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 May to April 2018. Please be considerate if some mistakes crop up in this work.

Julien

Some book reading is advised during the course, particularly:

- *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.

Analyticall, 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 neurol 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

Linear Regression

Least square regression at one dimension Let $p = 1$. 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 α and β for a particular 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 ϵ_i, the better the fit.

TODO!
Figure 1: geometrical intepretation of the SSR, where ϵ_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 composed of hundreds, 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 α and the best β. For this, we may look at the optimum, defined as the points where the derivative of l versus α and β vanishes. This is given by equations 3 and 4.

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

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

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

Let's define[1]:

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

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \quad (5)$$

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i \quad (6)$$

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

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

Thus, equations 3 and 4 can be reduced as:

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

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

This yields to:

$$\alpha = \bar{y} + \beta\bar{x} \quad (11)$$

$$\overline{xy} = -\bar{y}\bar{x} - \beta\bar{x}^2 - \beta\overline{x^2} \quad (12)$$

There we get $\hat{\beta} = \frac{\bar{xy} - \bar{x}\bar{y}}{\bar{x^2} - \bar{x}^2} = \frac{cov(x,y)}{cov(x,x)} = \frac{cov(x,y)}{var(x)}$

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

NB: $cov(x,y) = \bar{xy} - \bar{x}\bar{y} =$

$var(x) = com(x,x)$

$std(x) = \sqrt{var(x)}$

Pearson coefficient $\mathcal{R}$ is defined by: $\mathcal{R} = \frac{cov(x,y)}{std(x)std(y)}$

$\mathcal{R}^2 = 1 - \frac{\hat{l}}{var(y)}$

This quantity relates to the quality of the fit. $\mathcal{R}$ is comprised between 0 and 1. 1 corresponds to a good fit, 0, to a very bad one.

Illustration: M H Bornstein, Nature 1976. in general, we're looking at models where $\beta = 0$. In this case, the cost function l would be the sum of the square distance to the line. In the second case, much smaller l.

Data: speed at which people are working in different cities. They estimated the speed, and made a regression versus the population size of the city. The claim of the paper is to say there's a pretty good correlation of the speed people are walking and the size of the town. This is not a linear fit, because logarithmic scale in the

population. In practice, we plot x versus y, it is usually better to transform x by log(x) to find the best linear regression. Formally, it is the same formula.

We're now moving to p>1, and full data, that means that p«N. If we consider the previous example of the linear regression, let's consider other relevant parameters, for example the average heigh of people, sex, etc., let's examine many potential predictors. We need to generalise the same thing, where each input is a vector of dimension p:

$x_1, \ldots, x_p$.

Now we have $x_{i,j}$ where I is the sample, from 1 to N, and j is the dimension, from 1 to p.

There the generalisation $\hat{y}_i = \hat{\alpha} + \hat{\beta} x_i$ is now:

$$\hat{y}_i = \hat{\alpha} + \sum_{j=1}^{p} \hat{\beta}_j x_{ij} \tag{13}$$

If we have this key figure, we can always add 1 in the x vector, as the p+1 coordinate. Thus we can assume that α vanishes. In other terms, we can always redefine the data so that α vanishes. We can also rescale the variable, by removing the mean:

$$x' = x - \bar{x} \tag{14}$$
$$y' = y - \bar{y} \tag{15}$$

Therefore, let $\hat{y}_i$ be written as $\hat{y}_i = X\hat{\beta}$, 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.

$$l(\beta) = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \tag{16}$$

Let Z be a vector of which the components z_i are defined as: $\sum_{i=1}^{N} z_i^2 = ||Z||^2 = Z^T Z$.

Therefore we can write the cross-function as:

$$l(\beta) = \frac{1}{N} (Y - X\beta)^T (Y - X\beta). \tag{17}$$

The advantage of this form is that is easily differentiates with β:

$\frac{\partial l}{\partial \beta} = -\frac{Z}{N} X^T (Y - X\beta) = 0$ to get the extremum.

NB: T c'est la notation américaine, on met le signe transposée après la transposée, et non pas avant comme en français. Donc $X^T Xestlatransposede Xfois X$.

If we need to solve $X^T Y = X^T X\beta$ (the previous equation), we get:

$$\hat{\beta} = (X^T X)^{-1} X^T Y = C^{-1} X^T Y \tag{18}$$

with $C = X^T X$

$$C_{jk} = \sum_{i=1}^{N} x_{ij} x_{ik} \tag{19}$$

exercise, try to come with p=1 and...

All the geometry consists in fitting with an hyperplane. We can try to draw a figure of an hyperplane as an example in the handout. essentially, we are solving a system of equations. We have to consider the number of variables adapted to the number of equation that we get. If we have not enough equations, for example p too small, the system is underrmined. We cannot reduce it and do not have a single solution. Actually, when $p > N$, we can solve this problem with $\hat{l} = 0$. This is a situation where there are more parameters than there are equaltions. It's 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. We could find easy solutions, but this will overfit. (like trivial salutions). At this stage, the system cannot be inverted.

Even if p is comparable to N, it will still be true that we should be careful: there are several reasons. We have very big matrixes, inverted it could be tricky. It will take a lot of time, we could have problems of mathematical accuracy. Actually, it doesn't make sense to have too much parameters in the problem.

This is where people use big data. We have these problems in many situations. If we want to discriminate between different datasets., p can be very large with big data problems.

So, what do we do now? In general, we want to know what are the most interesting parameters. At the end, we can predict the issue with one or two parameters. Even if we start with a lot of data, we want to find what are the parameterts, the dimensions important in the problem we are working on.

Two advantages:

Interpretation With less parameters, we can estimate them with much more accuracy than if we have more. In general, it's easier and more accurate to estimate things on a condensed set of parameters than on a large set. The issue is: how do you compromise: having enough parameters to have a good enough estimation of the problem, without having too much and loosing precision. Enough information VS enough precision.

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

$$\tilde{l}(\beta) = l(\beta) + \lambda ||\beta||_q \tag{20}$$

With $||\beta||_0 = (\beta_j \neq 0)$ (cardinal)

We look for $min_\beta l(\beta)$ given $||\beta||_0 \leq C$

With $\beta = [0, \cdots, \beta_i, 0, \cdots, 0]$ There ain't any good numerical solution.

There's always a compromise between what we are able to optimise efficiently, and what is possible to optimise. A version of the problem that is easy to solve is:

$$min_\beta l(\beta) given ||\beta||_2 \leq C_1 \tag{21}$$

Ridge regression: there's a way to get to this problem, using the constraints that can be solved efficiently numerically.

Or $\min_\beta l(\beta)$, given $||\beta||_1 \leq C_2$. This is called the Lasso regression.

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

Why did we start to write it this way?

$$\tilde{l}(\beta) = (Y - X\beta)^T(Y - X\beta) + \lambda\beta^T\beta \tag{22}$$

$$\frac{\partial\tilde{l}(\beta)}{\partial\beta} = 2(-X^T(Y - X\beta) + \lambda\beta) = 2(-X^TY + (X^TX + \lambda\infty)\beta \tag{23}$$

∞ is the identity matrix.

What is doing is putting constraints. You add constraints and then you can solve mathematically the problem.

On wenesday, we will compare those two regression, and define the framework for machine learning.

$$C_{jk} p > N \tag{24}$$

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

$$\bar{x} = 0; C = X^TX \tag{25}$$

$$X_{ij} N x p \tag{26}$$

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

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

$C = XX^T$

here, C is of rank 1.

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

$$Z^Ty =< Z, y >= \vec{Z} \cdot \vec{y} \tag{27}$$

Mathematically, $rank(C) \leq N$. I have too many parametetrs, not enough samples. When I try to solve this linear regression problem, I have too many solutions.

There are issues when p>N, but also when they are of the same order of magnitude.

Example: financial data.

We want to get information from this data, but there's no label, no y data. In general, we don't take the raw data, but try to find

something more adapted to the problem. Here, we want to get rid of the α parameter; for this reason, we use the r_{ti} data instead of the $s_i(t)$ parameter.

Then we define the 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. Therefore each data vary within the same range.

If we move to the data C_{ij}

When we have some data, an important step is to watch it by eye, and try to find correlations. If something is obvious to the eye, we'll try to interpret it with the math. Example: ExxonChevron are strongly correlated, JP Morgan and Bank of America are also strongly correlated. 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.

$$C_{jk}, C_{jj} = 1; C_{ij} = C_{ji} \tag{28}$$

Thus we get $\lambda_1, \dots, \lambda_p$ eigenvalues, and $v_1, \dots, v_p$ eigenvectors.

TODO!
Figure 2: dispersion of the eigenvalues

There's no way that I can have a good estimation of these metrics.

$Bottom = control$. They just shuffle the data, make permutation of the values. It is the same stocks. Randomy shuffle the data, to remove all the interesting information (correlation between the different stocks).

It's a way to see what kind of correlation we can get just from randomness, in a case where there's no corraltion from the data.

We can quantify the quantity of noise.

98% of the eigenvalues are contained in the 1st part of the data. That is 98% of noise.

Therefore I write:

$$C = \sum_{j=1}^{p} {}_j v_j v_j^T \tag{29}$$

$$v_j^T v_k = \delta_j^k \tag{30}$$

Random metrics theory: branch of statustical physics, we can compute analytically the shape of the control series. RMT.

$min_x l(x)$ given $g(x) \leq C$

$min_x l(x) + \lambda g(x)$; $\lambda \geq 0$.

TODO!
Figure 3: scheme

I assume everything is differentiable.

$$ {}_x l(x) = -\lambda_x g(x) \tag{31}$$

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 β to be less that C value:

$$||\beta||_q \leq C_q \tag{32}$$

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 $j = 1...5$

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.

$$y = \sum_{j=1}^{p} \beta_j x_j \tag{33}$$

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 optimisation may be given as a function.

Graph on the right, ridge regression. What is plotted is the value of the β 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 β.

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.

$$y = F(x, \theta). \tag{34}$$

Linear models:

$$F(x, \theta) = \sum_{j=1}^{p} \theta_j x_j \tag{35}$$

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.

$$x_1, \cdots x_p, x_1^2, \cdots, x_p^2, \cdots cos x_1, \cdots \tag{36}$$

$$F(x, \theta) = \sum \theta_j h_j(x) \tag{37}$$

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:

$$err_{training} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, F(x_i, \theta)) \tag{38}$$

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.

$$Err_{test} = \frac{1}{N'}\sum_{i=1}^{N'} L(y_i', F(x_i', \theta)) \tag{39}$$

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

$$E[(F(x_0) - \hat{F}(x_0))^2] \tag{40}$$

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.

$$\mathscr{E}[] = F(x_0)^2 - 2F(x_0)E(\hat{F}(x_0)) + E(\hat{F}(x_0)^2) \tag{41}$$

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

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

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

We can generalise this when there is some noise ϵ (random variable) :

$$y = F(x) + \epsilon \tag{42}$$

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

$$L(y, F(x)) = (y - F(x))^2 \tag{43}$$

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.

General principles

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

Supervised learning: classificaton, regression, nearest neighbours

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