Shane Kopec

Professor Ho

Machine Learning

3/25/2022

Scikit: Metric Learning Algorithms

1. Introduction

A Significant portion of the machine learning methods require the measurement of the distance of points. Mathematicians and engineers alike have proven that there are many ways to calculate the distance too. This can be seen in methods such as Euclidean, City-Block (Manhattan Distance), and cosine distance metrics. These methods work great for two-dimensional data, (points with only a x and y). However, not all data is created uniformly, there can sometimes be some noise in the data. Noise within data is when the data has not as useful information attached to the useful data. Metric-Learn is a python library that is part of the larger scikit-learn-contrib library, a massive machine learning library with method implementations as well as datasets.

Of the supervised metric learning methods in my experiment, I will cover Large Margin Nearest Neighbor (LMNN), Neighborhood Components Analysis (NCA), and Local Fisher Discriminant Analysis (LFDA). The first one, Large Margin Nearest Neighbor, was mainly designed for a k-nearest neighbor classification. What makes it different from KNN is the addition of a pseudo metric that can help improve the classification accuracy.(Vazelhes) The next method is the Neighborhood Components Analysis. NCA uses linear transformations to help increase the accuracy of a rule that is close to k nearest neighbor. This rule is stochastic nearest neighbor. The last method covered within my experiments is the Local Fisher Discriminant Analysis. This method is a linear supervised dimensionality reduction method. (Vazelhes) Meaning it is good at removing unnecessary and noisy dimensions to reach a higher accuracy.

2. Metric-Learn Methods

To first get an understanding of these new Metric-Learn methods, I implemented them using a synthetic data set that was supplied and created through the “sklearn.datasets” import. With this I can create data with any number of classes, clusters per class, and how many classes are informative versus noise. This is to help simulate a higher dimensionality and allows me to test how Metric-Learn can handle noise.

Figure 1

In figure one, we see the data as it was generated. The three different classes were generated into two different cluster. From here we can now use the different metric learning methods on the data.

The method graphed is the Large Margin Nearest Neighbor (LMNN). It is evident that this method works fine with the data set did a clean division of the two different clusters. With no overlap between any of the colors, we are able to see a distinct separation between any of the different classes.

Figure 2(LMNN)

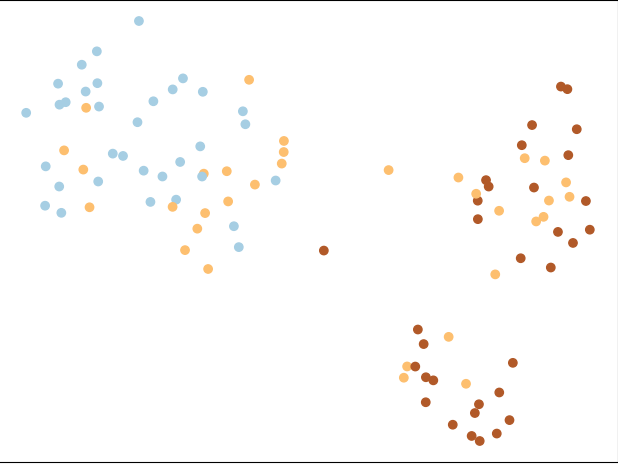
The third figure shows the NCA method. Having only had time to test the Neighborhood Component Analysis on the synthetic dataset. The goal of this set is a tiny different than the other two, NCA wants to improve the accuracy compared to that of standard Euclidean. The others mainly want to cluster similar class data points and create margins between different classes. The metrics created from this method seem to be the weakest of all the methods. Upon further investigation that will be discussed later in my report we can see that this method struggles when there is too much noise.

Figure 4(LFDA)

Figure 3(NCA)

Figure four shows the final Metric-Learn method that I used within my experiments. The method, Local Fisher Discriminant Analysis, separated the data into visible clusters by class. However, the clusters in LFDA appear to be closer than those in the original LMNN method. This could be due to the randomly generated data set or because of how the methods classify the data. I will continue to compare these methods over many iterations of data.

3. Pipelines & GridSearchCV

Metric learn excels at creating task specific distance metrics for k-NN classification. The goal of my upcoming experiments to see the difference in accuracy for k-nearest neighbor classification with and without metric learn. This can be achieved using pipelines from sklearn. The pipeline will send the new data metrics made from the input data to a k-nearest neighbor classifier. Each pipeline allows for one estimator to be created with the exact parameters for each metric method and k-nearest neighbor. The problem is there are many combinations of parameters that need to be tested.

GridSearchCV solved all the problems for testing many parameters. GridSearchCV allows for me to feed in a list of parameters for each step on the pipeline, even multiple parameters per method in need. This hyperparameter will check every possible combination between the sets on parameters given. Returning the accuracy & parameter settings that gives the highest testing score across the data. GridSearchCV also has a built-in feature for k-fold cross validation allowing for me to make sure my training and testing result can be accurate. This method was applied to each of the three metric learn methods talked about in this paper so far. I will compare the testing accuracies of all methods and a control k-nearest neighbor only set.

Each combination will also be tested on two different types of data, 10 times each, synthetic and real. For the synthetic data I will be using the method provided from sklearn. This allowed for me create a set of 1000 samples, with three informative features and two noise feature per sample. For the real-world data, I will be using the wine data set also provided from the sklearn.datasets import. This was a great set to test these metric methods on because there are twelve features per sample and three possible classifications.

4.1 Results Synthetic Data Size 1000

Overall, the results that we received from each method were very accurate. The first method tested was the Least Margin Nearest Neighbor (LMNN) to K-Nearest Neighbor (KNN). The results can be seen in figure 1 in appendix. The hyperparameters for this GridSearchCV were LMNN\_K, and KNN\_K. The k for both methods is how many nearest neighbors to compare the point to. LMNN\_K consisted of one through five, while KNN\_K was one through eight. Across the ten iterations that were run with these methods we saw a range of testing accuracy from 96% to 99%. Eight of the ten runs had different hyperparameter settings. Since most iterations picked a different estimator to be the best each time it can be inferred that all combos give a relatively close score. Least Margin Nearest Neighbor took the longest to run out of any of the metric learn methods. So, in the future it could be possible to slim down the number of hyperparameters passed in without a significant drop in accuracy.

The next method that was tested was the Neighborhood Component Analysis (NCA) to K-Nearest Neighbor (KNN). The two parameters that needed to be passed in for this method are again KNN\_K as well as NCA\_MAX\_ITER. The number of max iterations for run for each method of NCA. When looking at the testing accuracies from the ten runs we see a range of 96% to 99% just like LMNN. Where NCA differs from LMNN is the frequency of each hyperparameter. Over the ten runs there were four different parameters. This is to be expected because only KNN\_K has multiple parameters. We see k equals one shows up four times, k equals five shows up three times, k equals three twice, and k equals six once. Some of the most accurate results show up from k equal one, so we can assume just like LMNN that we could slim down the number of hyperparameters to pass in.

The last but certainly not the least is Local Fisher Discriminant Analysis(LFDA) to KNN. This combination had the hyperparameters LFDA\_K and KNN\_K. The new one being LFDA\_K, this k stands for checking the k nearest points This method as stated before is a dimensionality reduction method, so it performed the best out of any method. In figure three we can see that the frequency of the hyperparameters within this combination is odd. Five of the runs were all one hyperparameter while the other 5 were all unique. Over the ten runs performed we had testing accuracies ranging anywhere from 98% to 100%. However, two out the ten runs were 100%, two were 99%, and six 98%. This concerned me a little as I thought there could be a possibility of overfitting. To solve this concern, I ran LFDA to KNN on a data set with 2000 samples instead of 1000. What we saw was the accuracy shot up 99% across all the runs(Figure 4). This boosted my confidence in the overall testing accuracy for metric-learn method.

Comparing each metric learning method results to the results of just using the KNN classifier is night and day. The average of all 10 runs of each method is displayed in figure 6. While every method sat around 96% to 97% average accuracy across ten runs, the KNN only runs struggled significantly more. With an average of about 79%, KNN only is around 18-19% less accurate. When working with data that you may know has some noise within its features, it would be very beneficial to incorporate a metric learning method on top the classification method as well.

4.2 Results Wine Dataset

After testing each method on a synthetic dataset, the next step was to try in on data that had not been generated by myself. The wine dataset was the one I chose because, there are many features per data point (12 features). This made it a great set to stress test the metrics acquired from the metric learning algorithms. The first combination that was tested was the Least Margin Nearest Neighbor (LMNN) to KNN, which results can be seen in figure 6. The testing accuracies had a range of 87% to 100%. Over the course of ten runs we saw an average of around 95% testing accuracy. The result is to be expected however, there may some overfitting due to the data set only have around 187 data points.

The next method that I tested out was the Local Fishers Discriminant Analysis (LFDA) to KNN. The results for this set are in figure 7, with a smaller accuracy range than the LMNN. It ranged from 92% to around 97%, with an average of 95% . Comparing this to the results of the synthetic data it would make since why we see lower testing accuracy scores. Even though the goal of this method is dimensionality reduction, it will still have more trouble when each sample has more features.

The last method to be tested on the wine data set was Neighborhood Component Analysis. The results from these tests can been read in figure 8, overall, these testing scores were the lowest. The accuracies over ten runs ranged anywhere from 57% to 82%. This was significantly lower than the results we saw back in the synthetic data. Even compared to the other methods which are looking to reduce the dimensionality (LFDA) or create larger margins in-between the classes (LMNN). My hypothesis for why we see such a decrease is because, the synthetic data had less features and less noise.

In figure 9 the averages of all three methods plus a control KNN only runs are displayed on the bar graph. Both LMNN and LFDA performed on par with the scores from the synthetic runs. Both having an average testing accuracy around 95%. The interesting thing is NCA to KNN underperformed KNN only. NCA was a detriment to the KNN’s accuracy. To further understand why, I tested my hypothesis that about feature size and noise.

4.3 Synthetic Dataset Size 1000 double the number of features

To get accuracy similar to those of the Wine dataset, I doubled the number of features, number of informative features, and number of noise features. Both LFDA and LMNN performed more precise and accurate than the previous synthetic data testing. Both methods ranged anywhere from 97%-99% (Figures 10 &11). The data that was received from the NCA to KNN supported my hypothesis (Figure 12). The more features included in the data, less accurate NCA becomes. The average run was somewhere around 82% but, there were some outliers above and below the average. Comparing this to the averages we saw in the first set of synthetic data which was about 97%.

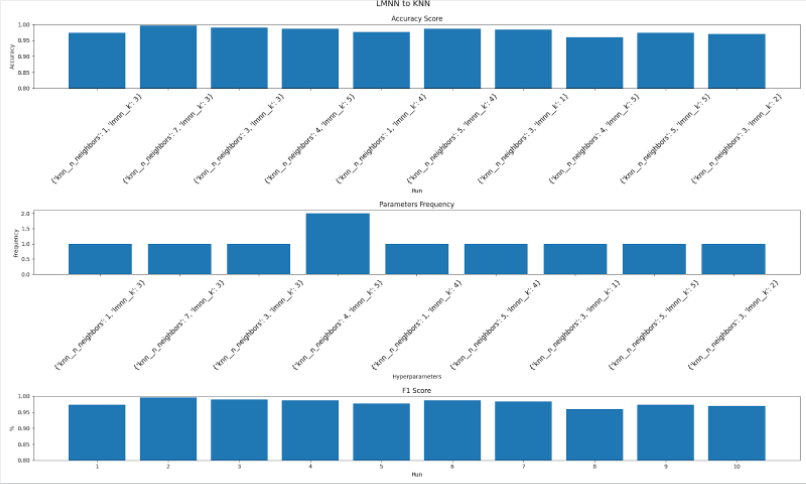
5. Conclusion

Overall, from every test, I was able to find a solid reason to use LMNN, LFDA, and NCA. Least Margin Nearest Neighbor excels at separating a cluster of points into smaller clusters by class. This can be seen in the fact that it was able to handle anything thrown out it with precision and accuracy. Local Fisher Discriminant Analysis works great for multimodality and data with noise features. This is evident because across all my experiments LFDA always had the highest average testing accuracy. As I gave it more features it only became more accurate. While LMNN and LFDA have a broader use case, there are still times where the Neighborhood Component Analysis is a viable option. In the first synthetic dataset with 5 features per sample, we see results just as precise and accurate as LMNN and LFDA. NCA also ran significantly faster than the other two methods. These metric-learn methods significantly increased the accuracy of the KNN classifier and can very valuable asset to any data scientist if used properly.

References

1. metric-learn: Metric Learning Algorithms in Python, de Vazelhes et al., Journal of Machine Learning Research, 21(138):1-6, 2020.
2. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Appendix A.

Figure 1. LMNN to KNN Synthetic Data

Timeline

Description automatically generated with low confidenceFigure 2. NCA to KNN Synthetic Data

Figure 3. LFDA to KNN Synthetic Data

A picture containing chart

Description automatically generated

A picture containing chart

Description automatically generatedFigure 4 LFDA to KNN Size 2000 Synthetic Data

Chart, bar chart

Description automatically generatedFigure 5 Synthetic Data Averages

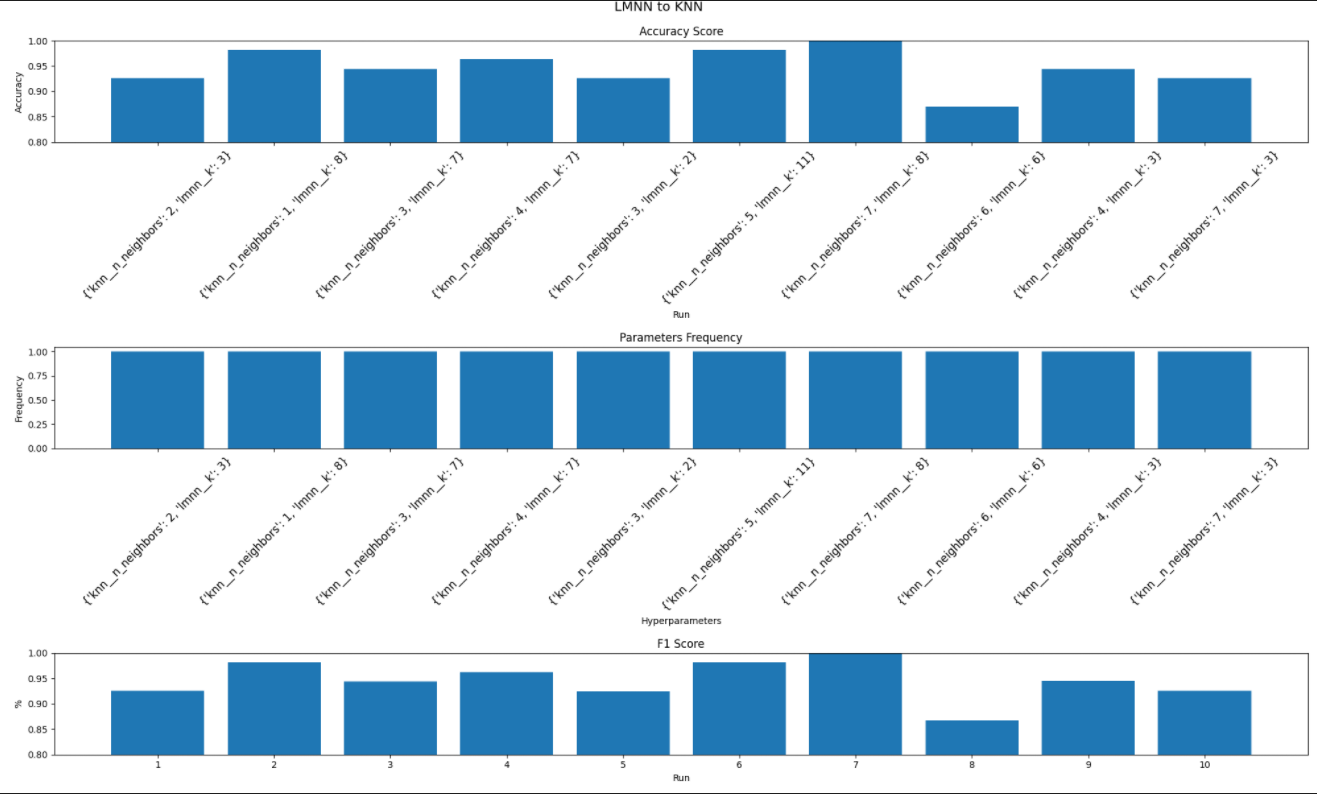
Figure 6 LMNN to KNN Wine Data

Figure 7 LFDA to KNN Wine Data

Chart

Description automatically generated with low confidence

A picture containing chart

Description automatically generatedFigure 8 NCA to KNN Wine Data

Figure 9 Wine Data Averages

A picture containing chart

Description automatically generatedChart

Description automatically generatedFigure 10 Synthetic LMNN 2x Features

Figure 11 LFDA Synthetic 2x Features

A picture containing timeline

Description automatically generatedFigure 12 NCA Synthetic 2x Features

A picture containing waterfall chart

Description automatically generated