UNIVERSITY OF CAPE TOWN

EEE4114F

ELECTRICAL AND COMPUTER ENGINEERING

Distance metric learning

Author:
Yusuf Heylen

Student Number:
HYLMUH001

June, 2020

Contents

1 INTRODUCTION

The use of distance functions in machine and statistical learning is ubiquitous. The *k*-Nearest Neighbours (*k*NN) algorithm classifies a input depending on the plurality type of its *k* 'nearest' neighbours. Thus, the distance function used to decide the 'nearest' elements can have a significant effect on the accuracy of the algorithm. Selecting a good distance function for a specific algorithm is an important step in the optimization of such techniques.

The question one can consider is, if a certain elements of a set are defined to be similar, can a distance function be learnt that prioritises these features of similarity [1]? For example, consider a set of different flowers. If flowers of a certain type have a highly correlated length of flower stem, can a distance metric be learnt such that very small distances are given to similar stem lengths?

We investigated the efficacy of an adaptive distance function as per [2] for a *k*NN classification algorithm. This was compared to other commonly used metrics in machine learning. These included the Minkowski distance and Mahalanobis distance. The accuracy of these were compared through an average of trials for three widely used classification databases.

2 LITERATURE REVIEW

A distance function $d(\mathbf{x}, \mathbf{y})$ or metric is a mapping of two elements in a set to a non-negative real number.

$$d : (\mathbf{x}, \mathbf{y}) \rightarrow \mathbb{R}_{>\mathbf{0}}$$

This non-negative real number is defined to be the distance between the two elements of the set. The metric defines a distance for each pairwise elements of the set [3]. The distance is a measure of how 'near' these two elements in the set are under the metric considered.

Metrics must satisfy the following three properties [4]:

1. $d(\boldsymbol{x}, \boldsymbol{y}) = \mathbf{0} \implies \boldsymbol{x} = \boldsymbol{y}$
2. $d(\boldsymbol{x}, \boldsymbol{y}) + d(\boldsymbol{x}, \boldsymbol{z}) \geq d(\boldsymbol{z}, \boldsymbol{y})$ (Triangle inequality)
3. $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$ (Symmetry)

In machine learning common metrics that are used are the Manhattan, Euclidean, Chebyshev, Mahalanobis and Cosine distances. The cosine distance is strictly not a metric as it does not obey the triangle inequality, hence it was not considered [5].

2.1 Learning Linear Metrics

Consider a metric of the form:

$$d_A : (\boldsymbol{x}, \boldsymbol{y}) \mapsto ||\boldsymbol{x} - \boldsymbol{y}||_A = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^T A (\boldsymbol{x} - \boldsymbol{y})} \tag{1}$$

Note that this is strictly not necessarily a metric [1] as $d_A(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{0} \implies \boldsymbol{x} = \boldsymbol{y}$.

The question posed can be restated as: Select an A such that the similar pair wise elements have a small distance between them. This leads to an optimization problem as seen in [1].

In [2] an optimal solution for a linear metric A is found as:

$$A = (\boldsymbol{X} \bar{\boldsymbol{K}}_D^{-1} \boldsymbol{X}^T)^{-1} \tag{2}$$

Where:

- $\boldsymbol{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$ is the set of n training examples with each $\boldsymbol{x_i} \in \mathbb{R}^m$ having m features.
- $\bar{\boldsymbol{K}}_D^{-1}$ is the inverse of the 'ideal kernel matrix.' Such that $\bar{\boldsymbol{K}}_D = (\boldsymbol{Y}^T \boldsymbol{Y} + \lambda \boldsymbol{I})^{-1}$, with:
  - $\boldsymbol{Y} = (\boldsymbol{y}_1, ..., \boldsymbol{y}_n)$ is the set of class labels assigned to each training examples, where each $\boldsymbol{y}_i \in \mathbb{R}^C$ is binary encoded.
  - $\lambda \in \mathbb{R}_{>0}$ is a smoothing parameter.

2.2 Minkowski distance

The Minkowski metric is a generalisation of the Manhattan, Euclidean and Chebyshev metrics. For $p \geq 1$ it is defined as:

$$d_p : (\boldsymbol{x}, \boldsymbol{y}) \mapsto ||\boldsymbol{x} - \boldsymbol{y}||_p = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}$$

2.2.1 *Manhattan distance*

When $p = 1$, the manhattan or taxicab distance is obtained:

$$d_1 : (\boldsymbol{x}, \boldsymbol{y}) \mapsto ||\boldsymbol{x} - \boldsymbol{y}||_1 = \sum_{i=1}^{n} |x_i - y_i|$$

It is named so as it can be thought of as the total distance a taxicab travels in a grid-like city. For a point x on the origin of $\mathbb{R}^2$, the manhattan distance that is less than or equal to one can can be seen as selecting for all those points y that lie on or within the diamond centred around the origin as in figure 1. This is known as the unit closed ball for metric d_1 in $\mathbb{R}^2$ [6]. The generic closed ball for a metric d acting on a set D that contains elements x and y, for a given radius $r > 0$ is given by:

$$B_r(x \in D) = \{y \in D : d(x, y) \leq r\}$$

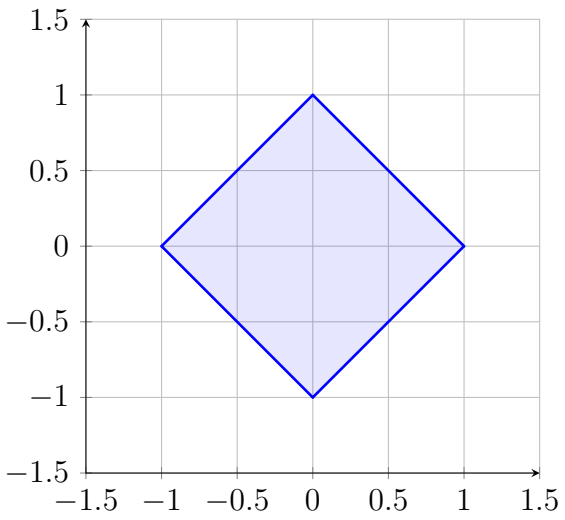

Figure 1: The unit closed ball $B_1(0)$ for metric d_1 in $\mathbb{R}^2$

2.2.2 *Euclidean distance*

When $p = 2$, the Euclidean distance is obtained:

$$d_2 : (\boldsymbol{x}, \boldsymbol{y}) \mapsto ||\boldsymbol{x} - \boldsymbol{y}||_2 = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$$

This distance metric is colloquially known as 'how the crow flies,' as it is the distance of a direct route that a bird flies between two locations. For a point x on the origin of $\mathbb{R}^2$, the euclidean distance that is less than or equal to one can can be seen as selecting for all those points y that lie on or within the circle centred around the origin as seen figure 2.

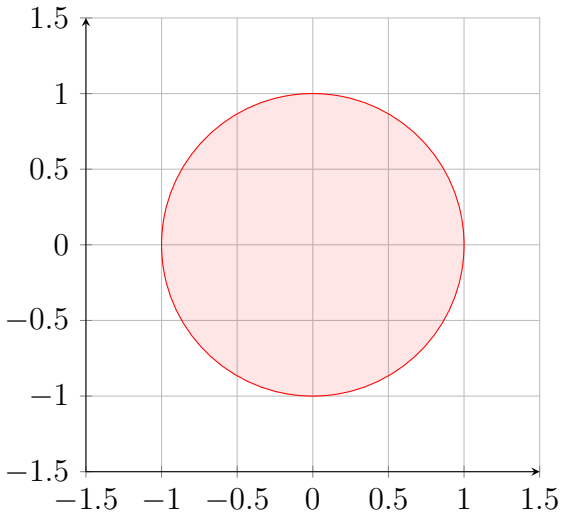

Figure 2: The unit closed ball $B_1(0)$ for metric d_2 in $\mathbb{R}^2$

The euclidean distance is also obtained by setting $A = I$ in equation 1. A weighted version of the euclidean distance can be used by setting $A = W$, where W is a diagonal matrix such that each diagonal entry sets the weighting of a particular feature [5].

2.2.3 Chebyshev distance

In the limit as $p \to \infty$ the Chebyshev distance is obtained. This simplifies to selecting the maximum of the absolute differences of relative components of the two pairwise elements:

$$d_\infty : (\boldsymbol{x}, \boldsymbol{y}) \mapsto ||\boldsymbol{x} - \boldsymbol{y}||_\infty = \lim_{p \to \infty} \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p} = \max_i |x_i - y_i|$$

This is the distance metric is known as 'chessboard distance,' for it is the distance that a king has to travel on a chess board. For a point x on the origin of $\mathbb{R}^2$, the chebyshev distance that is less than or equal to one can can be seen as selecting for all those points y that lie on or within the squared centred around the origin as in figure 3.

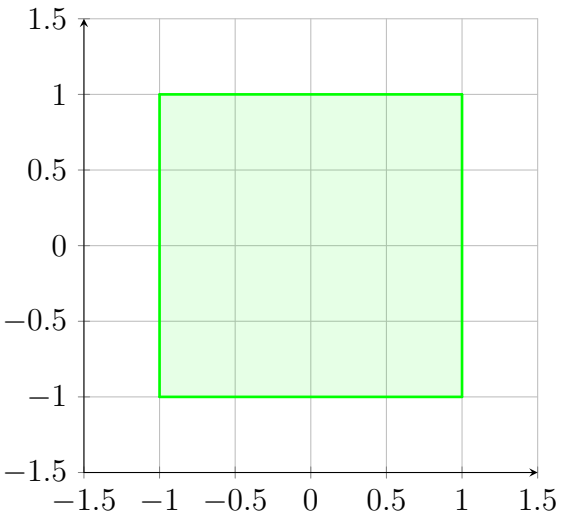

Figure 3: The unit closed ball $B_1(0)$ for metric d_∞ in $\mathbb{R}^2$

2.3 Mahalanobis distance

The Mahalanobis distance takes into account any covariance in a set. It effectively first rescales the set along its principle components and removes the covariance [7], before computing the euclidean distance between the points. This is seen in figure 4.

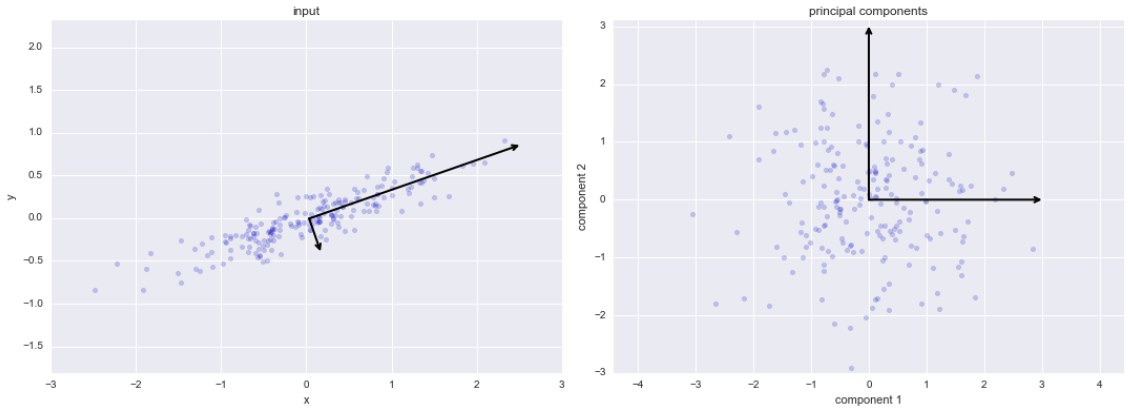

Figure 4: Rescaling of data along it's principle components (taken from [8], under MIT licence)

The mahalanobis distance is obtained by obtained by setting $A = S^{-1}$ in equation 1. Where S is the variance-covariance matrix of the shared set that $\boldsymbol{x}$ and $\boldsymbol{y}$ belong to:

$$d_M : (\boldsymbol{x}, \boldsymbol{y}) \mapsto \sqrt{(\boldsymbol{x} - \boldsymbol{y})^T S^{-1} (\boldsymbol{x} - \boldsymbol{y})}$$

3 METHOD

A *k*NN algorithm for clustering was developed in python 3.7. The four nearest neighbours were looked at (i.e. a value of $k = 4$ was selected.) The Manhattan and Euclidean distances were implemented using a generalised Minkowski distance function. The simplified version of the Chebyshev metric was implemented. The Mahalanobis distance was implemented using the 'scipy.spatial.distance' library. This was done as implementation of matrix operations is notoriously computationally expensive, and thus a library has a good chance of being pre-optimised. All of these where used for comparison to each other and the learnt distant metric.

The implementation of the learnt linear distance metric was done by calculating A as in equation 2. This was chosen instead of the implementation with the variance and mean matrices as defined by equation 11 in [2]. This is as the calculation of the variance-covariance matrices is computationally expensive and as the version in equation 2 is much simpler to implement in code. The value of the smoothing parameter λ was randomly selected to be a half. Then the final distance was computed using equation 1.

Three UCI databases were used. These are the Iris, Wine and Breast Cancer databases. These were chosen as they are classic databases used in classification (Iris and Wine are both used in [2] and all three are used in [1]), and as they come pre-packaged with in the 'sklearn.databases' library. Three were chosen in order to gain a better understanding of the performance of the various algorithms. Table 1 gives a detailed description of the datasets. The databases were randomly split half into a training set and half into a test set.

Dataset	Size	Classes	Dimensions
Iris	150	3	4
Wine	178	3	13
Breast Cancer	569	2	30

Table 1: Description of UCI datasets

The accuracy and confusion matrices for each distance metric was then computed using each distance metric. This was repeated one-hundred times and the average for the accuracy and the confusion matrices over the one-hundred trials per dataset and metric were reported to an accuracy of four decimal points. The databases were then randomly shuffled and split in half for each trial.

All code is available on GitHub at:
`https://github.com/yusufheylen/kNN_Distance_metric_learning`.

4 RESULTS

Table 2 presents the average accuracy of each metric over each dataset.

Dataset	Manhattan	Euclidean	Chebyshev	Mahalanobis	Learnt
Iris	0,9464	0,9521	0,9536	0,8560	**0,9731**
Wine	0,7228	0,6828	0,6713	0,8619	**0,9430**
Cancer	**0,9327**	0,9225	0,9177	0,8362	0,9201

Table 2: Average classification accuracy over 100 trials, best in bold.

4.1 Confusion matrices

The following section present the confusion matrices for each metric and dataset as asked. The results are normed such that 1 means a 100% positive result for the identification of a cell as such. This section can be safely skipped to the discussion 5.

4.1.1 Confusion matrices - Manhatten distance

d_1	Setosa	Versicolor	Virginica
Setosa	0.3315	0	0
Versicolor	0	0.3167	0.0173
Virginica	0	0.0363	0.2983

Table 3: Confusion matrix for the Iris dataset under the Manhattan metric

d_1	Class 1	Class 2	Class 3
Class 1	0.3036	0.0094	0.0191
Class 2	0.0289	0.3107	0.0601
Class 3	0.0306	0.1291	0.1085

Table 4: Confusion matrix for the Wine dataset under the Manhattan metric

d_1	**Malignant**	**Benign**
Malignant	0.3320	0.0378
Benign	0.0295	0.6007

Table 5: Confusion matrix for the Breast Cancer dataset under the Manhattan metric

4.1.2 Confusion matrices - Euclidean distance

d_2	**Setosa**	**Versicolor**	**Virginica**
Setosa	0.3315	0	0
Versicolor	0	0.3181	0.0159
Virginica	0	0.032	0.3025

Table 6: Confusion matrix for the Iris dataset under the Euclidean metric

d_2	**Class 1**	**Class 2**	**Class 3**
Class 1	0.2976	0.0112	0.0233
Class 2	0.0296	0.2940	0.0761
Class 3	0.0345	0.1426	0.0911

Table 7: Confusion matrix for the Wine dataset under the Euclidean metric

d_2	**Malignant**	**Benign**
Malignant	0.3305	0.0393
Benign	0.0382	0.5919

Table 8: Confusion matrix for the Breast Cancer dataset under the Euclidean metric

4.1.3 Confusion matrices - Chebyshev distance

d_∞	**Setosa**	**Versicolor**	**Virginica**
Setosa	0.3315	0	0
Versicolor	0	0.3213	0.0127
Virginica	0	0.0337	0.3008

Table 9: Confusion matrix for the Iris dataset under the Chebyshev metric

d_∞	**Class 1**	**Class 2**	**Class 3**
Class 1	0.2954	0.0121	0.0246
Class 2	0.0304	0.2879	0.0813
Class 3	0.0354	0.1447	0.0881

Table 10: Confusion matrix for the Wine dataset under the Chebyshev metric

d_∞	**Malignant**	**Benign**
Malignant	0.3296	0.0402
Benign	0.0421	0.5880

Table 11: Confusion matrix for the Breast Cancer dataset under the Chebyshev metric

4.1.4 Confusion matrices - Mahalanobis distance

d_M	**Setosa**	**Versicolor**	**Virginica**
Setosa	0.328	0.0035	0
Versicolor	0.0015	0.3128	0.0197
Virginica	0	0.1193	0.2152

Table 12: Confusion matrix for the Iris dataset under the Mahalanobis metric

d_M	**Class 1**	**Class 2**	**Class 3**
Class 1	0.3289	0.0027	0.0006
Class 2	0.0849	0.2912	0.0235
Class 3	0.0126	0.0138	0.2418

Table 13: Confusion matrix for the Wine dataset under the Mahalanobis metric

d_M	**Malignant**	**Benign**
Malignant	0.2185	0.1513
Benign	0.0125	0.6177

Table 14: Confusion matrix for the Breast Cancer dataset under the Mahalanobis metric

4.1.5 *Confusion matrices - Learnt metric*

d_A	**Setosa**	**Versicolor**	**Virginica**
Setosa	0.3315	0	0
Versicolor	0	0.3241	0.0099
Virginica	0	0.0171	0.3175

Table 15: Confusion matrix for the Iris dataset under the Learnt metric

d_A	**Class 1**	**Class 2**	**Class 3**
Class 1	0.3276	0.0045	0
Class 2	0.0409	0.3491	0.0097
Class 3	0	0.0019	0.2663

Table 16: Confusion matrix for the Wine dataset under the Learnt metric

d_A	**Malignant**	**Benign**
Malignant	0.3001	0.0697
Benign	0.0102	0.6199

Table 17: Confusion matrix for the Breast Cancer dataset under the Learnt metric

5 ANALYSIS

It is clear from the results that the learnt distance metric was the most accurate for the majority of the time. This is seen in table 2 where it is the best performing metric for all but the Breast Cancer dataset. Even in that dataset it is only off from the most accurate result (the Manhattan distance) by just over 1%. On the other hand, the learnt metric performs exceptionally well on the Wine dataset where the other metrics (besides the Mahalanobis one) perform poorly. In this database the Learnt metric is over 27% better than the worst accuracy and 8% better than the next best. In none of the scenarios does it fall below a 90% accuracy.

Of particular interest is the results from the wine dataset where all the Minkowski type metrics fail to get an accuracy above 75%. Upon inspection of the confusion matrices for these metrics, we see that they all suffer from around 13-14% error in predicting a class 3 type as a class 2 (and the converse error is also relatively high for these metrics.) Whereas the Mahalanobis only has slightly larger than 1% error and the Learnt metric an almost 0.2% error for this scenario. This is probably to do with the covariance removing effects of the Mahalanobis metric.

Interestingly enough, the Mahalanobis metric is very consistent in its accuracy spread – only deviating by just under 3%. Yet these accuracies can be described as decent as they are all in the mid 80 percent. The covariance in a dataset probably at times helps with classification. Yet, it can be a hindrance as in the Wine scenario. The Learnt metric seems to be able to use these when necessary.

6 CONCLUSION & RECOMMENDATIONS

In conclusion, the learnt distance metric as described in [2] has a significant effect on the accuracy and consistency of a kNN classifier. It seems to answer the question posed - a distance function can be learnt to preserve features of similarity.

Further analysis can be done to further investigate the adaptive metric. These would be:

1. Wrap the learning function in an optimization library to find the most optimal value for λ in a dataset.
2. Extend the learnt metric A to include non-linear possibilities. This is described in [2].
3. Examine the efficacy of the learning metric for other types of algorithms.

References

[1] E. Xing, A. Ng, M. Jordan, and S. Russell, "Distance metric learning, with application to clustering with side-information," *Adv. Neural Inf. Process. Sys*, vol. 15, 06 2003.

[2] S. Wang and R. Jin, "An information geometry approach for distance metric learning." *Journal of Machine Learning Research - Proceedings Track*, vol. 5, pp. 591–598, 01 2009.

[3] A. Arkangel'skii and L. Pontryagin, *General Topology I: Basic Concepts and Constructions Dimension Theory*. Springer, 1990.

[4] A. Gray, E. Abbena, and S. Salamon, *Modern differential geometry curves and surfaces*. Chapman and Hall/CRC, 2006.

[5] S. Raschka, *STAT 479: Machine Learning Lecture Notes - 2 Nearest Neighbor Methods*. University of Wisconsin-Madison, 2019. [Online]. Available: http://stat.wisc.edu/~sraschka/teaching/stat479-fs2019/

[6] J. Hunter, *An Introduction to Real Analysis - Chapter 7: Metric Spaces*. University of California at Davis, 2012. [Online]. Available: https://www.math.ucdavis.edu/~hunter/m125a/intro_analysis_ch7.pdf

[7] R. G. Brereton, "The mahalanobis distance and its relationship to principal component scores," *Journal of Chemometrics*, vol. 29, no. 3, pp. 143–145, 2015.

[8] J. VanderPlas, *Python Data Science Handbook*, 1st ed. O'Reilly Media, 2016.