Learning and data modelling

MA13-Probability and statistics: Courses 15-16

September 2014

Manuel SAMUELIDES[1] Zhigang SU[2]
[1]Professor
Institut Supereur de l'Aeronautique et de l'Espace

[2]Professor
Sino-European Institute of Aviation Engineering
Civil Aviation University of China

Example of a problem

Problem

Let us consider some input-output device. The model of the device is not available. Then we implement a set of experiments and we get a set of n **input-ouput measures** (x_i, y_i).

- From these measures we want to have a good approximation of the device model $y = h(x)$.
- Generally we have to choose h among a parameterized family $x \in \mathcal{R}^p \rightarrow h(x, w)\ with\ w \in \mathcal{W}$.
- The model is intended to be predictive outside the sample.
- The input x may be controlled or given by probability law of density f

Singular value decomposition (SVD) / 奇异值分解

Definition

Formally, the singular value decomposition of an $m \times n$ real or complex matrix $\mathbf{M}$ is a factorization of the form

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H,$$

where

- $\mathbf{U}$ is an $m \times m$ real or complex unitary matrix,
- $\mathbf{\Sigma}$ is an $m \times n$ rectangular diagonal matrix with non-negative real numbers on the diagonal,
- and $\mathbf{V}^H$ (the conjugate transpose of $\mathbf{V}$, or simply the transpose of $\mathbf{V}$ if $\mathbf{V}$ is real) is an $n \times n$ real or complex unitary matrix.

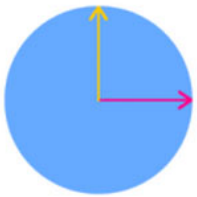
The singular value decomposition of an matrix $\mathbf{M}$ is a factorization of the form

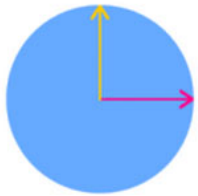
$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H,$$

- The diagonal entries $\Sigma_{i,i}$ of $\mathbf{\Sigma}$ are known as the singular values of $\mathbf{M}$,
- the m columns of $\mathbf{U}$ are called the left-singular vectors of $\mathbf{M}$,
- and the n columns of $\mathbf{V}$ are called the right-singular vectors of $\mathbf{M}$.

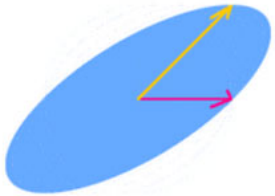
Proof of SVD theorem

- Unicity of $\mathbf{\Sigma}$ We get $\mathbf{M}^H\mathbf{M} = \mathbf{V}\mathbf{\Sigma}^H\mathbf{\Sigma}\mathbf{V}^H$. From that it appears that Σ is built from the square root of the spectral decomposition of $\mathbf{M}^H\mathbf{M}$.
- Building $\mathbf{V}$ So we can choose for the column vectors $\mathbf{v}_i$ of $\mathbf{V}$ an orthonormal basis of eigenvectors of $\mathbf{M}^H\mathbf{M}$.
- Building $\mathbf{U}$ So we can choose for the column vectors $\mathbf{u}_i$ of $\mathbf{U}$ an orthonormal basis of eigenvectors of $\mathbf{M}\mathbf{M}^H$.

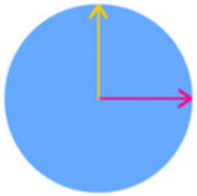




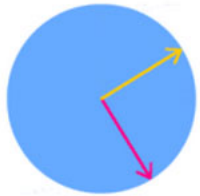
$$M = \begin{bmatrix} M_{1,1} & M_{1,2} \\ M_{2,1} & M_{2,2} \end{bmatrix}$$



$$M = \begin{bmatrix} M_{1,1} & M_{1,2} \\ M_{2,1} & M_{2,2} \end{bmatrix}$$



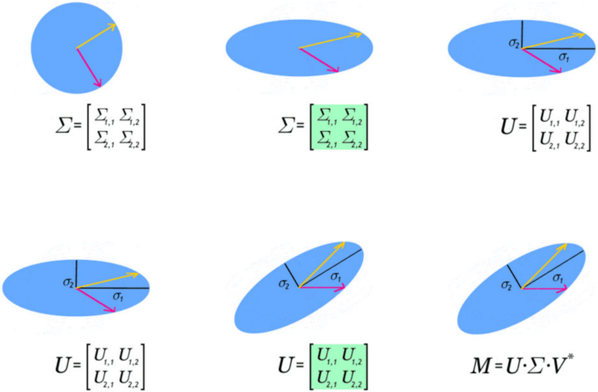
$$V^* = \begin{bmatrix} V^*_{1,1} & V^*_{1,2} \\ V^*_{2,1} & V^*_{2,2} \end{bmatrix}$$



$$V^* = \begin{bmatrix} V^*_{1,1} & V^*_{1,2} \\ V^*_{2,1} & V^*_{2,2} \end{bmatrix}$$

Manuel SAMUELIDES, Zhigang SU

中国民航大学



Relationship between SVD and EVD

The singular value decomposition and the eigendecomposition are closely related.

- The left-singular vectors of $\mathbf{M}$ are eigenvectors of $\mathbf{M}\mathbf{M}^H$,
- the right-singular vectors of $\mathbf{M}$ are eigenvectors of $\mathbf{M}^H\mathbf{M}$,
- and the non-zero singular values of $\mathbf{M}$ (found on the diagonal entries of $\mathbf{\Sigma}$) are the square roots of the non-zero eigenvalues of both $\mathbf{M}^H\mathbf{M}$ and $\mathbf{M}\mathbf{M}^H$.

Applications that employ the SVD include computing the pseudo-inverse, least squares fitting of data, matrix approximation, and determining the rank, range and null space of a matrix.

Moore-Penrose pseudo-inverse

Definition

- With the SVD previous notations $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H$, we define the $n \times m$ matrix $\mathbf{M}^+$ by

$$\mathbf{M}^+ = \mathbf{V}\mathbf{\Sigma}^+\mathbf{U}^H$$

  where
  - $\mathbf{\Sigma}^+$ is obtained from $\mathbf{\Sigma}^H$ (if rectangle) and then by replacing the strictly positive diagonal terms by their inverse.
- $\mathbf{M}^+ = \mathbf{V}\mathbf{\Sigma}^+\mathbf{U}^H$ is called the Moore-Penrose pseudo-inverse of $\mathbf{M}$.
- We have $Ker(\mathbf{M}^+) = Im(\mathbf{M})^\perp$ and $Im(\mathbf{M}^+) = Ker(\mathbf{M})^\perp$

- If $\mathbf{M}$ is invertible, its pseudo-inverse is $\mathbf{M}^+ = \mathbf{M}^{-1}$.
- If $\mathbf{x}$ is a column vector, its pseudo-inverse is $\frac{1}{||\mathbf{x}||^2}\mathbf{x}^H$

Least square fitting of data

Theorem

Assume that the sample vector $\mathbf{b} \in \mathbb{R}^n$ obtained from the model

$$\mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{n}$$

where

- $\mathbf{x}$ is the input data vector,
- $\mathbf{A}$ is the model,
- and $\mathbf{n}$ is the noise vector.

To get the minimal norm solution by solvingthe optimization problem

$$\min_{\mathbf{x}} ||\mathbf{A}\mathbf{x} - \mathbf{b}||^2 \qquad \mathbf{x} \in \mathbb{R}^n$$

then

$$\mathbf{x} = \mathbf{A}^+\mathbf{b}.$$

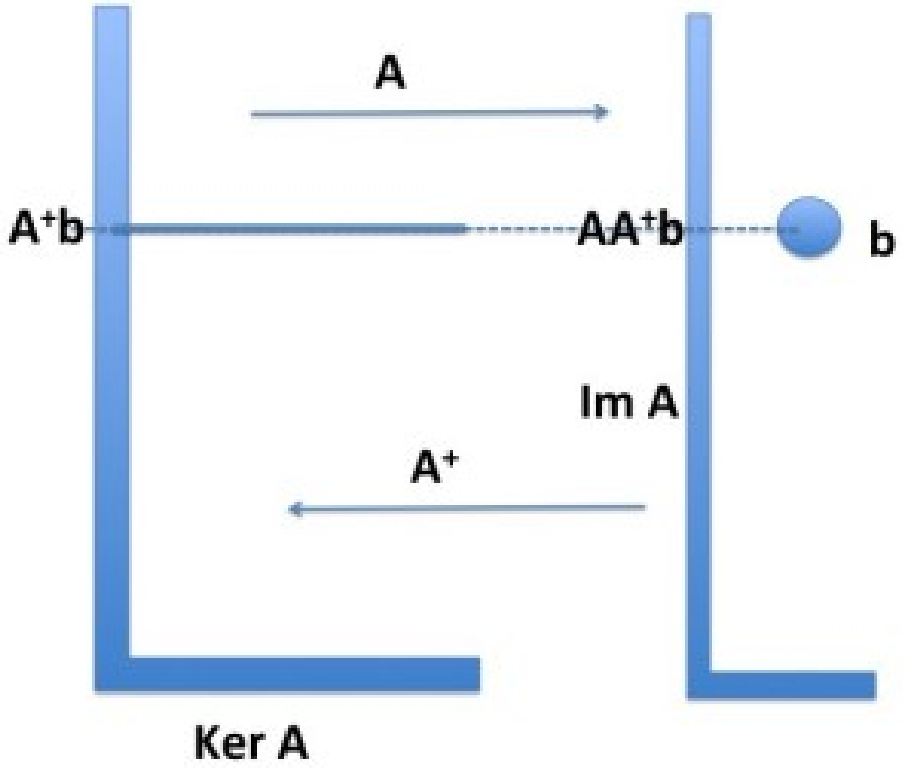


Figure: Least square solution of a linear system

Unsupervised learning

In machine learning, the problem of unsupervised learning is that of trying to find hidden structure in unlabeled data. Since the examples given to the learner are unlabeled, there is no error or reward signal to evaluate a potential solution.

- Unsupervised learning is considered as "data analysis" or "data mining",
- its objective is not to model an input-output system but to achieve "data compression".
- It is not only useful to save computation time memory space.
- Data analysis is necessary to implement it in a preliminary phase to build robust models, especially in case of high-dimensional data.
- Data analysis is nice to obtain a graphical representation of data (2d or 3d)

The problem of a principal component analysis

- Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ a n-sample of p-dimensional data
- $\mathbf{X}$ is a $(p \times n)$-array. The data may be qualitative or quantitative, we shall generally consider here real data.
- The problem of principal component analysis is to determine a q-D subspace $\mathbf{F}$ of $\mathbb{R}^p, (q < p)$ and to replace the data sample by its orthogonal projection onto $\mathbf{F}$ with respect to an appropriate scalar product.

Problem

Find $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_q]$ q-orthonormal subsytem of $\mathbb{R}^p$ solution of

$$\min_{i=1}^{n} ||\mathbf{x}_i - \sum_{j=1}^{q} \mathbf{x}_i^H \mathbf{v}_j \mathbf{v}_j||^2$$

Notation: $\mathbf{x}_i - \sum_{j=1}^{q} \mathbf{x}_i^H \mathbf{v}_j \mathbf{v}_j$ is called
the residue of the projection

- Form the (p, p) symmetric positive matrix $\mathbf{\Gamma} = \mathbf{X}^H \mathbf{X}$
- Form its spectral decomposition $(\lambda_j, \mathbf{v}_j)$ of with $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_p$
- Choose the q first eigenvectors of the orthonormal basis. Control the cumulated spectral ratio $\frac{\sum_{k=1}^{i} \lambda_k}{\sum_{k=1}^{p} \lambda_k}$

Remark

$\mathbf{X}^H$ is formed with the centered sample (covariance matrix) or with the centered reduced sample (correlation matrix)

The problem of clustering

- The problem is to divide the data space $\mathbb{R}^p$ into k regions.
- Each region is represented by a "centroid", i.e. a representative element
- Each data x_i is replaced by the closest centroid μ_i(Voronoi partition of the sample into $(S_j)_{j\in\{1\dots k\}}$.
- The centroids are chosen in order to minimize SSE (sum squared error)

Problem

Find $(\mu_j)_{j\in\{1\dots k\}}$in order to minimize

$$SSE = \sum_{i=1}^{k} \sum_{x_j \in S_i} ||x_j - \mu_i||^2$$

is the Voronoi partition of the sample.

k-means algorithms

K-means algorithm divide the samples into k cluster.

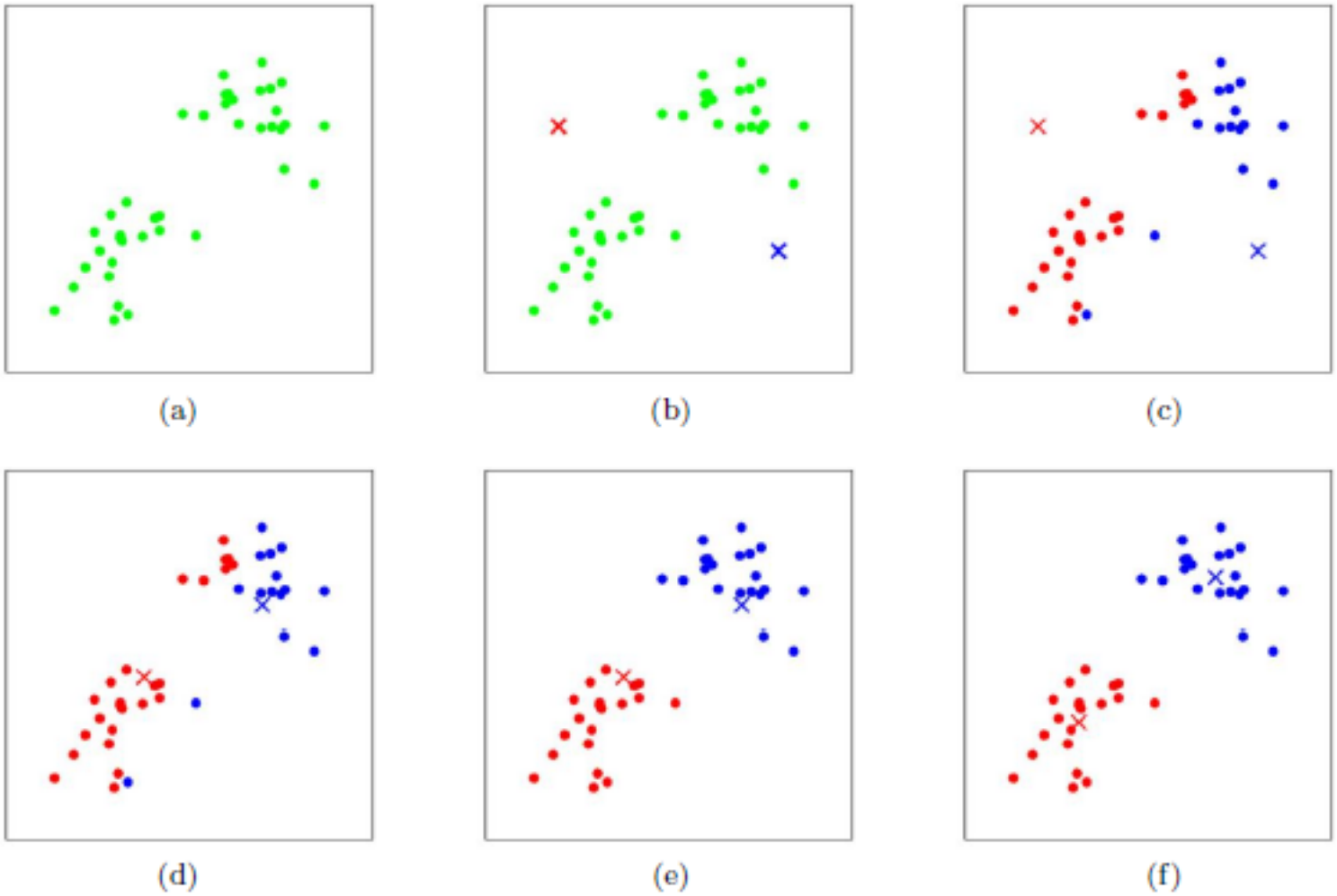
1. Initialize the k-means（cluster centroids）
   为$\mu_1, \mu_2, \cdots, \mu_k \in R^n$。
2. THEN REPEAT UNTIL CONVERGENCE:
   - implement the Voronoi partition $c^{(i)}$ of the sample $x^{(i)}$ according to μ_j current position

$$c^{(i)} = \arg\min_j \|x^{(i)} - \mu_j\|^2.$$

   - update the k-means position μ_j

$$\mu_j := \frac{\sum_{i=1}^m \mathbf{1}\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m \mathbf{1}\{c^{(i)} = j\}}$$

k-means algorithms



Density estimation

Problem

Estimate a probability density $f_X(x)$ from a iid sample (x_i)

The solution is obtained through the Parzen window as the convolution of the empirical law with a kernel k_σ:

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} k_\sigma(x - x_i) \qquad k_\sigma(x) = \frac{1}{\sigma}k\left(\frac{x}{\sigma}\right)$$

σ is chosen in function of the size of the sample: it is small when the size is large and reversely. The choice may be local.

Supervised learning

Definition

Supervised learning is the machine learning task of inferring a function from labeled training data. The training data consist of a set of training examples.

In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal).

A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

Linear regression

Given a data set $\{y_i, x_{i1}, \ldots, x_{ip}\}_{i=1}^n$ of n statistical units, a linear regression model assumes that the relationship between the dependent variable y_i and the p-vector of regressors $\mathbf{x}_i$ is linear.

This relationship is modeled through a disturbance term or error variable ε_i — an unobserved random variable that adds noise to the linear relationship between the dependent variable and regressors. Thus the model takes the form

$$y_i = \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\mathrm{T}} \boldsymbol{\beta} + \varepsilon_i, \qquad i = 1, \ldots, n,$$

where T denotes the transpose, so that $\mathbf{x}_i^{\mathrm{T}} \boldsymbol{\beta}$ is the inner product between vectors $\mathbf{x}_i$ and $\boldsymbol{\beta}$.

Often these n equations are stacked together and written in vector form as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} \mathbf{x}_1^{\mathrm{T}} \\ \mathbf{x}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{x}_n^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix},$$

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$

Example

Consider a situation where a small ball is being tossed up in the air and then we measure its heights of ascent h_i at various moments in time t_i. Physics tells us that, ignoring the drag, the relationship can be modeled as

$$h_i = \beta_1 t_i + \beta_2 t_i^2 + \varepsilon_i,$$

where

- β_1 determines the initial velocity of the ball,
- β_2 is proportional to the standard gravity,
- and ε_i is due to measurement errors.

Linear regression can be used to estimate the values of β_1 and β_2 from the measured data. This model is non-linear in the time variable, but it is linear in the parameters β_1 and β_2; if we take regressors $\mathbf{x}_i = (x_{i1}, x_{i2})^H = (t_i, t_i^2)^H$, the model takes on the standard form

$$h_i = \mathbf{x}_i^{\mathrm{T}} \boldsymbol{\beta} + \varepsilon_i.$$

The problem of linear regression

Problem

Modified the linear regression

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

as

$$\mathbf{y} = \widetilde{\mathbf{X}}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where

- $\widetilde{\mathbf{X}} = [\mathbf{x}_{\cdot 1} - \overline{\mathbf{x}}_{\cdot 1}\mathbf{1}, \ldots, \mathbf{x}_{\cdot p} - \overline{\mathbf{x}}_{\cdot p}\mathbf{1}, \mathbf{1}]$ be the $(n, p+1)$ matrix of centered sample,
- $\overline{\mathbf{x}}_{\cdot k} = \frac{1}{n}\sum_{i=1}^{n} x_{ik}$ is the empirical means of $\mathbf{x}_{\cdot k}$,
- and β is the $(p+1, 1)$ column vector of coefficients of linear regression.

We want to solve the LSE (Least Square Error) problem

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} ||\mathbf{y} - \widetilde{\mathbf{X}}\boldsymbol{\beta}||^2$$

Computation of linear regression

We saw that the solution is provided by the pseudo-inverse

$$\hat{\boldsymbol{\beta}} = \widetilde{\mathbf{X}}^{+}\mathbf{y}$$

. More precisely we have

$$\widetilde{\mathbf{X}}^{+} = (\widetilde{\mathbf{X}}^{H}\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}^{H}$$

We get $\widetilde{X}^{H}\widetilde{X}=n\begin{bmatrix} Cov(X) & 0 \\ 0 & 1 \end{bmatrix}$ and $\widetilde{X}^{H}Y = n\begin{bmatrix} Cov(X,Y) \\ \overline{Y} \end{bmatrix}$

Eventually, the predictive regression model is

$$Y = Cov(X)^{-1}Cov(X,Y)(X - \overline{X}) + \overline{Y}$$

- Here $Cov(X)$ and $Cov(X,Y)$ are empirical covariances. We find the empirical version of linear regression (C0304).
- We supposed the rank of the centered sample matrix is $p+1$, i.e. that the input sample covariance is invertible.

Additional remarks

Remark

① PCA is useful to stabilize the inverse of the variance matrix and to make covariance estimations more robust
② Non linear model with respect to x may be considered by adding new features to the sample as higher degrees monoms.
③ The method just needs that the linearity of the model with respect to parameter β

Manuel SAMUELIDES, Zhigang SU

中国民航大学

Recursive training and on-line training

- Instead of applying straightforward inversion, it may be more convenient to use recursive algorithms as gradient algorithms. For linear models the formula is

$$\beta_{k+1} = \beta_k + \gamma_{k+1} \widetilde{X}^H (Y - \widetilde{X}\beta_k)$$

  The convergence speed is controlled by the step $\gamma_k \downarrow 0$
- Notice that the Newton algorithm amounts to the straightforward formula
- On-line training amounts to train the current model with a single example $(\widetilde{x}^{(n+1)}, y^{(n+1)})$ at each step.

On-line estimation

- LLN ensures the convergence of the empirical mean of an increasing iid sample $\overline{X}_n = \frac{1}{n}(X_1 + \cdots + X_n)$ which is the solution of min $min_\theta \, \mathbb{E}(||X - \theta||^2)$
- Following the same track, it can be shown that

$$\beta^{(n+1)} = \beta^{(n)} + \frac{1}{n}[y^{(n+1)} - (\widetilde{x}^{(n+1)}|\beta^n)]\widetilde{x}^{(n+1)}$$

converges towards the solution of $min_\beta \, \mathbb{E}(||Y - (\widetilde{X}|\beta)||^2$ which is linear regression if $\widetilde{X} = (X - \mathbb{E}(X), 1)$ or polynomial regression if $\widetilde{X} = (X^d, \ldots, X, 1)$

Variable selection

- Variable selection has to be used especially in the case of NL dependent variables as in polynomial regression.
- Feature variable are selected sequentially according to the following recursive process:
  ① Compute the empirical correlation coefficients $\frac{Cov(y,x_{.j})^2}{Var(y)Var(x_{.j})}$ of the output with all remaining input variables,
  ② Select the more correlated input variable
  ③Project orthogonally on the selected feature the output and the remaining input variables and replace them by their projection residue.
- Process is stopped when the correlation is too low. The process may be designed using the Fisher test where the null hypothesis is a null regression coefficient.

Non linear models

When the model is non linear with respect to the parameter, recursive algorithms are used to minimize MSE. The most used models are

- Radial basis functions (RBF): $f(x,w) = \sum \alpha_m g(\frac{||x-x_m||}{\sigma_m})$ where g may be the gaussian kernel. The parameter is $w = ((\alpha_m, \sigma_m))$.
- Neural networks (NN): :$f(x,w) = \sum \alpha_m g((w_m|\widetilde{x}))$ where g is the sigmoid kernel (arg). The parameter is $w = ((\alpha m, w_m))$.
- Gradient of these models with respect to w are easy to compute.

Learning non linear models

- The current learning algorithms are
  - quasi-Newton algorithms (BFGS)
  - Bayes algorithms (Levenberg-Marquardt)
  - stochastic gradient descent for on-line learning
- The problem is to avoid bad local minima which may occur since MSE is non quadratic with respect to w.
- Stochastic gradient, penalization of unlikely values of parameter (Bayes) are currently used for that purpose.
- A lot of implementations with different initial values of w are useful to select the best final configuration

Training error and generalization error

Problem

The aim of supervised learning is to minimize $\mathbb{E}([Y - f(X, w)]^2)$. But the probability law of the data is not known.

- So it is replaced by an estimation on the learning base $\sum(y_i - f(x_i, w))^2$the so-called learning error
- But the best model to minimize the learning error is generally not the best for the generalization error.
- Notably, high dimensional models (high degree polynoms, many center RBF, many neuron NN) are learning noise. This is the "overfitting" phenomenon.
- To check generalization model ability, learning and testing have to rely on different process

Training error and generalization error

- The simpler way is to partition the available data into the learning base and the test base. The models are built using the learning base and tested on the test base.
- When the data are scarse, Leave-One-Out is used to test the model hyperparameter (structure or dimension):

Leave one out Algorithm

FOR $i = 1 : n$

- build a learning base $\mathcal{B}_i$ with the original base without x_i,
- implement the learning process and get a model w_i from $\mathcal{B}_i$,
- Test this model and compute the error
  $SE_i = (y_i - f(x_i, w))^2$
  Estimate the generalization error by $MSE = \frac{1}{n}\sum_{i=1}^{n} SE_i$

summary

- The main techniques of statistical learning have been shortly reviewed.
- The basic technique is linear regression, it i still very helpful.
- Unsupervised and supervised learning use massive parallel computing.
- These techniques are between numerical analysis and statistics
- Other learning technique as reinforcement learning are closer from adaptive learning or living beings.
- In that sense, learning is part of Artifical Intelligence