Machine Learning for Medicine TP 5

Logistic Regression
K-fold cross validation

The goal of the TME is to implement an optimization procedure to fit a binary logistic regression. Another objective is to learn how to avoid overfitting, and to use a k-fold cross validation procedure.

Data (three simulated data sets + data sets of TME 1)

We explore two data sets downloadable from the Machine Learning Repository (`http://archive.ics.uci.edu/ml/index.php`)

- Breast Cancer Wisconsin (Diagnostic) Data Set (`https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)`)
- Mice Protein Expression Data Set (`https://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression`)

Libraries

You will need to load the following packages:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.datasets import make_blobs
from sklearn.datasets import make_moons
from sklearn import linear_model, datasets
```

Analysis

1. Test the logistic regression on the three simulated data sets (generated using `make_classification()`, `make_blobs()`, `make_moons()`) which we already explored in TME 2.

   ```
   logreg = linear_model.LogisticRegression(C=1e5)
   logreg.fit(X, Y)
   ```

   and plot the class boundaries. Here is an example how to do it:

   `http://scikit-learn.org/stable/auto_examples/linear_model/plot_iris_logistic.html#sphx-glr-auto-examples-linear-model-plot-iris-logistic-py`

2. We are in the context of supervised learning. To do the experiments properly, and not to overfit, a common practice is to use a k-fold cross validation technique. So, in all your experiments, apply 5-fold cross validation, i.e., split your data into 5 subsets, train on 4 parts, and test on 1, and repeat the procedure 5 times. The resulting accuracy is the average of 5 runs.

3. Implement in Python the interative procedure to fit a binary logistic regression.

   Binary Logistic Regression

   We have a training set of N observation $\{X_n, Y_n\}_{n=1}^N$. Here, we consider a binary logistic regression, and the variable Y $\in \{1, 0\}$. The variables X can be continuous or binary.

   The logistic regression is a parametric probabilistic models, and its log-likelihood is given as follows:

$$\ell(Y|X;\theta) = -\sum_{i=n}^{N} \Big(y_n \theta^T x_n - \log(1 + \exp(\theta^T x_n)) \Big), \tag{1}$$

   where θ is the vector of parameters to optimize.

   To classify a new observation X, we compute the probabilities of classes, and take the maximum:

$$p(Y = 1|X) = \frac{\exp \theta^T X}{1 + \exp \theta^T X}, \tag{2}$$

$$p(Y = 0|X) = \frac{1}{1 + \exp \theta^T X}. \tag{3}$$

   Optimization procedure: the method of Newton-Raphson

   The Newton-Raphson method is used to minimize the negative log-likelihood (to maximize the positive log-likelihood) and to estimate the parameters θ of the model. This is an iterative procedure of the gradient descent.

   For the case of the binary logistic regression, the algorithm is as follows:

   Initialize $\theta = (0, \dots, 0)$

   // Do some iterations or continue until convergence
   for $t = 1 : T$ **do**

   //Compute the first derivative
   $\frac{\partial \ell(\theta)}{\partial \theta} = -\sum_{n=1}^N x_n (y_n - p(y = 1|x_n))$ // dimension = 1 × nb of parameters

   //Compute the Hessian matrix
   $\frac{\partial^2 \ell(\theta)}{\partial \theta \partial \theta'} = \sum_{n=1}^N x_n x_n^t p(y = 1|x_n)(1 - p(y = 1|x_n))$
   //dimension = nb of parameters × nb of parameters

   //Update the parameters
   $\theta = \theta - \frac{\partial^2 \ell(\theta)}{\partial \theta \partial \theta'}^{-1} \frac{\partial \ell(\theta)}{\partial \theta}$
   end for

4. If you are lost, have a look in this blog

   `https://beckernick.github.io/logistic-regression-from-scratch/`

5. Test your version and the Python function of the binary logistic regression on three simulated data sets. Apply 10-fold cross validation.

6. Test both logistic regression versions on the Mice and Breast cancer data. Apply 10-fold cross validation.

7. Are the results similar? How many iterations were needed to converge?