Linear Classification

Guillaume Wisniewski
wisniews@limsi.fr

January 2017

1 Binary Classification

1.1 Provided Functions

In this lab, all examples are represented by a pair (label, observation) in which the observation is vector of reals. Functions to load all the datasets are provided on the lecture web site.

1.2 Toy Dataset

In this lab, we will implement the perceptron algorithm to solve binary classification problems. To test our implementation we will use a small and easy dataset, the Congressional Voting Records Data Set.[1]

This task aims at predicting whether a U.S. House of Representatives Congressmen is a Republican or a Democrat based on their votes for (or against) 15 key decisions.

The dataset is made of 435 examples. We will consider 100 examples as a test set. These examples have to be chosen randomly, for instance by using the following code:

```python
import random

# load the dataset as a list
data = ...
random.shuffle(data)
```

A function to load the dataset is available on the lecture website. This method returns a list of pairs (label, observation). The labels is either 1 (Democrat) or -1 (Republican); observations are instances of `numpy.array`. The function add a constant feature.

1. Describe two tasks that can be formalized as binary classification problem and identify what kind of features can be used.

2. Why do we add a constant feature?

[1] https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records

In a binary classification problem, a perceptron can be represented by a single vector that has as many components as there are features.

3. Write a method `classify` that takes as input an observation and a parameter vector and return the label associated to an observation.
4. Write a method `test` that takes as input a dataset and a parameter vector and compute the error rate achieved on this corpus.
5. What is the error rate achieved on the test corpus when the parameter vector is: `[25, -12, 67, -104, -43, 46, -18, -10, 45, -33, 54, -39, 43, -19, 5, -2, 55]`

We will now implement the training method. In the following, we will assume that the number of epochs (ie. how many time the trainset is considered) is a parameter specified by the user. Recall that, in case of error, the update rule is:

$$\mathbf{w} \leftarrow \mathbf{w} + y \cdot \mathbf{x} \tag{1}$$

where y is the gold label and $\mathbf{x}$ the observation.

6. Implement the `learn` learn method.
7. Represent the error rate on the train and test set with respect to the number of iterations. What can you conclude?

1.3 Stopping criterion

Achieving good performance relies on our capacity to stop training at the right time. Several stopping criterion can be used:

- iterates as long as there are errors on the training set;
- iterates for a fixed number of epochs;
- iterates as long as the error on a *validation set* decreases (the validation set has to be different from the train and test set).

8. Will we always be able to find a parameter vector that correctly classify all examples of the training set?
9. Why does the validation set have to be different from the train and test sets?
10. Implement and compare these three methods.

We will now consider the spam dataset available at the url `http://archive.ics.uci.edu/ml/datasets/Spambase`. This corpus contains 4,601 mails described by 57 features and labeled either as spam (-1) or non-spam (+1). Methods to read the dataset are available on the lecture web site. As previously, this method add a constant feature. The train set is made of the first 3,500 examples, the test set of the remaining examples.

11. What is the best performance that can be achieved on this dataset?