UNIVERSITY OF WATERLOO

Literature Review of Neighbourhood Components Analysis & Implementation

ROLL #: 20854901

Rahij Imran Gillani

May 20, 2020

1 Introduction

K-Nearest Neighbours (KNN) is one of the most simple classification methods, and is also pretty effective. Many times if we don't know a problem well enough beforehand and we have to bet on a classifier, we are much better off betting on the simplest classifiers. The simple classifiers are mostly never the best classifiers for any given problem, but they perform consistently well. KNN is one of those simple classifiers, and the most appealing part of KNN's is that it forms a non-linear decision boundary; furthermore, there is only one parameter to tune which is 'k' (the number of nearest neighbours to take into account for classification) which can be determined through cross validation, and the quality of predictions also improve automatically with more training data.

However, KNN suffers from two main drawbacks.

- **Computational cost.**
  For predicting class of each input, the entire data-set must be stored and searched through. So if the data-set is very large then the computational cost is also very high.
- **What does 'nearest' mean?**
  What distance metric should be used?

Neighbourhood Component Analysis (NCA) targets both these shortcomings to improve the result and performance of KNN.

2 Distance Metric Optimization

We might think that it's easy to pick out the best distance metric if we're given a finite set of distance metrics, we could simply chose the best metric using the **Leave-one-out (LOO)** approach on the training data as we do not have access to the test data. However, what about when we're given a continuous set of distance metrics?

For a continuous set of metrics it is not feasible to optimize with respect to the actual LOO method for KNN, as it forms a quite discontinuous function. Hence, for a smoother differential cost function, instead of picking out a fixed number of neighbours, we pick out one neighbour randomly each time. However, all this is done in a transformed space, which is calculated by learning a linear transformation 'A' of the input space such that in the transformed space, the average leave-one-out (LOO) classification performance is maximized.

So basically we want to maximise our cost function with respect to a transformation 'A'. Specifically when we input a point 'i', it then randomly selects another point 'j' as its neighbour with some probability p_{ij}, and inherits the class label of 'j'.p_{ij} is defined using a softmax over Euclidean distances in the transformed space:

$$p_{ij} = \frac{\exp(-||Ax_i - Ax_j||^2)}{\sum_{k \neq i} \exp(-||Ax_i - Ax_j||^2)}, \quad p_{ii} = 0$$

The probability that p_i is correctly classified is:

$$p_i = \sum_{j \in C_i} p_{ij}$$

This is the function that we want to maximize, the expected number of points correctly classified:

$$f(A) = \sum_i \sum_{j \in C_i} p_{ij} = \sum_i p_i$$

After differentiating with respect to A and some reordering we get the objective function:

$$\frac{\delta f}{\delta A} = 2A \sum_i \left(p_i \sum_k p_{ik} x_{ik} x_{ik}^T - \sum_{j \in C_i} p_{ij} x_{ij} x_{ij}^T \right)$$

Now all that needs to be done for the NCA is to maximise the function above using a gradient based optimizer.

3 Low Rank Distance Metrics

The second shortcoming that NCA targets is the computational cost. NCA improves on that by restricting A to be a non-square matrix of size dxD, By making such a restriction and choosing $d << D$, NCA vastly reduces the storage and search-time requirements of KNN as there is a significant dimensionality reduction.

NCA does this by first finding the best transformation A, after which only the projections of the training points $y_n = Ax_n$ are saved in a much lower dimension The storage requirements of NCA is $O(dN) + Dd$ compared with $O(DN)$ for KNN in the original input space.

4 Task

We are now going to classify Human Activity based on collected sensory data. The task is to predict six activities (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING) based on signals extracted from an accelerometer and gyroscope of a smartphone, which is attached to the waist of a group of 30 volunteers within an age bracket of 19-48 years.

The dataset is one of the UCI datasets available at `https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones`.

5 Methodology (NCA)

K-NN is one of the simplest classification algorithms available for supervised learning. The idea is to search for the closest match of the test data in feature space.

For this task we will be using **Neighbourhood Component Analysis(NCA)** instance that learns the optimal transformation and then combined it with Weighted K-NN. NCA is attractive for classification because it can naturally handle multiclass problems without any increase in the model size, and does not introduce additional parameters that require fine-tuning by the user.

NCA classification works well in for data sets of varying size and complexity[2]. Compared to other Classification methods such as Linear Discriminant Analysis (LDA), NCA doesn't make any assumptions about the class distributions. Moreover, the NCA can naturally produce highly irregular decision boundaries.

6 Parameter Selection

6.1 Estimating Best 'k'

To find the best 'k' (number of neighbours) we reduce dimensionality using PCA and then measure accuracy using Vanilla K-NN without NCA.

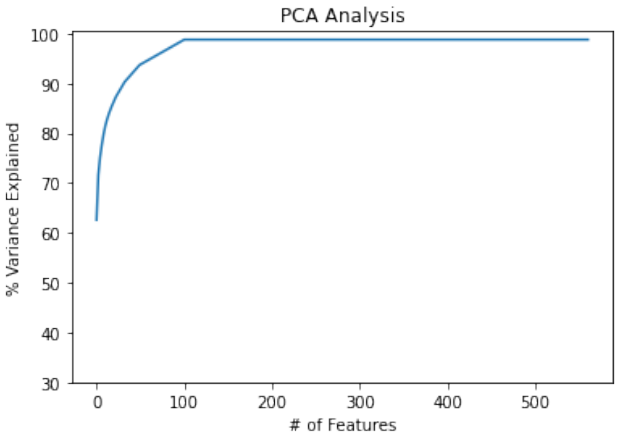

Figure 6.1: **99% of variance in data occurs due to first 99 components.**

After reducing dimensionality using PCA, the data is passed to **Vanilla Weighted K-NN** algorithm and goes through **10 times 10 folds** cross validation on the train data for different values of 'k'.

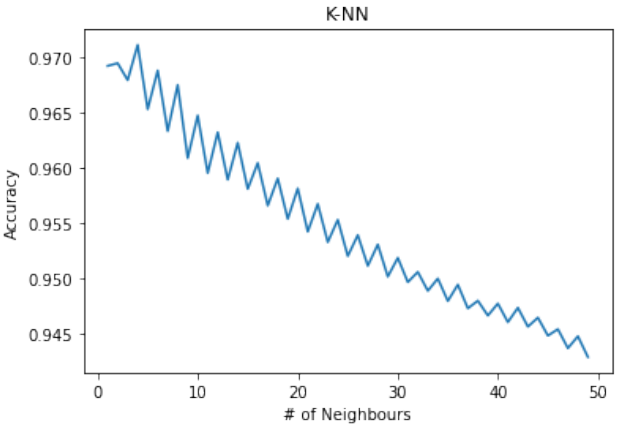

Figure 6.2: **Maximum accuracy is 0.971 with 4 neighbours.**

6.2 Reducing Dimensions Using NCA

Now we want to reduce the dimensions of the dataset using NCA, so we test accuracy on test data for different number of dimensions with k=4 for K-NN.

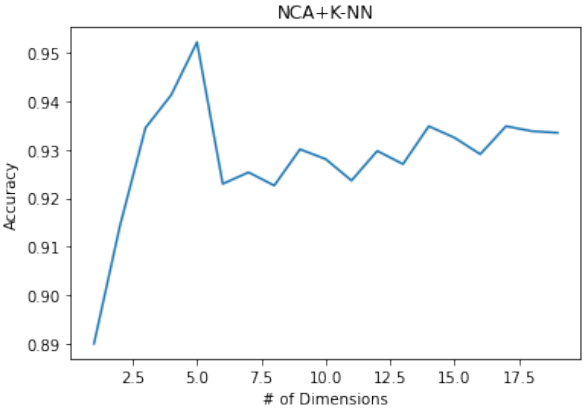

Figure 6.3: **Best accuracy is 0.952 with 5 components.**

Now we further cross validate our parameters and tune our number of neighbours 'k' with the number of dimensions fixed to **'n=5'**.

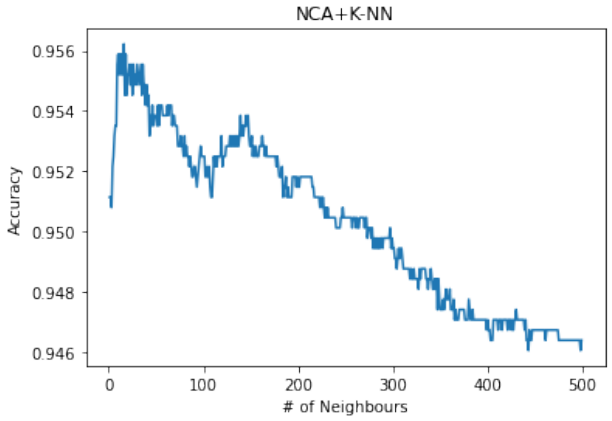

Figure 6.4: **Maximum accuracy is 0.956 with 16 neighbours.**

6.3 Final Parameters

Now we once again perform **10 times 10 fold** validation with our parameters **k=16** and **n=5**.

10x10 Fold Accuracy NCA:
Training Accuracy: 0.9979598861283644

7 Results

7.1 Testing Accuracy

Accuracy: 0.9562266711910418

Confusion Matrix

487	7	2	0	0	0
21	443	7	0	0	0
5	18	397	0	0	0
0	1	0	440	50	0
0	0	0	18	514	0
0	0	0	0	0	537

8 Findings & Comparisons

8.1 K-NN vs NCA+K-NN vs SVM

The training accuracy is calculated after 10 times 10 fold validation.

	Train Accu	Test Accu	Pred Time(sec)
K-NN	0.971	0.893	1.523
NCA+K-NN	0.998 ↑	0.956 ↑	0.069 ↓
SVM	0.987	0.964	1.768

As we can see that there is a **6.3% increase** in accuracy of NCA compared to weighted K-NN alone, which is a huge increase. Furthermore, if we compare the result with SVM then we can see that the testing accuracy of SVM is just 0.8% greater than that of NCA+K-NN, however the prediction time for SVM is approximately **25 times** greater!

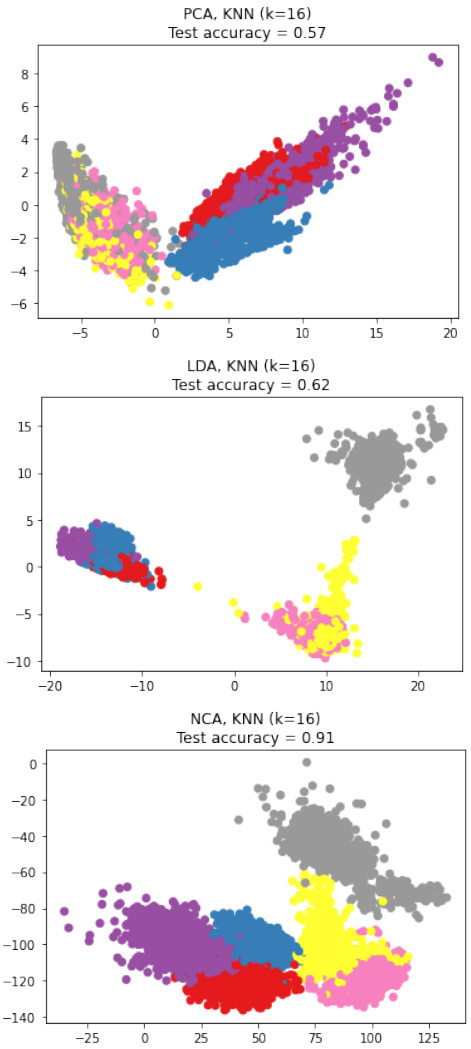

Figure 8.1: PCA vs LDA vs NCA Dimensions reduced to 2 dimensions.

9 Future Improvements

Future improvements that could be made to this algorithm is that currently the user has to tune 2 parameters manually, i.e the number of **dimensions 'n'** and the number of **neighbours 'k'**. Future versions of the algorithm should select the most optimal value for these two parameters automatically.

REFERENCES

[1] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. *In Advances in Neural Information Processing Systems*, volume 17:513–520, 2005.

[2] Scikit. 1.6. nearest neighbors. `scikit-learn.org/stable/modules/neighbors.html#nca`.