Lab 1 : supervised learning

January 16, 2017

You will need to use the two following files :

- the source file `lab_ML_supervise_source.py`
- an example of script `lab_ML_supervise_script.py`

Some website may be useful to refresh your knowledge of python

- `http://perso.telecom-paristech.fr/ gramfort/liesse_python/1-Intro-Python.html`
- `http://perso.telecom-paristech.fr/ gramfort/liesse_python/2-Numpy.html`
- `http://perso.telecom-paristech.fr/ gramfort/liesse_python/3-Scipy.html`
- `http://scikit-learn.org/stable/index.html`
- `http://www.loria.fr/ rougier/teaching/matplotlib/matplotlib.html`
- `http://jrjohansson.github.io/`

1 Introduction

Definitions and notations
In what follows we shall use the following notations :

- $\mathcal{Y}$ denotes the labels. Usually $\mathcal{Y} = \{-1, 1\}$ (binary classification)
- $x = (x_1, \cdots, x_p) \in \mathcal{X} \subset \mathbb{R}^p$ are the features
- $D_n = \{(x_i, y_i),\, i = 1, \cdots, n\}$ is the training set
- We assume that there exists a probabilistic model explaining the generation of our observations :
$$\forall i \in \{1, \cdots, n\}, (x_i, y_i) \text{ i.i.d. } \sim (X, Y) \ .$$
- Using the training set D_n we want to construct a prediction function $\widehat{f} : \mathcal{X} \to \{-1, 1\}$ which which predicts an output y for a given new x with a minimum probability of error.

Data generation
We assume in this part that our data are bidimensional

1. Test the functions rand_gauss(n,mu,sigma), rand_bi_gauss, rand_clown and rand_checkers. Explain what each function is doing

2. Save some datasets

3. Plot some dataset using the function plot_2d

Extension to the multi-class setting
In the case where the oupput variable Y is not binary there is several ways to extend the binary setting

- One against one : one considers all possible pairs of labels (k, l) and fit a classifieur $C_{k,l}(X)$ for each one. We then predict the output which won most of the fights.

- One against all : for each k, we learn a classifieur discriminating between $Y = k$ and $Y \neq k$. Using the a posteriori probabilities, we set the most probable label

2 Logistic regression

Import the package `sklearn.linear_model` which contains the class `LogisticRegression`
from sklearn import linear_model

1. Create a model LogisticRegression :
   `my_log = linear_model.LogisticRegression()`

2. To learn the model from the data `dataX` and their corresponding labels `dataY`, one can use `fit`
   `my_log.fit(dataX,dataY)`
   One can see some examples on the following website :
   http://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html#example-linear-model-plot-ols-py
   http://scikit-learn.org/stable/auto_examples/linear_model/plot_iris_logistic.html

3. What is the variable `coef_` of the model ? `intercep_` ?

4. What is the output of the function `predict` ? This of the function `score` ?

5. Visualise the decision boundary.

6. Apply the logistic regression of the data from the database zipcode available on the website
   `http://www-stat.stanford.edu/ElemStatLearn`.

3 The perceptron

A perceptron is a linear binary classifieur projecting each observation in $\mathbb{R}$. The set of decision boundaries is the set of affine hyperplanes defined from a given vector $w = (w_0, w_1, \cdots, w_p) \in \mathbb{R}^{p+1}$ as

$$H_w = \{x : \widehat{f}_w(x) = w_0 + \sum_i w_i x_i = 0\}$$

To classify an observation x, one considers the position of x with respect to the hyperplane H_w. The predicted label is then $sign(\widehat{f}_w(x))$ The aim is to find the best hyperplane which separates the datas according to their labels.

Classical perceptron We assume here that our datas are bidimensionnal. Use the synthetic datas to answer the following questions

1. What is the boundary of the perceptron? When is $\widehat{f}_w(x)$ large ? negative ? positive ? What does it mean ?
2. Implement a function predict(x,w) which gives $\widehat{f}_w(x)$ from x. Implement also the function `predict_class(x,w)` which gives as output the label $sign(\widehat{f}_w(x))$

Cost function We want to measure the error on a dataset D_n. We then need to precise the loss function ℓ that we consider. The cost $Cw = \mathbb{E}\left[\ell(y, \widehat{f}_w(x))\right]$ is the expectation of the loss function on the whole dataset.

Three loss functions are classically used

- The zero-one loss: $ZeroOneloss(y; \widehat{f}_w(x)) = \left|y - sign(\widehat{f}_w(x))\right|/2$
- the quadratic error : $MSEloss(y; \widehat{f}_w(x)) = (y - \widehat{f}_w(x))^2$
- the hinge loss : $HingeLoss(y; \widehat{f}_w(x)) = \max(0, 1 - y \cdot \widehat{f}_w(x))$

We now study these different loss functions

1. Implement these functions.
2. Fix w on a bidimensional example. How vary these function with respect to $\widehat{f}_w(x)$? With respect to x ? What is the interpretation?
3. How can we observe the evolution of these two functions with respect to w for a given data set? Where is the vector

$$w \in argmin_{w \in \mathbb{R}^{p+1}} \frac{1}{n} \sum_{i=1}^{n} \ell(y, \widehat{f}_w(x_i))$$

Learning the perceptron in practice

In the general case, to minimize the cost function one use a gradient descent.

1. Recall the general perceptron algorithm
2. Experiment it on several datasets. Use either the functions given in the file either the `sklearn` package. One can also use the function `SGD` (Stochastic Gradient Descent). For more informations on this function see : `http://scikit-learn.org/stable/modules/sgd.html`
   `from sklearn.linear_model import SGDClassifier`

3. Study the numerical performances of the algorithm
4. The stochastic version of the algorithm consists in taking into account only the error on a randomly drawn examplecat each iteration. Modify the code and test it
5. Propose some variants on the stopping conditions of the algorithm
6. What is the main problem of perceptron ?

Is Perceptron linear?

1. What is the analytic formula of an ellipsis, an hyperbol and a parabol ?
2. Propose a method to classify the clown dataset. Can we generalize? One can use the function`poly2` of the source file associated to the lab. How can we use it to learn a perceptron?
3. Give the stochastic version of the perceptron algorithm
4. On the clown dataset, perform some numerical experiments and plot the decision boundaries