ICT for Health
Linear regression

Monica Visintin

Politecnico di Torino

2016/17

Table of Contents

Table of Contents

What is regression?

- To “regress” means to “go back”
- From the observed values $y(n)$, $n = 1, \ldots, N$, we go back to the variable $\mathbf{x}(n)$ (dimension F), that can be used later to explain or predict $y(n)$, $n > N$, using $\mathbf{x}(n)$, $n > N$.
- $\mathbf{x}(n)$ is the independent variable, $y(n)$ is the dependent variable, but we don’t know the relation that links $\mathbf{x}(n)$ and $y(n)$; $\mathbf{x}(n)$ is also called **regressor**, or **predictor** or **explanatory variable**; $y(n)$ is also called **regressand**, **response variable**, **measured variable**
- “When we regress Y on X, we use the values of variable X to predict those of Y”

Regression techniques [1]

- In **linear regression** we assume that $y(n) = [\mathbf{x}(n)]^T \mathbf{w} + \nu(n)$ where $\mathbf{x}(n) = [x_1(n), \ldots, x_F(n)]^T$ is the vector of the regressors (independent variables), $\mathbf{w} = [w_1, \ldots, w_F]$ is the set of weights to be found, and ν is a measurement error; each of the following techniques generates a different value of $\mathbf{w}$, starting from N observations, i.e. the column vector $\mathbf{y} = [y(1), \ldots, y(N)]^T$ and the matrix $\mathbf{X}$ whose n-th row is the vector $[\mathbf{x}(n)]^T = [x_1(n), \ldots, x_F(n)]$:
  1. Least-squares estimation: minimization of $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$ gives $\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ (this lecture)
  2. Least-squares estimation with regularization (ridge regression): minimization of $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda\|\mathbf{w}\|^2$ (we avoid that $\|\mathbf{w}\|$ gets to infinity if the problem is ill posed) gives $\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$
  3. Lasso regression: minimize $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$ subject to $\sum_i \|w_i\| < t$
  4. Maximum likelihood regression
  5. Bayesian linear regression
  6. Principal Component Regression (PCR) (next lecture)

Regression techniques [2]

- Nonlinear regression: $\mathbf{y} = f(\mathbf{w}, \mathbf{x})$ where $f(\cdot)$ is a nonlinear function, which depends on the specific problem (for example, $y(n) = w_1 \exp(-w_2 x(n))$ where w_1 and w_2 are the unknown parameters

Table of Contents

Definition of the problem [1]

- **(1-dimensional problem)** We have a set of values $u(n)$, $n = 1, \ldots, N$, and we know that the observed/measured value $y(n)$ is related to $u(n)$ through the equation

$$y(n) = af(u(n)) + \nu(n) = ax(n) + \nu(n)$$

where a is an unknown coefficient, $\nu(n)$ is the measuring error, $x(n) = f(u(n))$ is a known function of $u(n)$. Unfortunately noise $\nu(n)$ affects the measure $y(n)$, otherwise a single measurement would be sufficient to know the exact value of a.
Thus, we have a known column vector $\mathbf{x} \in \mathbb{R}^{N\times 1}$, a known column vector $\mathbf{y} \in \mathbb{R}^{N\times 1}$ and

$$\mathbf{y} = a\mathbf{x} + \boldsymbol{\nu}$$

where $\boldsymbol{\nu} \in \mathbb{R}^{N\times 1}$ identifies noise, and a is a real unknown constant.

Definition of the problem [2]

- **(1-dimensional problem with a variation)** We assume that

$$y(n) = ax(n) + b + \nu(n) \quad \Longrightarrow y(n) = ax(n) + b1(n) + \nu(n)$$

$$y(n) = [x(n), 1(n)] \begin{bmatrix} a \\ b \end{bmatrix} + \nu(n), \quad n = 1, \ldots, N$$

$$\mathbf{y} = [\mathbf{x}, \mathbf{1}] \begin{bmatrix} a \\ b \end{bmatrix} + \boldsymbol{\nu} = \mathbf{X}\mathbf{a} + \boldsymbol{\nu}$$

Actually we have a two-dimensional problem: the two unknowns (a and b) are stored in vector $\mathbf{a}$, and matrix $\mathbf{X}$ has a first column equal to $\mathbf{x}$, and a second column of N ones.

Definition of the problem [3]

- **(M-dimensional problem)** More in general, we assume that

$$y(n) = a_1 f_1(u(n)) + a_2 f_2(u(n)) + \cdots + a_M f_M(u(n)) + \nu(n)$$

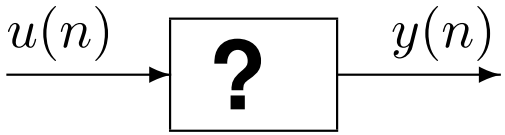
and we perform N measurements, by feeding the system with $u(n)$ and observing the corresponding $y(n)$; we want to estimate $\mathbf{a} = [a_1, a_2, \ldots, a_M]^T$ (we go back to the previous case by setting $M = 1$). By defining the matrix $\mathbf{X}$ with M columns and N rows, such that $\mathbf{X}(n, k) = f_k(u(n)) = x_k(n)$, we can write

$$\mathbf{y} = \mathbf{X}\mathbf{a} + \boldsymbol{\nu}$$

As an example, we have $\mathbf{X}(n, k) = [u(n)]^{k-1}$ if we assume that

$$y(n) = a_1 + a_2 u(n) + a_3 [u(n)]^2 + \cdots + a_M [u(n)]^{M-1} + \nu(n)$$

A pictorial view



Model:

$$y(n) = a_1 f_1(u(n)) + a_2 f_2(u(n)) + a_3 f_3(u(n)) + \ldots a_M f_M(u(n)) + \nu(n)$$

	Ex. 1	Ex. 2	Ex. 3
$f_1(x)$	1	$\sin(2\pi x)$	$g(x)$
$f_2(x)$	x	$\cos(2\pi x)$	$g(x-1)$
$f_3(x)$	x^2	$\sin(4\pi x)$	$g(x-2)$
$f_4(x)$	x^3	$\cos(4\pi x)$	$g(x-3)$
$f_5(x)$	x^4	$\sin(6\pi x)$	$g(x-4)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

$g(x) = \exp(-|x|)$ or something else.

Comment

In our previous discussion, the model is given. In real life, the main problem is that of finding the correct functions $f_k(u)$ for the specific system we are analyzing. However we concentrate on the solution of the problem of finding $\mathbf{a}$, assuming that the correct model has been devised.
Therefore, in the equation

$$\mathbf{y} = \mathbf{X}\mathbf{a} + \boldsymbol{\nu}$$

matrix $\mathbf{X}$ is given, $\mathbf{y}$ is measured, $\mathbf{a}$ has to be estimated.
In particular, in the laboratory experience, $\mathbf{X} \in \mathbb{R}^{N \times F}$ stores the F features, which can be thought of as F functions of $u(n)$, $n = 1, \ldots, N$.

Table of Contents

First solution (MSE)

MSE: Minimum Square Error

We find $\mathbf{a}$ that minimizes the square error

$$e(\mathbf{a}) = \|\mathbf{y} - \mathbf{X}\mathbf{a}\|^2$$

($e(\mathbf{a})$ is a scalar, $\mathbf{y}$ is a column vector with N elements, $\mathbf{a}$ is the unknown column vector with M elements, $\mathbf{X}$ is a matrix with N rows and M columns).
The square error can be rewritten as

$$\begin{aligned} e(\mathbf{a}) &= [\mathbf{y} - \mathbf{X}\mathbf{a}]^T[\mathbf{y} - \mathbf{X}\mathbf{a}] \\ &= \mathbf{y}^T\mathbf{y} - \mathbf{a}^T\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\mathbf{a} + \mathbf{a}^T\mathbf{X}^T\mathbf{X}\mathbf{a} \end{aligned}$$

We say that $e(\mathbf{a})$ is a quadratic form of $\mathbf{a}$. In a one dimensional space (i.e. $M = 1$), $e(a)$ is a quadratic function (convex up) corresponding to a parabola; in a two dimensional space (i.e. $M = 2$), $e(\mathbf{a})$ corresponds to a bowl. Whatever is M, $e(\mathbf{a})$ has just one minimum value; we must find $\hat{\mathbf{a}}$ that corresponds to this minimum.

First solution (MSE)

To find the minimum is thus sufficient to find the value $\hat{\mathbf{a}}$ for which the gradient of $e(\mathbf{a})$ vanishes:

$$\nabla e(\mathbf{a}) = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\mathbf{a} = \mathbf{0}$$

We get the solution

MSE

$$\hat{\mathbf{a}} = [\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{y}$$

Note that $[\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T$ is the so-called **pseudo-inverse** of the rectangular matrix $\mathbf{X}$.
Then $\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{a}}$ and

$$\begin{aligned} e(\hat{\mathbf{a}}) &= \mathbf{y}^T\mathbf{y} - \hat{\mathbf{a}}^T\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\hat{\mathbf{a}} + \hat{\mathbf{a}}^T\mathbf{X}^T\mathbf{X}\hat{\mathbf{a}} \\ &= \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}[\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}[\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{y} + \\ &+ \mathbf{y}^T\mathbf{X}[\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{X}[\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{y} \\ &= \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}[\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{y} \end{aligned}$$

Second solution (iterative, gradient algorithm)

The MSE solution requires that a matrix is inverted, which might be too complex is some cases (for example image processing). Then the following iterative solution can be used:

Gradient algorithm

1. Start with an initial guess of $\hat{\mathbf{a}}$, which can be just a vector of M random variables. Let $\hat{\mathbf{a}}(0)$ be this initial guess. Set $i = 0$.
2. Evaluate the gradient
$$\nabla e(\hat{\mathbf{a}}(i)) = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\hat{\mathbf{a}}(i)$$
3. Update the guess:
$$\hat{\mathbf{a}}(i+1) = \hat{\mathbf{a}}(i) - \gamma\nabla e(\hat{\mathbf{a}}(i))$$
where $\gamma > 0$ is the so-called learning coefficient
4. Increase i by 1 and go back to step 2, until a stopping rule is satisfied

A couple of details

- The **correct value of** γ depends on the specific function to be minimized:
  - if γ is too large, then a sort of hysterical behavior arises in the algorithm (we jump around the optimum value but we never reach it, because the jumps are too large)
  - if γ is too small, then it takes a lot of time to reach the optimum value.
- An example of stopping rule is $\|\hat{\mathbf{a}}(i+1) - \hat{\mathbf{a}}(i)\| < \epsilon$.

Overfitting

If $\mathbf{y} = \mathbf{X}\mathbf{a} + \boldsymbol{\nu}$ has some large values of noise/error, it is possible that the error $\hat{\mathbf{a}}$ takes very large values. Then, it might be convenient to solve the new problem

$$\min_{\mathbf{a}} \|\mathbf{y} - \mathbf{X}\mathbf{a}\|^2 + \mu\|\mathbf{a}\|^2$$

where μ has to be set conveniently (trial and error).

Third solution (iterative, steepest descent algorithm)

A faster convergence can be obtained with the steepest descent algorithm, which finds the "optimum" value of γ at each step.

The steepest descent algorithm

1. Start from an initial guess $\hat{\mathbf{a}}(0)$, set $i = 0$
2. Evaluate the gradient and the Hessian matrix at point $\hat{\mathbf{a}}(i)$,

$$\nabla e(\hat{\mathbf{a}}(i)) = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\hat{\mathbf{a}}(i)$$

$$\mathbf{H}(\hat{\mathbf{a}}(i)) = 4\mathbf{X}^T\mathbf{X}$$

3. Find the new point as

$$\hat{\mathbf{a}}(i+1) = \hat{\mathbf{a}}(i) - \frac{\|\nabla e(\hat{\mathbf{a}}(i))\|^2}{\nabla e(\hat{\mathbf{a}}(i))^T \mathbf{H}(\hat{\mathbf{a}}(i)) \nabla e(\hat{\mathbf{a}}(i))} \nabla e(\hat{\mathbf{a}}(i))$$

4. set $i := i + 1$, go back to step 2, unless a stop condition is met

A variation: the stochastic gradient algorithm

1. Start with an initial guess $\hat{\mathbf{a}}(0)$; set $n = 1$
2. Define the error at the n-th epoch:
$$e(\hat{\mathbf{a}}(n), n) = \|y(n) - \mathbf{x}(n)\mathbf{a}\|^2$$
where $\hat{\mathbf{a}}(n)$ is the estimation of the unknown vector $\mathbf{a}$ at the n-th epoch, and $\mathbf{x}(n)$ is the n-th row of matrix $\mathbf{X}$
3. Evaluate the gradient of $e(\hat{\mathbf{a}}(n), n)$:
$$\nabla e(\hat{\mathbf{a}}(n), n) = -2y(n)\mathbf{x}(n) + 2[\mathbf{x}(n)]^T\mathbf{x}(n)\mathbf{a}$$
4. Update the estimate as
$$\hat{\mathbf{a}}(n+1) = \hat{\mathbf{a}}(n) - \gamma\nabla e(\hat{\mathbf{a}}(n), n)$$
5. Set $n := n + 1$; if $n > N$, set $n = 1$; go back to step 2 unless a stopping rule is satisfied.

With respect to the gradient algorithm, we consider one epoch at a time, instead of all the epochs together. Intermediate solutions are possible in which, for example, having $N = 1000$, you take 10 epochs at a time (a mini-batch).

Table of Contents

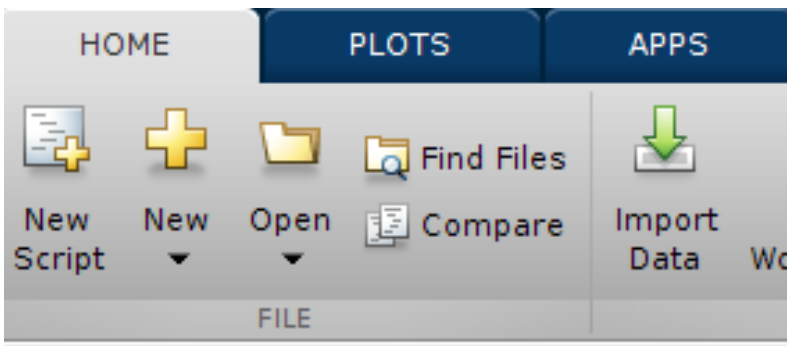
Training

- In general we have a set of N cases, and we divide them into two subsets: the cases $n = 1, \ldots, N_{tr}$ are used for training the machine, the cases $n = N_{tr} + 1, \ldots, N$ are used to test the machine
- Of course, if N_{tr} is too small, the machine is not correctly trained; if N_{tr} is too large, the results obained during the testing phase are not significant from a statistical point of view; a compromise must be found
- In our specific case, matrix $\mathbf{X}$ used to find $\hat{\mathbf{a}}$ has N_{tr} rows and M columns; then $\hat{y}(n) = \mathbf{x}(n)\hat{\mathbf{a}}$ for $n = N_{tr} + 1, \ldots, N$ is the estimate of $y(n)$ and the prediction error is $e(n) = y(n) - \hat{y}(n)$.

Table of Contents

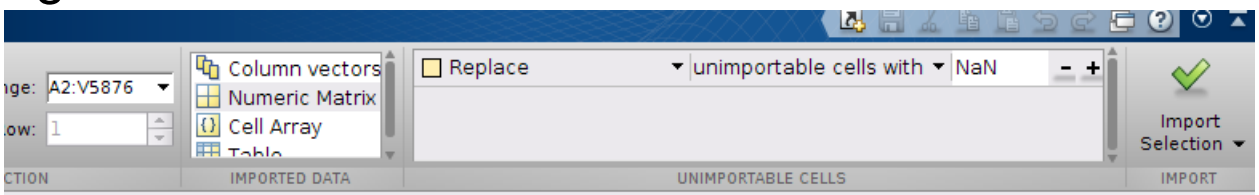
Prepare the data [1]

- Download from `https://archive.ics.uci.edu/ml/datasets/Parkinsons+Telemonitoring` the Data Folder and Data Set Description. In particular, download files `parkinsons_updrs.data` and `parkinsons_updrs.names` from `https://archive.ics.uci.edu/ml/machine-learning-databases/parkinsons/telemonitoring/`
- File `parkinsons_updrs.data` stores F columns and N_T rows; the columns are separated by commas.
- Open Matlab and click on “Import Data”



Prepare the data [2]

- Select `parkinsons_updrs.data` by changing “Files of type” from the default value to “All files”
- You’ll see 22 columns (the first line shows the content of each column).
- Select “Numeric Matrix” on the left and “Import Selection” on the right



- In the Matlab Command Window, write
  `save('updrs.mat','parkinsonsupdrs');`
  so that later, in the Matlab script, you can simply write
  `load('updrs.mat')`
  to work on matrix `updrs`.

Prepare the data [3]

- For each patient (identified by an integer in the first column), the interesting features are stored in columns from 5 to 22
- Column 4 stores the time (in days) at which the measurements were taken during the study, which lasted 6 months
- In each measurement day, the physician analyzed the patient and gave him/her two grades in the Unified Parkinson's Disease Rating Scale (UPDRS):
  - the total UPDRS value (column 6)
  - the motor UPDRS (column 5)

  these two measurements are expensive and we would like to just estimate them using the other features.
- Columns from 7 to 22 store these other features. Each feature was actually measured 5-6 times during each measurement day, and the files stores each of these 5-6 values (not the average).

Prepare the data [4]

- For each patient, time goes from approximately 0 to approximately 180 days 5-6 times in subsequent blocks of data; since total and motor UPDRS values were measured just once in the day, these values are just copied in each of the 5-6 blocks, whereas each block stores a different value for each of the other features.
- **Prepare a new matrix** in which column 4 goes just once from 0 to 180 for each patient, and the values stored in columns from 7 to 22 are the means of the values stored in the 5-6 blocks of the original file. You should get a new matrix `data` with 990 rows (instead of the 5876 present in `parkinsons_updrs.data`).

Table of Contents

Perform regression [1]

- Consider only the data related to the first 36 patients (leave patients from 37 to 42 for the testing phase of the learning machine), and store these data into a new matrix `data_train`. Define the matrix `data_test` with the data of patients from 37 to 42.
- It is convenient to normalize the data, so that each feature has mean zero and variance 1; therefore find `m_data_train=mean(data_train,1);` and `v_data_train=var(data_train,1);` then generate the normalized data `data_train_norm` and check that the normalized features have zero mean and unit variance.
- Generate `data_test_norm` using the values `m_data_train` and `v_data_train` measured at the previous point

Perform regression [2]

- Define `F0` as the feature that we want to estimate from the other features; then `y_train=data_train_norm(:,F0);` and `X_train=data_train_norm;X_train(:,F0)=[];` similarly define `y_test=data_test_norm(:,F0);` and `X_test=data_test_norm;X_test(:,F0)=[];`
- start with `F0=7` (once the script works with `F0=7`, then you'll use `F0=5`)
- Perform regression using
  1. MSE,
  2. the gradient algorithm,
  3. the steepest descent algorithm

  and predicting feature `F0` from the other features. In the last two cases, start with a random vector $\hat{\mathbf{a}}(0)$, but generate it after the command `rng('default')`, so that each time you run your script you get the same random vector.

Perform regression [3]

- Generate the following plots:
  1. `yhat_train` versus `y_train` and `yhat_train_L` versus `y_train`
  2. `yhat_test` versus `y_test` and `yhat_test_L` versus `y_test`
  3. the histograms of `y_train-yhat_train` (50 bins)
  4. the histograms of `y_test-yhat_test` (50 bins)
  5. the values of $\mathbf{w}$
- Write the report:
  - use the publish function of the Matlab **editor** to automatically generate a first pdf file that contains the Matlab script together with the obtained figures (for `F0=7` and `F0=5`) (this is automatic),
  - write another pdf file with your comments, referring to the figures of the Matlab report.