IN4320 Machine Learning Computer Exercise
ECorpChallenge

Dilan Gecmen 4221168

August 6, 2020

Estimate of words: 1100

1 Datasets

The problem is a two-class classification problem in 204 dimensions. The small labeled training set has a size of 50 samples per class. The unlabeled data provides 20,000 additional samples. These samples have undergone a corruption. The corruption is additive uniform noise, which comes from a p-norm ball with radius r:

$$||x||_p \leq r \tag{1}$$

The norm p and the radius r are both unknown.

2 Noise

As the training set is not corrupted with noise an idea is to corrupt it with the same additive noise as from the test set. In literature the following method is given to generate uniform samples in p-norm balls with an unknown radius for probabilistic robustness analysis [1]:

Algorithm for real uniform generation
Given n, p, and r, the algorithm returns a real random vector y which is uniformly distributed in $B(r)$.

1. Generate n independent random real scalars

$$\eta_i \sim \hat{G}(\frac{1}{p}, p)$$

.

2. Construct vector $\mathbf{x} \in \mathbb{R}^n$ of components $\mathbf{x}_i = s_i \eta_i$, where s_i are independent random signs.

3. Generate $\mathbf{z} = \mathbf{w}^{\frac{1}{n}}$, where $\mathbf{w}$ is a random variable uniformly distributed in the interval [0,1].

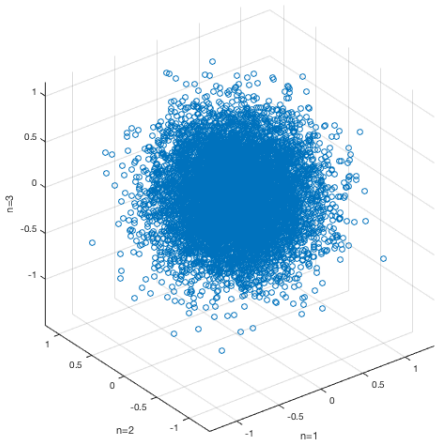
4. Return $\mathbf{y} = r\mathbf{z}\frac{\mathbf{z}}{||\mathbf{x}||_p}$

Here $\hat{G}$ is the generalized gamma distribution [1]:
A random variable $\boldsymbol{x} \in \mathbb{R}$ is generalized gamma distributed with parameters $(a, c), \boldsymbol{x} \sim \hat{G}(a, c)$, when it has density function

$$f_{\mathbf{x}}(x) = \frac{c}{\Gamma(a)} x^{ca-1} e^{-x^c}, \quad x \geq 0 \tag{2}$$

For illustration the noise is generated for 10000 samples in 3 dimensions:

Figure 1: Noise generation example



Note that we need to generate noise for 200 samples in 204 dimensions. This means that we get a 200×204 noise matrix and add this to the 200×204 training matrix.

3 Method 1: Label Propagation

An approach to label the large test set with a small labeled data set is to use Semi-Supervised Learning. So, we make use of unlabeled data for training.

1. Set values for p and r.
2. Create noise using the algorithm from Section 2.
3. Add noise to the training set.
4. Apply cross validation. Split the the training set in 80% training and 20% validation.
5. Add q random unlabeled points from the test set to the training set and apply label propagation [2] to label the unlabeled points:
   (SEMI SUPERVISED LEARNING).
6. Combine the new labeled points and the 80% training points to a new training set.

7. Using the new training set apply label propagation again to the validation set and evaluate the error.

8. Optimization: Minimize the error by varying for different values for p and r using particle swarm.

9. Use the optimized p and r to predict the labels of the remaining test samples in the test set. $\Rightarrow$ The smaller the error, the better the estimated noise (which was added to the training set). In other words, the generated noise looks like the noise with which the test set is corrupted.

For this assignment, I got the highest accuracy for p=2 and r=1 with adding q=1000 unlabeled points from the test set to the (80%) noised training set and applied label propagation. After that I used this new set to predict the labels of the validation set. This gave me an accuracy of: **56%**. The accuracy is not what we expected, so we try PCA and look if it influences the accuracy.

3.1 Principal Component Analysis (PCA)

To get even better results Principal Component Analysis is considered to reduce the large dataset of variables to a smaller dataset. This smaller dataset still contains most of the information in the larger dataset. In other words, the number of correlated variables is transformed to a smaller number of variables that is uncorrelated, which are called the principal components. The most variance is accounted by the first principal component. For succeeding components this amount decreases. Hence, the factors with the lowest eigenvalues can be seen as redundant.
The eigenvalues of the covariance matrix of the train and test set are plotted in Figures 2 and 3. When we look at Figure 3 we see clearly that three components can be removed from the set. This reduces the dimensionality from 204 to 201. We do the same to the training set.
Using PCA to reduce the dimensions and afterwards apply the method described in Section 3 did not yield in good results.

Figure 2: Eigenvalues from the covariance matrix from the training set plotted

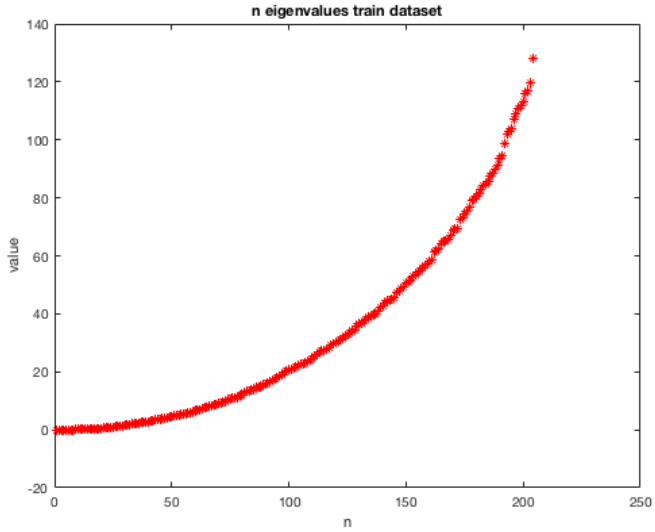
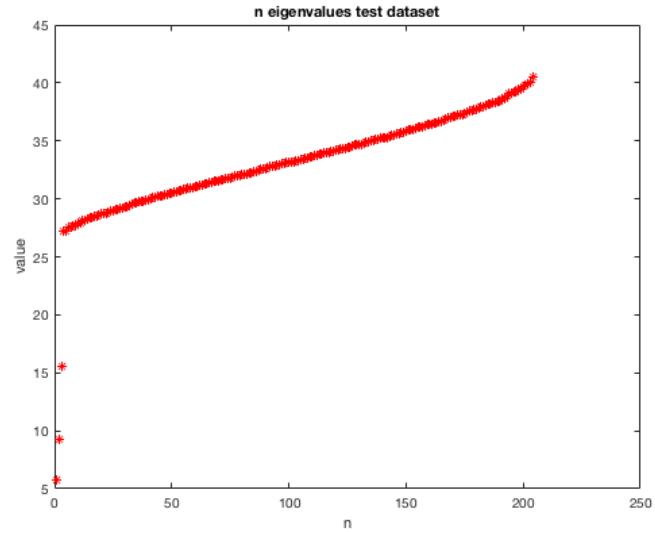


Figure 3: Eigenvalues from the covariance matrix from the test set plotted



4 Method 2: Support Vector Machine

The accuracy found using label propagation did not yield in good results. Combining label propagation with PCA also did not do the trick. Hence, I tried another approach which involved using support vector machine [3].

1. Set p=2 and r=1.
2. Create noise using the algorithm from Section 2.
3. Add noise to the training set.
4. Classify the unlabeled points using Support Vector Machine.

Using the described method yields an accuracy of 75%. Using cross validation did not improve the accuracy and applying PCA also did not do the trick.

5 Discussion

The achieved accuracy for the best classifier SVM is 75%. The accuracy found with label propagation is 25%, which indicates that the approach to estimate optimal values for p and r did not work. This means that the noise added to our training set does not resemblance the noise with which the test set is corrupted. Using a better technique to optimize these values would certainly increase the accuracy, as a semi supervised approach to this assignment is the approach that should work.
Principal Component Analysis or Singular Value Decomposition are not of use in this assignment as there is no benefit in reducing from 204 to 201 dimensions.

References

1 G. Calafiore, F. Dabbene and R. Tempo. Uniform Sample Generation in l_p Balls for Probabilistic Robustness Analysis. Politecnico di Torino, Italy, 1998.

2 M. E. J. Newman, Detecting community structure in networks.

3 S. Ding, Z. Zhu, X. ZhangAn. Overview on semi-supervised support vector machine. Neural Computing and Applications, 2017, Volume 28, Number 5, Page 969.