ this exists also for a and b that are not integer values, and this is called the gamma function. If we use this formula, we would find this to be:

$c(N, N_1) = \frac{(N+1)!}{N_1!(N-N_1)!}$ with the proper normalisation. I need just one line to compute the stuff. If I compute this:

$P(x_{N+1} = 1|x^N) = \int_0^1 d\theta\theta \frac{(N+1)!}{N_1!(N-N_1)!}\theta^{N_1}(1-\theta)^{N-N_1} = \frac{(N+1)!}{N_1!(N-N_1)!}\frac{(N_1+1)!(N-N_1)!}{(N+2)!} = \frac{N_1+1}{N+2} \neq \frac{N_1}{N}(MLE)$

this is called as the laplacian formula.

we have a non-zero probability to observe something that we have never observed before.

$N = 3.X^N = (1,1,1).canwebetthatx_4 = 1$? maybe it is not very wise to say this. this rule takes this into account. here, the probability to be 0 will be $1/5$, not zero.

people that carry out statistics use pseudo-counts. we are adding one zero and one one. It is a way to regularise the variation. In this case, a frequentist would say that we have too few points and that we must give up the problem. for a bayesian, the calculation would be very dependant on the prior. Prediction on the next outcomes.

commentary: $I(\theta) = -\frac{\partial^2\mathcal{L}}{\partial\theta_i\partial\theta_j}$

$\mathcal{F}(\theta) = \mathrm{E}[I(\theta)]$ MLE: $\hat{\theta}(\ldots)$ For each dataset, we can use another expectation. It is mathematical consideration, considering all the possibilities of my data.If we arepartialificially generating data when we want to prove all mathematical results analytically, we need all the possible datasets that we can get. variance about everything we can get when generating different datasets.

second hour.

Now I want to discuss the computational issues.

$\hat{\theta}$ that I want to maximise. log likelihood: $\mathcal{L}(\theta|x^N)$. we would have, in general, to get these data numerically, and not analytically, with optimised function. compromises to be done.

Very simple, but the problem can be complicated if the function has several minimum. Let's assume the problem is convex: the function is convex, as well as the set of data. Any global minimum is a global minimum. It can be generate, but,...

In all these problems, we can consider a gradient descent.

If we want to minimise the function,

scheme fig 2. If we look at the gradient, and spartial from a point

(random). we look at the grandient, and move in the direction of it. we usually take a very small displacement. spartial iteratively.

$\theta_{t+1} = \theta_t - \eta \left(\frac{\partial L}{\partial \theta}\right)_{\theta_t}$

scheme: cf notes.

cf learning rate, etc.

$L(\theta) = \sum_{i=1}^{N}(y_i - f(x_i, \theta))^2 = \sum L_i(\theta)$. for each calculation, we have to recompute the data. a version of this algorithm is used very often, and is called stochastic gradient. What we do is: compute $L_i(\theta)$ for a subset of the points. So we take a subset of samples to estimate $L(\theta) = \sum_{i \in subset}$ and we change at each iteration. The sample is called mini-batch.

There's one version of this algorithm, where everytime we take a single value as a subset: subset =1. this is called on-line learning. in this case, what we are doing is:

$\theta_{t+1} = \theta_t - \eta \nabla_\theta L_i(\theta_t)$.

The issue is to get faster. In general, we have to take the sum. At the end, it is equivalent to do a move at everytime than getting the sum from the very beginning. Actually ,this algorithm is very powerful. In neural networks, backpropagation is nothing more than this algorithm put in application.

This is really a local method. If I spartial from one point, I can end somewhere different. I can be trapped in local minima. It is really difficult to find the correct minimum. That's why we usually work with convex functions, where local minima do not exist elswhere than the global minimum.

$l_0$ norm, $l_1$ norm. $||\beta||_0 = \#nonzero\beta_j; ||\beta||_1 = \sum_j |\beta|_j$

For this reason the closest problem to the first is with $l_1$, and it is convex so that it can be solved with an iterative method. Generally, these are considerations we want to take into account. We will see examples of doing this next time.

Next week: exercise as homework. practical. that will be the grade.

$min_\beta L(\beta) -> \hat{\beta}$

Lasso regression. this is one in which we are going to impose a condition: $||\beta|| \leq t$ this is equivalent to the fact that we want to minimize : $min_\beta L(\beta) + \lambda||\beta||withsomeparameter\lambda$.

ridge regression: $||\beta|| = ||\beta||_2^2 = \sum \beta_i^2$

if we take the $l_1$ norm:

lasso regression: $||\beta|| = ||beta||_1 = \sum |\beta_i|$

At this stage, we can take it as an exercise.

Le'ts spartial with the function I want to minimize. $\mathcal{L}(\beta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \beta x_i)^2 + \lambda|\beta|$

differentiation: we must be careful.

$\frac{\partial \mathcal{L}}{\partial \beta} = 0$

the maximum, if $\beta$ is positive ,let's say, we can get the value: $(\beta > 0)$

$\frac{\partial \mathcal{L}}{\partial \beta} = -\frac{2}{N}\sum_{i=1}^{N}(y_i - \beta x_i)x_i + \lambda$.

I would take $\bar{x} = 0, \bar{y} = 0$ and $\overline{x^2} = 1$. normalise all the data.

In general, everything can have completely different units. It makes sense here to normalise, so that each data has the same range of variation. So

$dL/dbeta = -2(\frac{X^T y}{N} - \beta) + \lambda = 0$

thus $\hat{\beta} = \frac{X^T y}{N} - \frac{\lambda}{2}$

If $\frac{X^T y}{N} > \frac{\lambda}{2}$, thus $\hat{\beta} = \frac{X^T y}{N} - \frac{\lambda}{2} > 0$

if $\beta < 0$, then we can use the same approach.

$\hat{\beta} = S_{\lambda/2}\left(\frac{X^T y}{N}\right)$

$S_a(u) = \text{sign}(u)\max(O, |u| - a)$. soft-thresholding operator. cf figure notes.

This is a figure for p=1.

cf on the website a slide with the formulas.

for $p > 1$,

$L(\beta) = \frac{1}{N}\sum_i (y_i - \sum_k \beta_k x_{ik})^2 + \lambda \sum_k |\beta_k|$

$= \frac{1}{N}\sum_i (y_i - \beta_j x_{ij} - \sum_{k\neq j} \beta_k x_{ik})^2 + \lambda|\beta_j| + \lambda \sum_{k\neq j} |\beta_k|$

let's define $r_{ij} = y_j - \sum_{k\neq j} \beta_k x_{ik}$ $\hat{\beta}_j < -S_{lambda/2}(\frac{1}{N} x_j^T r_j)$ cyclical coordinate descent. we do this for j, and then repeat for j + 1 until convergenc. we get an interative algorithm. $\beta_1$, then $\beta_2, \ldots$

Because the truc is convex, this is going to converge to the minimum of the function $L(\beta)$.

cf note on this algorithm. If we understand the case for p=1, then we repeat, and because the problem is convex, we're going to converge to the single minimum.

On wednesday, we will see single classification.

# *Supervised learning: classificaton, regression, nearest neighbours*

regression = supervised learning for quantitative data.

$x -> y$

if we want to classify pictures between cats and dogs, two kinds of methods: - supervised: give information before - unsupervised: you figure out that there are two categories, understand that there are two categories, and learn from that.

fig 5

distincition on the nature of the variable that we want to predict. quantitative data : why is it a real number or a vector?

we are going to see today classification. so far we've seen just supervised quantitative : regression. there are more task to do.

today : classification. next course : clustering. after that : neural networks, that can be used for any of those tasks, that involve a bit more methodology.

CLASSIFICATION.

training data: $(x_i, k_i)$

label : $k_i = 1, 2, \dots K$.

Again, the challenge is using large datasets. we use big data. $x_{N+1} \rightarrow^{?} k_{N+1}$

so far we've used only regression. if we start with only two categories : $K = 2$. $k_i \in \{0, 1\}$

from these data, I can try to predict the value of K in unsupervised learning. the closest to 0 and 1, and then get a classification.

let's start with the case where $p = 1$. fig 6

we have to find a value somewhere to split the value between two sets of data. generally, we have more complicated problems, in larger dimensions. here, if we do a linear regression:

$y = \hat{\alpha} + \hat{\beta}x$. this is what we did in the first lecture: compute $\hat{\alpha}$ and $\hat{\beta}$ that gave us the best result.

decision boundary: $if \hat{y} is larger than \frac{1}{2}, then \hat{k} = 1$.

$\hat{y} < \frac{1}{2} => \hat{k} = 0$

if we start again with p=2. it can be in two dimensions.

fig 7.

data linearly separable: we can put an hyperplan that separates the data. define a decision boundary. there we can define an error.

NB: in advance, we cannot preduct wich method would work, before looking at the data. In practical, no method would separate

perfeclty the data. if we augment the dimension of the space, we can consider other functions : for example quadratic instead of linear. tradeoff : if we add more parameters, sure we can reduce the training error, but also, in general, increase the variance. back to the problem already discussed: how to find the correct level of complexity. surely it can fit the data, but may not be generalised.

data still have to be linearly separable. again we'll se a second class of methods. ability to solve different issues. fig 8 for example (marginfigure)

now, let's add another data. if $y = 0, 1, 2$, there is no practical meaning. no classical order. here it is classification. it works well with two categories. if we have three categories, then we can split between two categories. data: 1 VS 1. based on the boundaries, I will decide which boundary I can consider.

This was a naive approach. If I start again with p=1: fig 9.

we added data far enough to put the blue squares in another category.

linear regression:

$$\min_{\beta} \frac{1}{N} \sum_{i} (y_i - \beta x_i)^2 \tag{59}$$

it is more convenient to work with more symmetric data. the problem with saying that the error for i is:

$$error_i = (y_i - \beta x_i)^2. \tag{60}$$

$K = 2$: $k_i = \pm 1 bd$ if $\hat{\beta} x > 0 => +1$ $if \hat{\beta} x < 0 => -1$

what matters is wether the quantity $\hat{y} = \hat{\beta} x$ has the correct sign or not.

$$L(\beta) = \frac{1}{N} \sum_{i} \max(0, -y_i \beta^T x_i) \tag{61}$$

here again, the idea is that when $\beta x$ is of the same sign as $y$, the maximum is zero, and the error is zero.

this is called hinge regression.

the idea is gradient descent

smooth version -> logistic regression.

the idea is to replace this versiona fig 10.

$$L(\beta) = \frac{1}{N} \sum_{i=1}^{N} \ln(1 + e^{-y_i \beta^T x_i}) \tag{62}$$

he tries to minimise the error, and the split value with that logistic regression.

we can also arrive to similar results by chosing different approaches. this is connected to the kind of operations that are done in neural networks. we will see this later on.

instead of taking the reflection y to be k, it's easier to consider y to be a probability. it makes here much more sense to make a linear regression. a probabilistic model, in which I want to describe the probability, given some data. parameter $\omega$: weight, standard

notation in neural networks, whereas $\beta$ is much more used in classical regression/statistical learning models.

$$P(k=1|x,\omega) \tag{63}$$

$$P(k=1|x,\omega) \quad > P(k=-1|x,\omega) \quad => \hat{k}=1 \tag{64}$$

$$< \quad => \hat{k}=-1 \tag{65}$$

if we are considering the log of the proabilities:

$$\ln \frac{P(k=1|x,\omega)}{P(k=-1|x,\omega)} = \omega^T x \tag{66}$$

$$= \sum_{j=1}^{P} \omega_j x_j (+\omega_0) \tag{67}$$

I don't have to always write $+\omega_0$, for convenience, just changes in the problem that have no influence.

In general, I want to define a more relevant model.

Let's call:

$$y = P(k=1|x,\omega) \tag{68}$$

$$\ln(\frac{y}{1-y}) = \omega^T x \tag{69}$$

$$\frac{y}{1-y} = \exp(\omega^T x) \tag{70}$$

thus

$$y = e^{\omega^T x}(1-y) \tag{71}$$

therefore

$$y(1+e^{\omega^T x}) = e^{\omega^T x} \tag{72}$$

$$y = \frac{e^{\omega^T x}}{1+e^{\omega^T x}} = \frac{1}{1+e^{-\omega^T x}} \tag{73}$$

$$P(k|x,\omega) = \frac{1}{1+e^{-k\omega^T x}} \tag{74}$$

$$\mathcal{L}(\omega) = \frac{1}{N}\sum_i \ln P(k_i|x_i,\omega) MLE \tag{75}$$

if i replace P by its form:

$$\mathcal{L}(\omega) = -\frac{1}{N}\sum_i \ln(1+e^{-k_i\omega^T x_i}) \tag{76}$$

Essentially, the idea was to apply the maximum likelihood. I had to make hypothesis on the general form of the model. Once I do this, I end up with this function. The different with the one above: minus sign. not surprising: a likelihood we want to maximise, and not error we want to minimize.

$$\max_\omega \mathcal{L}(\omega) <=> \min_\beta L(\beta) \tag{77}$$

in neural networks,
fig11

$$u = \sum_{j=1}^{p} \omega_j x_j = \omega^T x \tag{78}$$

is called the activation.

several inputs, and one output based on this. if $u > 0$, then I have output $y = 1$. if $u < 0$, then $y = 0$.

in the context of neural networks, people prefer work with 0 and 1 rather than $\pm 1$.

neuron is a classifier
fig11

$$y = \phi(u) \tag{79}$$

$$\phi(u) = \frac{1}{1 + e^{-u}} \tag{80}$$

essentially, what we do, is considering the different inputs with different weights. At the end, we will end with more complex classifier models. this one is very elementary. there are very simple algorithms, like the gradient descent.

let's start again with the log likelihood:

$$\begin{aligned} \mathcal{L}(\omega) &= \frac{1}{N} \sum_i (k_i \ln y_i + (1 - k_i) \ln(1 - y_i)) && (81) \\ &= \frac{1}{N} \sum_i \ln P(k_i | x_i, \omega) && (82) \end{aligned}$$

$$y_i = P(k_i = 1 | x, \omega) \tag{83}$$

$$\begin{aligned} ln(P(\ldots)) &= y_i \quad \text{if } k_i = 1 && (84) \\ &= 1 - y_i \quad \text{if } k_i = 0 && (85) \end{aligned}$$

then we make the model by assuming:

$$y_i = \phi(u_i) where u_i = \omega^T x_i = \sum_{j=1} \omega_j x_{ij} \tag{86}$$

now, we want to compute the gradient. it means to compute the derivative of the log likelihood:

$$\frac{\partial \mathcal{L}}{\partial \omega_j} = \frac{1}{N} \sum_i (k_i \frac{1}{y_i} \frac{\partial y_i}{\partial \omega_j} - (1 - k_i) \frac{1}{1 - y_i}) \frac{\partial y_i}{\partial \omega_j} \tag{87}$$

with

$$(\ldots) = \frac{k_i(1 - y_i) - (1 - k_i) y_i}{y_i(1 - y_i)} = \frac{k_i - y_i}{y_i(1 - y_i)} \tag{88}$$

and

$$\frac{\partial y_i}{\partial \omega_j} = \phi'(u_i) \frac{\partial u_i}{\partial \omega_j} = \phi'(u_i) X_{ij} \tag{89}$$

thus

$$\frac{\partial \mathcal{L}}{\partial \omega_j} = \frac{1}{N} \sum_i \frac{\phi'(u_i)}{\phi(u_i)(1-\phi(u_i))}(k_i - y_i)x_{ij} \tag{90}$$

$$\phi(u) = \frac{1}{1+e^{-u}} => \frac{\phi''}{\phi(1-\phi)} = 1 \tag{91}$$

$$\frac{\partial \mathcal{L}}{\partial \omega_j} = \frac{1}{N} \sum_i (k_i - y_i)x_{ij} \tag{92}$$

for neural networks, the same idea as gradinet descent is called backpropagation , but it is a bit more complicated.

fig 2.1: representation of classification.

support vector machine : SVM problem formally attitred:

$$\min_{\omega} \sum_i \max(0, 1 - k_i \omega^T x_i) + \lambda ||\omega||^2 \tag{93}$$

the exercise is trying to find what this function is doing.

standard linear regression: take care about the distance to the line. when doing SVM: introducing some margin of errors. this approach would count as an error even if we are on the right side of the line but close to it. we're not going to explain it here, but it looks very much linke the other frameworks.

another approach: linear discriminant analysis (LDA)

we take a model (probabilistic), more gaussian like. the objective function is to fit two classes by the gaussian, in such a way we maximise the distance between the gaussians, and minimise the variance of the gaussians. that can easily be done with linear algebra.

there are many other methods on the same lines.

Now we want to be able to adress the problem of non linear separability. where the real boundary would be a circle. in and out.

fig12

we can also do this locally.

it is called k-nearest neighbours

again, let's start with a batch of values: fig13

we are given with a particular point, and say if it is red or blue. what are the neighbours doing?

if $k = 3$, i am looking at the three nearest points, and then take the colour of the majority.

mathematically, it can be written as:

$$\hat{y}(x) = \frac{1}{k} \sum_{i \in N_k(x)} y_i \tag{94}$$

$N_k(x)$ is the set of $k$ nearest neighbours to $x$

$y_i \in \{0, 1\}$

Again, we can be in a situation described by fig 14. we have, there, a decision boundary. i'm deciding what is the coulour of the nearest points with that boundary.

fig 2.2 and 2.3. variations of K

now the question is: what should we take for K

$\frac{N}{k} \sim p$ : number of degrees of freedom.

fig 15

figure slide: 'selection of K': using 10,000 datapoints and tested with 200.

minimal training error if we take k=1. very complicated thing. as we are taking larger and larger values of k, we are averaging and get more errors, even in the training set. if we have too few degrees of freedom, we are completely missing the structure of data. it is also not good if we have too many degrees of freedom.

15 is relatively well located in this area. between 20 and 10, there's not that much difference, maybe 10 is better because it uses less resources. we have to pick the right model, and there are models of different levels of complexity. different values of k that can be used.

there is nothing into this algorithm, but also nothing trivial in finding the right datapoints.

there are many variations along this method. in this example, we are taking the same density of points, everywhere. but if I take any point, it doesn't mean anything to take just three points randomly. we may take into account the distance between the points. essentially, we would average around the point that are close to the considered point. more generally, we can do it with a hard cutoff close/far or smooth cutoffs.

$$\hat{y}(x) = \frac{\sum_i K_\lambda(x, x_i) y_i}{\sum_i k_\lambda(x, x_i)} \tag{95}$$

$$K_\lambda(x, x_i) = \frac{1}{\lambda} e^{-\frac{||x - x'||^2}{2\lambda}} \tag{96}$$

or $K_\lambda(x, x_i) = 1$ if $||x - x'|| < \lambda$, 0 otherwise.

kernel methods also posisble. if we do this, it is also a case where we have one parameter: $\lambda$. It plays the same role. big lambda: averaging a lot. we would reduce the variance, but with a too large bias. in the opposite: reduce the bias would end in a too large variance.

again, there are variety of methods that can be used. one way to see this problem of classification is to fit a function. let's say, zero for orange points, 1 for blue points.

At one dimension, we would have fig 16.

maybe there could be errorsa

the thing is that we may smooth all those functinos. we may think about these problems as fitting problems. correcting locally the errors of the functions.

kernel method smoothing, to locally have something smoother. what these things are doing is similar to the problem of smoothly inferring a function from just any perfect observation.

This is very simple, but the problem is that none of those things are working for big data. issue of classifying dogs and cats.

If we go in any of those methods, there is no way we can succeed. Several problems, the greatest is a geometrical one: the curse of

dimensionality.

essentially, it has to do with all the assumption we have about neighbourhoods are false when we are working with high dimensional spaces.

example in 2D in fig 17.

volume is scaling as power p of the radius. $V = \alpha_p r^p$.

If we look at $\frac{V}{V_{tot}} = \left(\frac{r}{r_{tot}}\right)^p$

then we have

$$\frac{r}{r_{tot}} = \left(\frac{V}{V_{tot}}\right)^{1/p} \tag{97}$$

If the ratio V/Vtot = 0.01. if we take something like p=10, we'll see that r/rtot = 0.6.

if we want to cover even a very small fraction of the volume, we have to go very far from the point. ridiculous amount of the volume. the problem of scaling with the radius is not interesting. in high dimensional spaces, all the volume is close to the centre.

to cover a small fraction of the volume, we have to go very far.

All the ideas of closest neighhbourgs cannot be used in high dimensionality spaces.

we can also reduce the dimensionality of the data. that is the first approach with this problem. solutions:

- reduce dimensionality -> PCA (unsupervised)
- representation, that has to do with what kind of space we are woking with. we have to imagine that we have all the pictures of cats and dogs, all put in the same place. something invariant with translation, rotation, etc. so this will be seen in another lecture, about neural networks. essentially, one class of netwkrs: convolutional, is based on this. chose the right space in which it is easy to chose the classification.

exam In priciple, everything should be comprised in the lecture. code it ourself. not using things already written. the game here is to write ourself our own libraries. it is not very complicated, each question is just a few lines of code.

with python: using the jupyter. matlab: not sure it is doeable. livescripts. write a report in which he can see the code. in a format he can actually work the code. minimal thing: write the code and display the figures. this is a public problem, neither on the solutions presented on the internet is good, we might think about it from ourselves.

*Unsupervised learning: dimensionality reduction, PCA, SVD*

*Unsupervised learning: clustering, K-means, hierarchical*

*Neural networks, from single neuron to multilayer networks*

*Physics of machine learning, statistical mechanics of machine learning, applications*