IN4320 Machine Learning

Exercises: Semi-Supervised Learning

Author:
Milan Niestijl, 4311728

March 27, 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
        }
    }
    self.covariance, self.means = result (TODO)
```

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

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

Furthermore, 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 propagation algorithm is shown below:

```
repeat untill convergence {
    1. propagate:  Y = TY
    2. Row-normalize Y.
    3. Clamp the labelled data:  Y_ic = δ(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)
    self.covariance, self.means = result (TODO)
```

Bibliography

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