

Apredizagem Automática

Assignment 1 - Classifiers

Andrea Mascaretti N52222

Daniel Pimenta N45404

October 21, 2017

Abstract

The main goal of this assignment was the parameterization, fitting and comparison of Logistic Regression, K-nearest Neighbours and Naive Bayes classifiers.

The data set used was the banknote authentication which was obtained from the UCI machine learning repository.

To achieve our goals we used the Spyder IDE with programming language Python 3.6 and some of its modules such as *Panda* to load and manage our data, *NumPy* which is the fundamental package for scientific computing with Python and was used for various mathematical calculations and managing multidimensional arrays, *sklearn (scikit-learn)* which has efficient tools for data mining and data analysis used in our assignment to preprocess data (shuffle, standardize, split), fold our data into k stratified folds, calculate cross validation score and fit our data on Logistic Regression and K-nearest Neighbours classifiers. Another imported module was Matplotlib.Pyplot which provides a MATLAB-like plotting framework used to build the error plots.

1 Introduction

WHAT WE INITIALLY HAD,
WHAT WAS ASKED TO DO,
HOW WE PLAN ON DOING IT

2 Classifiers

2.1 Logistic Regression

The logistic regression is a generalised linear model. Before considering a more classic machine learning framework, we provide a brief introduction to the main idea behind the model.

The logistic regression is composed, as every linear model, of three part:

- A sequence of *response variables* Y_1, \dots, Y_n . Such response variables are called random components. The main assumption we shall make regarding those variables is that they are all independent random variables. Moreover, each one of them will have a distribution from the exponential family. Bear in mind that we are not imposing that the various Y_i s are identically distributed.
- The *systematic component* is our model. It is a function of some predictors (also known as regressors or covariates), linear in the parameter and related to the mean of Y_i .
- The *link function* $g(\mu_i)$ will allow us to link the two components, allowing us to say that

$$g(\mu_i) = \beta_0 + \sum_{i=1}^r \beta_i x_i \quad (1)$$

where $\mu_i = \mathbb{E}[Y_i]$. In the machine learning framework, it is usually more common to consider the inverse of the link function.

In our model we will assume the following:

$$Y_i \overset{ind}{\sim} \text{Bernoulli}(\pi_i) \quad (2)$$

and therefore we will have that

$$\mathbb{E}[Y_i] = \pi_i. \quad (3)$$

Now, in this model we will assume a relationship between the logarithm of the odds of success for Y_i (the log odds or *logit*) and the predictor $\mathbf{x} = (1, x_1, \dots, x_r)$. Mathematically, this entails

$$\ln\left(\frac{\pi}{1-\pi}\right) = \beta' \mathbf{x} \quad (4)$$

and $\beta = (\beta_0, \dots, \beta_r)$.

Rewriting in order to obtain the exponential family form we have

$$\pi^y (1-\pi)^{1-y} = (1-\pi) \exp\left\{y \ln\left(\frac{\pi}{1-\pi}\right)\right\}, \quad (5)$$

where the term $\ln\left(\frac{\pi}{1-\pi}\right)$ is the natural parameter of the exponential family and shows why we will implement the link function

$$g(\pi) = \ln\left(\frac{\pi}{1-\pi}\right). \quad (6)$$

Rewriting 4, we obtain

$$\pi(x) = \frac{\exp\{\beta' \mathbf{x}\}}{1 + \exp\{\beta' \mathbf{x}\}}. \quad (7)$$

What we have written is equivalent to saying that

$$\mathbf{p}(\mathcal{C}_1 | \mathbf{x}) = \pi(\mathbf{x}) = \sigma(\beta' \mathbf{x}) \quad (8)$$

where $\sigma(\cdot)$ is the logistic sigmoid and \mathcal{C}_1 is to identify class 1. Notice that it holds $\mathbf{p}(\mathcal{C}_2 | \mathbf{x}) = 1 - \mathbf{p}(\mathcal{C}_1 | \mathbf{x}) = 1 - \sigma(\beta' \mathbf{x})$.

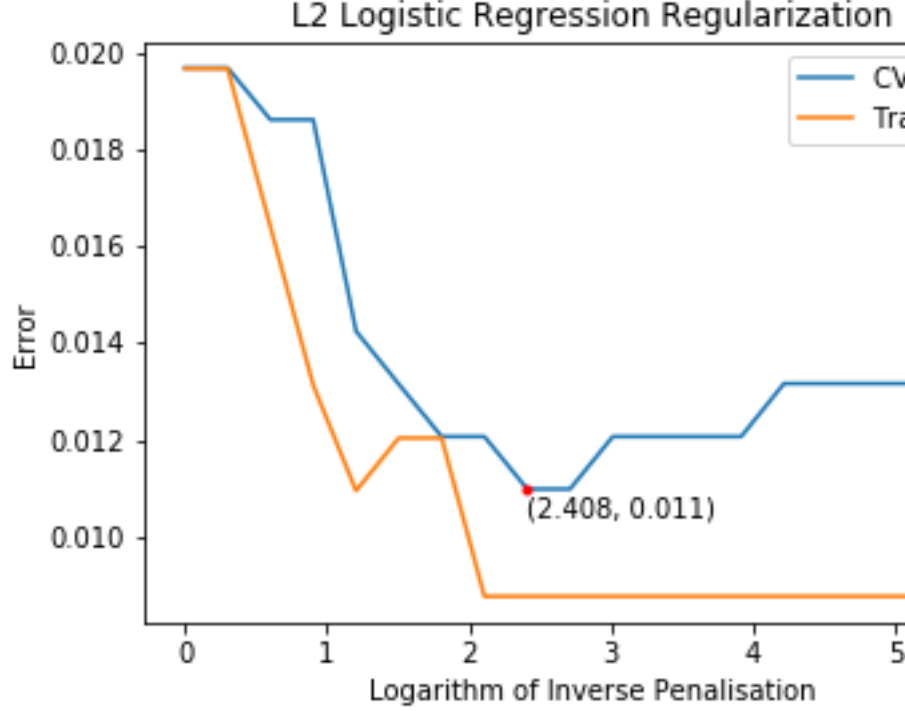
Suppose we are given a dataset $\{\mathbf{x}_i, y_i\}$ where now $y_i \in \{-1, 1\}$ and $\mathbf{x}_i \in \mathbb{R}^r$ for $i = 1, \dots, n$. Given our hypotheses, we can write the negative log-likelihood function as

$$\sum_{i=1}^n \ln(1 + \exp\{-y_i (\beta' \mathbf{x}_i)\}). \quad (9)$$

To this term we added a L2 penalization term to obtain a more regularized result, preferring this to its L1 version as the latter usually provides sparser solutions and we were working in an environment with only four covariates. This lead to the following problem:

$$\min_{\beta} f(\beta) = \frac{1}{2} \beta' \beta + C \sum_{i=1}^n \ln(1 + \exp\{-y_i (\beta' \mathbf{x}_i)\}), \quad (10)$$

where $C > 0$ is a parameter that can be assigned in order to balance the two terms. To tune it, we relied on a stratified 5-fold cross-validation on the training set, picking the value that yielded the lowest value. From a numerical point of view, the trust region Newton method built in the `ScikitLearn` class for L2-logistic regression was used.



C value - L2 Logistic Regression.png

2.2 k-Nearest Neighbours

Suppose that we have some classes \mathcal{C}_k such that each class contains N_k points and let $N = \sum_k N_k$ be the total number of points. Let us suppose that we are also given a distance function $\rho : \mathbb{R}^r \rightarrow \mathbb{R}^+$. To classify a new data point \mathbf{x} , we draw a sphere (according to ρ) such that exactly k points, regardless of their class, fall into it. If we let V be the volume of such sphere and K_k be the number of points of class k contained in it, we see that

$$\mathbf{p}(\mathbf{x}|\mathcal{C}_k) = \frac{K_k}{N_k V}, \quad (11)$$

the marginal probability is given by

$$\mathbf{p}(\mathbf{x}) = \frac{K}{NV} \quad (12)$$

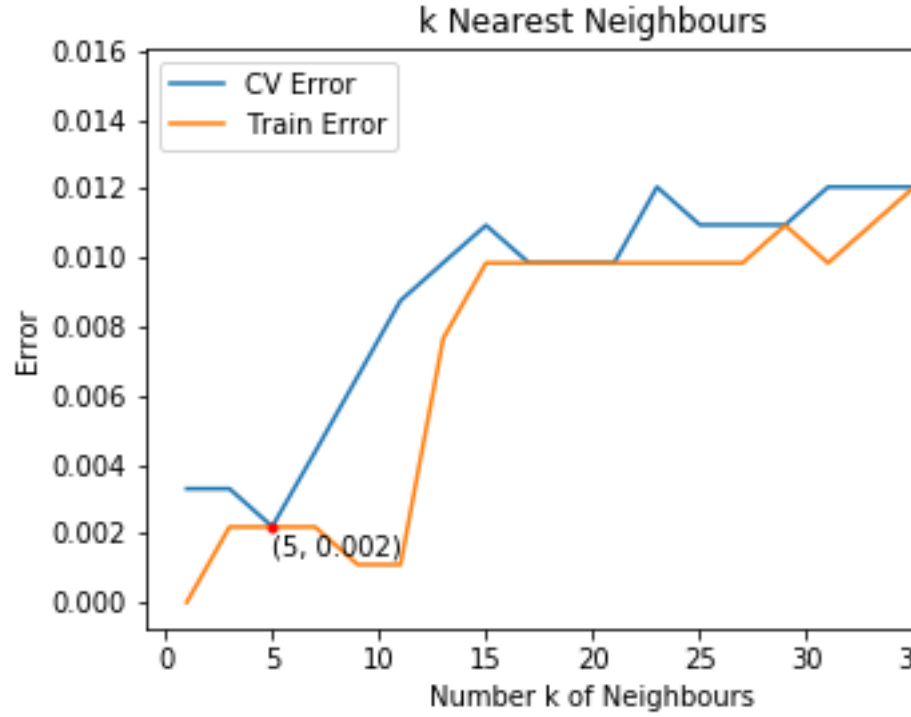
and the prior distributions are

$$\mathbf{p}(\mathcal{C}_k) = \frac{N_k}{N}. \quad (13)$$

Therefore, applying Bayes' theorem we have

$$\mathbf{p}(\mathcal{C}_k|\mathbf{x}) = \frac{\mathbf{p}(\mathbf{x}|\mathcal{C}_k) \mathbf{p}(\mathcal{C}_k)}{\mathbf{p}(\mathbf{x})} = \frac{K_k}{K}. \quad (14)$$

Comparing the posterior probabilities, we classified each data picking the maximum *a posteriori*. We consider the usual Euclidean distance for our distance function and trained only for odds value of k (between 1 and 39), picking the one that yielded the best result on a Stratified 5-fold cross validation on our data set.



K value - K-nearest Neighbours.png

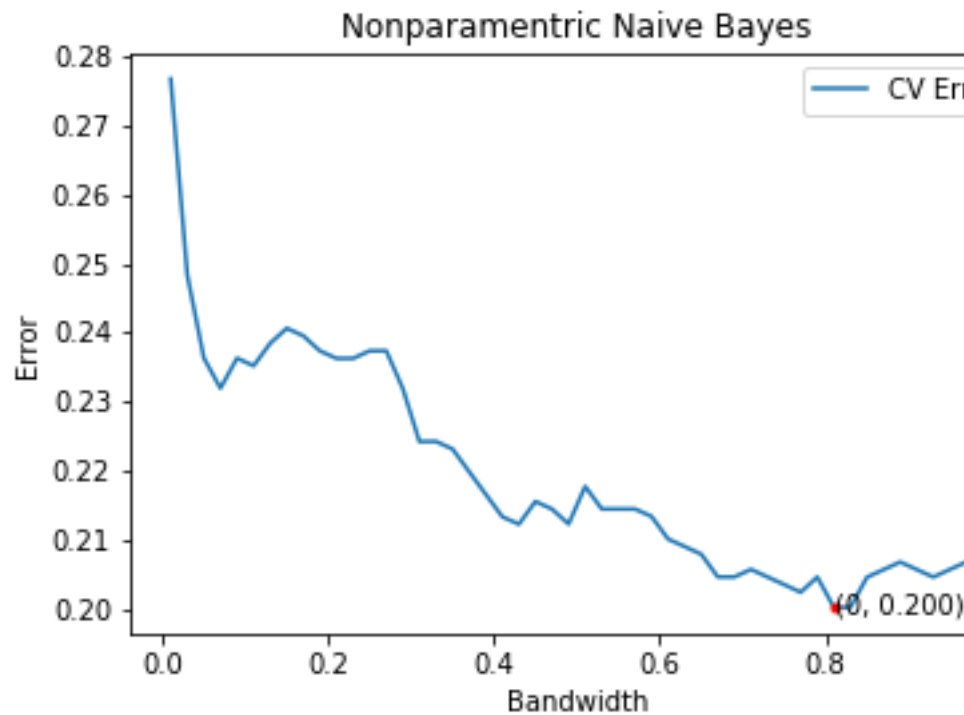
2.3 Kernel Naive Bayes Classifier

Let C_k be a class and let \mathbf{x} our data vector. As done before, we can apply Bayes' theorem to provide the *a posteriori* probability of belonging to such class by stating 14. Naive Bayes' classifier is based on the hypothesis that the attributes of our data are conditionally independent, allowing us to rewrite the conditional likelihood as

$$p(\mathbf{x}|C_k) = \prod_{i=1}^r p(x_i|C_k). \quad (15)$$

This assumption, that is in fact *naive*, does not usually hurt too much as we are interested in picking the maximum probability *a posteriori*.

Naive Bayes is usually implemented by considering a probability density function, that is fitted to the feature being taken into account. However, we decided to model the conditional distribution with a more complex kernel density estimation to allow for more parameter freedom. As far as the prior are concerned, we decided to consider the relative frequencies of the two classes.



Bandwidth - Naive Bayes.png

Explain what are the three parameters that were optimized and their effects on their respective classifiers. This will also require a brief explanation of how each classifier works.

Explain the method by which the optimal values were found, noting the differences between the errors and the importance of leaving out a test set for the final evaluation

The report should show the error plots but no other plots are necessary

3 Comparing Classifiers

After parameterizing and fitting our train data into our classifiers we used our test data to calculate the true error of all three classifiers.

3.1 True Error

3.1.1 Logistic Regression

True Negative: 253

False Positive: 1

False Negative: 2

True Positive: 201

Test Error: 0.0066

3.1.2 K-nearest Neighbours

True Negative: 254

False Positive: 0

False Negative: 0

True Positive: 203

Test Error: 0.0000

3.1.3 Naive Bayes

True Negative: 240
False Positive: 14
False Negative: 73
True Positive: 130
Test Error: 0.1904

For comparing the three classifiers performances, we used McNemar's test with a 95% confidence interval.

3.2 McNemar's test

Let e_{01} be the number of examples the first classifier misclassifies but the second classifies correctly, and e_{10} be the number of examples the second classifier classifies incorrectly but the first classifier classifies correctly. The difference divided by the total follows approximately a chi-squared distribution with one degree of freedom:

$$\frac{(|e_{01} - e_{10}| - 1)^2}{e_{01} + e_{10}} \approx \chi_1^2 \quad (16)$$

From the predicted classification values of two classifiers we were able to calculate the e_{01} and e_{10} values by doing a simple loop and comparing both classifications with the real classification value:

Algorithm 1: McNemar's test

Data: first predictions, second predictions, real values

Result: McNemar's test

```
1 initialize e01 and e10;
2 for every value in real values do
3   get current value of first predictions;
4   get current value of second predictions;
5   if |currentFirstPrediction - value| equals 1 and second prediction - value equals 0
6     then
7       increment e01;
8   if first prediction - value equals 0 and |currentSecondPrediction - value| equals 1
9     then
10      increment e10;
11 return  $\frac{(|e_{01} - e_{10}| - 1)^2}{e_{01} + e_{10}}$ 
```

After running the McNemar's test to compare all three classifiers we got this result:

Logistic Regression VS K-nearest neighbours: 1.333
Logistic Regression VS Naive Bayes: 78.284
K-nearest neighbours VS Naive Bayes: 85.011

Which indicates that Logistic Regression and K-nearest neighbours classifiers have similar performance.

TODO compare naive bayes

4 Discussion and Conclusion

This assignment aimed to help us understand the fundamental principles of three classifiers: Logistic Regression, K-nearest Neighbours and Naive Bayes. It also allowed us to understand and apply some data management techniques such as preprocessing data (Load, Shuffle, Split) and

the process of Cross Validation. The first step on this assignment was to load and preprocessing the data from a file which held data from the UCI machine learning repository about bank notes authenticity. The goal of the three classifiers was to predict either a bank note was real or fake based on four features (Variance, Skewness, Kurtosis, Entropy). A test of performance of those classifiers was done to compare and eventually choose the best and to discuss if any is significantly better than the others. From the McNemar's tests we noticed that Logistic Regression and K-nearest Neighbours classifiers performances were similar. But in this particular case, Naive Bayes performance was way worse than the other two classifiers.

5 Bibliography

- Lecture 5, Lecture 5 Notes, Lecture 6, Lecture 6 Notes, Lecture 7, Lecture 7 Notes
- Tutorial Notes
- Scikit-learn Preprocessing docs
- Scikit-learn Supervised Learning docs
- Scikit-learn Model Selection docs