Logistic regression / Classification; cours 3

22/01/2019

Reminder

* **Supervised learning**
  + Regression
  + **Classification**
* Unsupervised learning
  + Clustering
  + Dimension reduction
* Reinforcement learning

# Classification

Logistic regression model = binary classification

“Regression veut dire que l’on cherche une relation entre x et y”

Goal : inputs are divided in classes and the ML must produce an algorithm that assign inputs to one or more of these classes

Main difference : we have classes and not numerical values.

*Eg : spam emails*

* *Spam or not spam*
* *Output has 2 categories*

Classification algorithms:

* Linear classifiers
  + **Logistic regression**
* Most used model

Naive Bayes classifier

Linear discriminant

* Support vector machines
  + Can be used both for linear regression and classification
* Decision trees
  + Can be used both for linear regression and classification
  + Can be more powerful but depend of the problem and the data
  + Random Forests
  + Boosting
* Quadratic classifiers
* Neural networks
* K-nearest neighbor
  + Used for classification

K-means most well known algorithm for clustering (in unsupervised learning)

### 1/ Why not Linear regression?

I want to know if the loan is safe or risky

* We can use linear regression for this.
* The y (output) values must be discrete.
* We fit one line between the loan safety and age.
  + y<0.5 : risky
  + y>0.5 : safe
  + Eg : x age = 30 ⇒ 0.33 → risky value
  + If add new data, this doesn’t work anymore. There will be points misclassified because they should be safe values, but they are risky

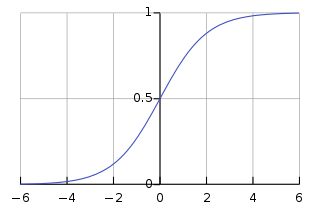
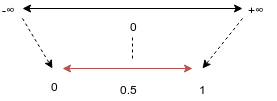
1. Linear regression can sometimes be lucky but is often not useful for classification problems
2. Classification need categorical values

* y = 0 or 1
* But LR : can be >1 or <0

### 2/ Logistic regression model

Change the form of our hypothesis: , to satisfy 0 < h < 1

→ use the sigmoid/logistic function :



When z = h(x) is between -∞ and 0 it gives g(z) between 0 and 0.5

is between +∞ and 0 it gives g(z) between 0.5 and 1

: estimated probability that y = 1 given the input x parameterized by

* This means that if = 0,8, y=1 is equal to 80%

Hypothesis :

* y = 1 when →
* y = 0 when →

### 3/ Decision Boundary

#### **a/ Linear :**

C’est la droite qui représente la frontière entre les 2 classes.

is our decision boundary

#### **b/ Non linear :**

La courbe peut être non linéaire et donc former d’autres formes non linéaire (due to a non-linear relationship (deg >=2))

* As with polynomial regression, we can have more complex decision boundaries by adding higher polynomial terms.

### 4/ Cost function

#### **a/ Quality Metric**

* Real class / Prediction class

#### **b/ Cost function**

* We use the same definition for the cost function the same way as we used for the Linear Regression
  + Pb : not linear nor convex.

⇒ We have to find a new cost function.

⇒ We need a **cost function with a convex shape**.

* When prediction = real ⇒ cost = 0

But we need something that gives high penalties (has a high cost) when it isn’t the case.

For one data point :

For all points :

)

* If y = 0; =
  + If = 0 ⇒ the cost = 0
  + If = 1 ⇒ the cost = ∞, this is the high penalty
* Same of y = 1 :
  + If = 1 ⇒ the cost = 0
  + If = 0 ⇒ the cost = ∞, this is the high penalty
* If equals something close to the real value then the cost would be small, it is far from the real value then the cost is important to correct it (high penalty).

### 

To minimize the cost function, we can use the gradient descent (it is a function used to minimize or maximize a function):

* Initiate with a random set of values for
* At each iteration, update the values of

#### **c/ Cost function: Maximum likelihood**

We choose a so we can maximize our probability for each training example

### Multiclass classification

* You can have c possible classes : 1, 2, … c
* m data points.
* One vs all : create a 2 class model for each class

### Evaluating classifiers

Accuracy = number of data points classified correctly / all data points

99% accuracy good ?

* Can be anything
* It depends on the proportion of the classes in your dataset

### Confusion matrix

In many real life problems you care more about predicting well one class more than the others

Use a confusion matrix

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Predicted class | |
| Negative | Positive |
| tmActual  class | Negative | True negative | False positive |
| positive | False negative | True positive |

Precision (%)=

### F1 score

Optimize to get better precision or recall. Can optimize for a single value, called F1score

F1score = 2 \* (precision \* recall) / (precision + recall) = … %

# K-fold cross-validation

**Data** = training set + testing set

* Each point is used only once for training and testing
* Variability !

Sometimes not enough data to have reliable partitions

**One-round Cross Validation**

|  |  |  |
| --- | --- | --- |
| X\_train | y\_train | 70 - 80% |
| X\_test | y\_test | 20 - 30% |

* It gives an insight on how the model will **generalize to an independent dataset.**

### K-fold

We want to partition our data in several data sets of the same size.

|  |  |  |
| --- | --- | --- |
| K = 5 | Fold 1 | Training set  Folds |
| Fold 2 |
| Fold 3 |
| Fold 4 |
| Fold 5 | Testing set  Fold 5 |

What if the interesting data is in the fold 5?

|  |  |  |
| --- | --- | --- |
| K = 5 | Fold 1 | Training set  Folds 1, 2, 3, 5 |
| Fold 2 |
| Fold 3 |
| Fold 4 | Testing set  Fold 4 |
| Fold 5 |  |

Average the fold values to know if it is a good model compared to other models.

Average is necessary to make sure we have good values in general to avoid outline values and bad fitting

Common types of cross-validations:

* Non exhaustive cross-validation
  + K-fold
  + 2-fold
* Exhaustive cross-validation
  + Leave-p-out
  + Leave-one-out
    - Lots of calculations because learn many many times since it does n-1 learning iterations
    - Use when not much data