IN4320 MACHINE LEARNING

Exercises: Semi-Supervised Learning

Author:
MILAN NIESTIJL, 4311728

March 29, 2017

a.

In these exercises, two different methods of doing semi-supervised learning on a two-class LDA classifier are investigated. We now describe both methods on an algorithmic level. Recall that the probability density function in two-class LDA is given by:

$$f(x, y|\pi_1, \pi_2, \mu_1, \mu_2, \Sigma) = \pi_1 \mathcal{N}(x|\mu_1, \Sigma)\mathbb{1}_{y=1} + \pi_2 \mathcal{N}(x|\mu_2, \Sigma)\mathbb{1}_{y=2}$$

Where $\pi_1, \pi_2 \in [0, 1] : \pi_1 + \pi_2 = 1$ and $\mathcal{N}(x|\mu, \Sigma)$ corresponds to the probability density function of a normal distribution with mean μ and covariance Σ.

Supervised LDA

The maximum likelihood solution of the supervised problem can be shown to be given by:

$$\pi_i = \frac{1}{N}\sum_{n=1}^{N} x_n \mathbb{1}_{\{y_n=i\}}$$

$$\mu_i = \frac{\sum_{n=1}^{N} x_n \mathbb{1}_{\{y_n=i\}}}{\sum_{n=1}^{N} \mathbb{1}_{\{y_n=i\}}}$$

$$\Sigma = \frac{1}{N}\sum_{n=1}^{N} (x_n - \mu_{y_n})(x_n - \mu_{y_n})^T$$

Where x_i and y_i denote the feature-values and label of the i^{th} training sample.

Self-Training

The first method of extending the supervised learner to a semi-supervised setting is called 'Self-Training', which first fits LDA using only the labelled data and then iteratively assigns the predicted label to unlabelled data if the confidence is bigger than a certain threshold (set to 0.7). In pseudo-code:

```
def fit_with_Self_training(X,y, max_iterations=100, treshold=0.7)
    counter = 0
    repeat {
        fit LDA on labelled data
        if (counter==max_iterations or no unlabelled data){
            break
        }
        predict labels of unlabelled data
        foreach point in unlabelled data do {
            if (confidence>treshold):
                label point as predicted
        }
    }
```

Label-Propagation

The second method is called 'Label-propagation' (Zhu & Ghagramani, 2002), which first defines a graph on the data by specifying the weights of all edges. There are multiple ways to do this, but in our case the weight w_{ij} is given by $w_{ij} = \mathbb{1}_{kNN(x_j)}(x_i)$. That is, the weight of the edge connecting point i to point j is 1 if i is one of the k-nearest neighbours of j and 0 otherwise. Next, a transition matrix T is defined by

$$T_{ij} = \mathbb{P}(j \to i) = \frac{w_{ij}}{\sum_k w_{kj}}$$

We define a label matrix Y, where the i^{th} row represents the probability distribution over the different classes for the i^{th} data point. The algorithm iteratively calculates the label matrix after letting the labels propagate according to the transition matrix, while fixing the distribution over the different classes for the labelled samples throughout the process. The propagation algorithm is shown below:

```
repeat untill convergence {
    1. propagate:  $Y = TY$
    2. Row-normalize $Y$.
    3. Clamp the labelled data:  $Y_{ic} = \delta(y_i, c)$
}
```

So that the corresponding semi-supervised LDA algorithm is given by the following pseudo code:

```
def fit_with_LabelPropagation(X,y, treshold=0.7)
    newLabels = labelPropagation(X,y)
    LDA.fit(X,newlabels)
```

b. and c.

The algorithm is tested on set `Spambase Data Set` from the UCI repository[1]. First the features are standardized, i.e., they are rescaled to have unit variance. As training data, a subset of the dataset is used, containing 75 random samples from each class and a varying number of unlabelled data points. We test the various implementations on the training data. Repeating the experiment a total of 50 times, the mean of the training error and the log-likelihood of the labelled data is plotted against the number of unlabelled samples in the training data. The corresponding standard deviations are shown as error bars. The result can be seen in figures 1 and 2. We see that the error on the training data increases for increasing number of unlabelled samples. This is not necessarily unexpected, as this may happen if the assumptions of the model are not valid. That is, the data may not be well modelled by two Gaussian distributions (with equal covariance).

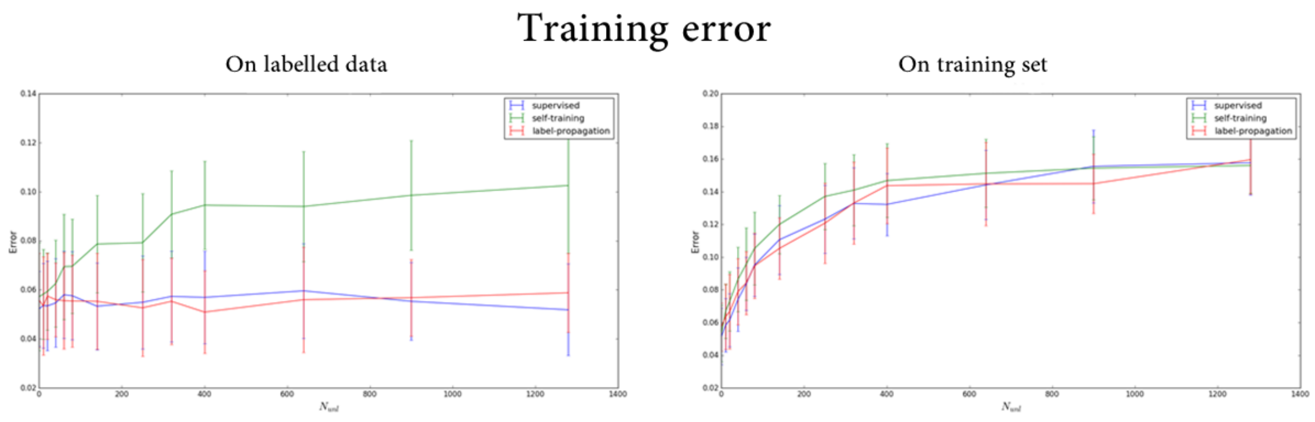

Figure 1: Error on the training data versus number of unlabelled samples, for various implementations of semi-supervised LDA. The left image shows the error on only the labelled data whereas the right image shows the error on the entire training set

[1]See http://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.data

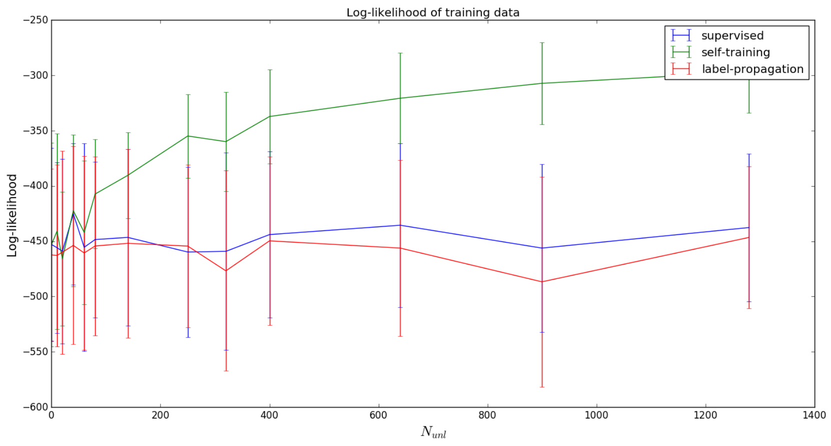

Figure 2: Log-likelihood of the labelled samples in the training data versus number of unlabelled data points, for various implementations of semi-supervised LDA.

d. We construct two datasets, such that the performance of one of the two methods for semi-supervised learning is worse than the supervised learner.

Let $C \sim Ber(\frac{1}{2})$, $Z_1 \sim Ber(0.3)$ and $Z_2 \sim\ Ber(0.2)$. Let X be a random variable distributed as:

$$X \sim \left[\mathcal{N}([3,5], I_2)\mathbb{1}_{\{Z_1=0\}} + \mathcal{N}([-1,2], I_2)\mathbb{1}_{\{Z_1=1\}}\right]\mathbb{1}_{\{C=0\}}$$
$$+ \left[\mathcal{N}([5,2], I_2)\mathbb{1}_{\{Z_2=0\}} + \mathcal{N}([12,-8], I_2)\mathbb{1}_{\{Z_2=1\}}\right]\mathbb{1}_{\{C=1\}}$$

A dataset drawn from the first custom distribution, along with the predictions of the various methods is shown in figure 3. It can be seen that the error on the training data gets significantly worse for the self-training algorithm, whereas the label-propagation method performs similar to the supervised method. This is made even more clear in figure 4, where the training error is plotted versus the number of unlabelled data.

The self-training algorithm iteratively labels unlabelled samples red in the two red clusters. The same can't be done for the blue labels that are near the red cluster as the confidence is not high enough. Eventually, the red labels start to dominate causing the entire blue cluster to be predicted as red labels. The label-propagation algorithm on the other hand, also labels the unlabelled samples in the upper blue cluster blue before finally fitting the LDA, as the labels are propagated to the k nearest neighbours for each point. This explains the difference in performance.

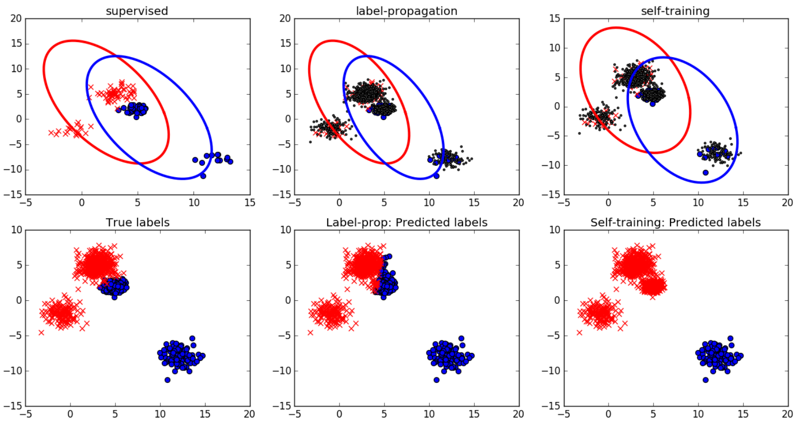

Figure 3: Scatter plot of dataset or predicted labels for various methods of semi-supervised LDA. The ellipses represent the covariance matrices of the normal distributions, centered around the corresponding mean.

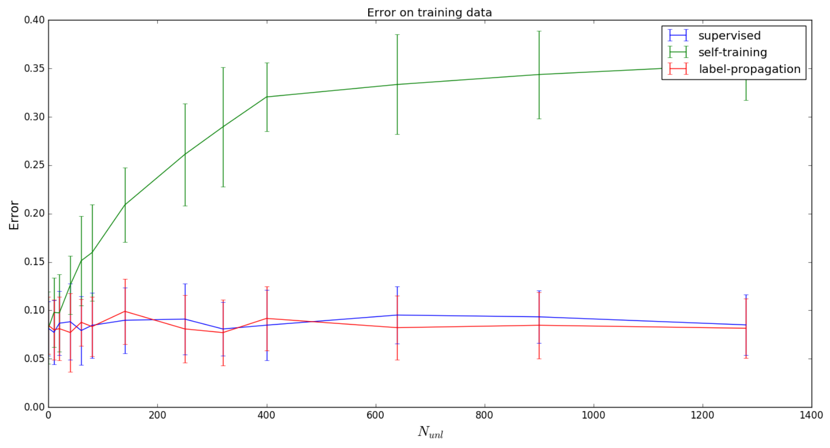

Figure 4: Training error on the first custom distribution versus number of unlabelled samples, for various implementations of semi-supervised LDA.

A dataset drawn from the second distribution is plotted in figure 5. It can be seen that the self-training algorithm performs significantly better than the other two. This is further verified in figure 6. This is explained by the fact that the problem is not linearly separable. The label-propagation algorithm labels all the samples correctly in the propagation step, but then tries to fit two-class LDA, which inherently results in poor performance due to the nature of the dataset. On the other hand, the self-training algorithm iteratively labels the unlabelled samples red but does not label as many data points in the right-most cluster blue due to a lack of confidence. Therefore, eventually the blue cluster on the right is also predicted red, which results in all red samples being predicted correctly.

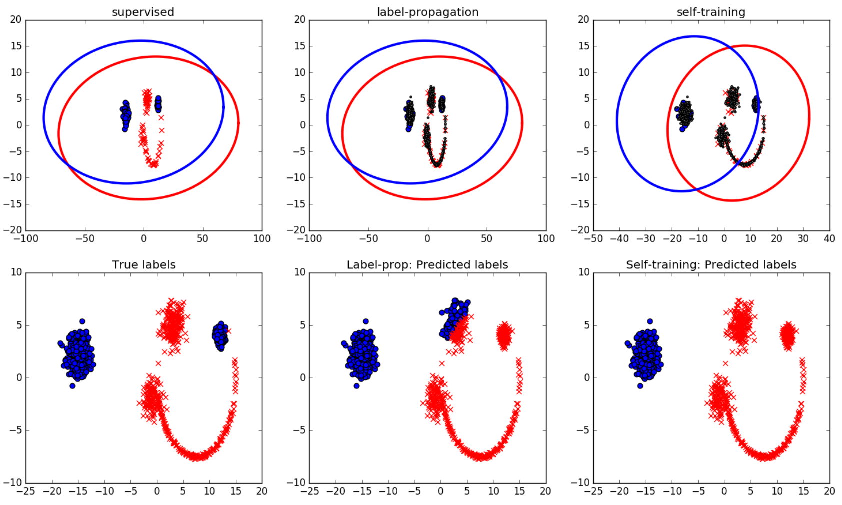

Figure 5: Scatter plot of dataset or predicted labels for various methods of semi-supervised LDA. The ellipses represent the covariance matrices of the normal distributions, centered around the corresponding mean.

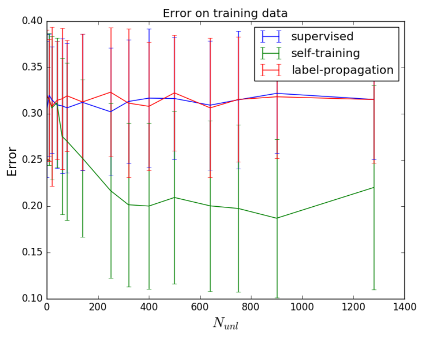

Figure 6: Training error on the second custom distribution versus number of unlabelled samples, for various implementations of semi-supervised LDA.

Bibliography

Zhu, Xiaojin, & Ghagramani, Zoubin. 2002. Learning from Labeled and Unlabeled Data with Label Propagation. *CMU-CALD-02-107*.