Distance Metrics for Classification

Angelos Filos
Imperial College London
angelos.filos14@imperial.ac.uk

Youssef Rizk
Imperial College London
youssef.rizk14@imperial.ac.uk

Abstract

We present a variety of approaches for multi-class classification on the given data set, including K-Nearest-Neighbours (KNN) classifiers, Nearest Centroid (NC) classifiers and finally Multi-Layer-Perceptron (MLP) classifiers. Extensive research is conducted to find the hyperparameters that maximize classification accuracy.[1]

1. Dataset & Definitions

UCI Wine Data Set is used. The dataset is splitted by construction to D_{train} and D_{test} sets. We obtain $N_{train} = 118$ (training points) and $N_{test} = 60$ (testing points). Classifier $\mathbf{f}$ is trained to map the feature space $\mathcal{X} \subseteq \mathbb{R}^{13}$ to the target set $\mathcal{Y} = \{1, 2, 3\}$, such that:

$$\mathbf{f} : \mathcal{X} \mapsto \mathcal{Y} \tag{1}$$

Let feature vectors $\mathbf{x_i} \in \mathcal{X}$ and corresponding labels $y_i \in \mathcal{Y}$. For visualisation purposes, the training data set is decomposed with PCA (2 principle components used) and is illustrated in Figure 3.

2. Distance Metrics

KNN and NC classification algorithms rely heavily on distance and similarity measures[3]. Consequently, we are investigating the effectiveness of each of the distance metrics: Manhattan Distance (L_1), Euclidean Distance (L_2), Chebyshev Distance (L_∞), Chi-Square Distance, Mahalanobis Distance, Correlation and Intersection. Additionally, various transformations and normalisations[4] are used (see Figure 3) to minimize classification error. Note that there are two general classes of normalisations:

1. Datapoint-Wise (row-wise) Transformations: L_p Unit Norm transforms
2. Feature-Wise (column-wise) Transformations: Centering and Scaling transforms

[1]The code for all the experiments is available on https://github.com/filangel/distance-metrics.

3. K-Nearest-Neighbours Classification

K-Nearest-Neighbours (KNN) Classification algorithm is provided in Algorithm 1. It is parametrized by the number of nearest neighbours k, the distance metric function $\mathbf{d}$ and the transformation/normalisation function $\mathbf{g}$.

Algorithm 1: k Nearest Neighbours Classification

input: integer k,
transformation $\mathbf{g} : x \mapsto \mathbf{g}(x)$,
distance metric $\mathbf{d} : \mathbf{x}_1 \times \mathbf{x}_2 \mapsto \mathbb{R}$

foreach $x_i^{(test)} \in \mathcal{D}_{test}$ **do**
  initialize distance vector $\mathbf{r} \leftarrow \mathbf{0}$
  foreach $x_j^{(train)} \in \mathcal{D}_{train}$ **do**
    calculate $\mathbf{r}[j] \leftarrow \mathbf{d}(\mathbf{g}(x_i^{(test)}), \mathbf{g}(x_j^{(train)}))$
  end
  keep k labels of training points with smallest $\mathbf{r}$
  take majority vote (mode) of k labels
end

3.1. Model Selection & Results

3-fold grid-search cross-validation is performed for selecting the best hyperparameters for:

- $k \in [1, 20]$
- $\mathbf{g} \in \{$ 'L_2 Unit Norm', 'L_1 Unit Norm', 'L_∞ Unit Norm' 'Standardisation to $\mathcal{N}(0, 1)$', 'Scaled to Max Absolute Value', 'Scaled to Min and Max Values', 'Localised to Mean and Scaled to IQR' $\}$
- $\mathbf{d} \in \{$ 'Manhattan Distance', 'Euclidean Distance', 'Chebyshev Distance', 'Chi-Square Distance', 'Mahalanobis Distance' $\}$

$(k, \mathbf{g}, \mathbf{d})$	**CV Score**	**Accuracy**
(1, Raw, Chi-Square)	93.22 %	91.67 %
Datapoint-Wise Transformations		
(1, L_1, Chi-Square)	99.15 %	95 %
(1, L_2, Chi-Square)	100 %	95 %
(1, L_∞, Chi-Square)	99.15 %	93.34 %
Feature-Wise Transformations		
(7, $\mathcal{N}(0,1)$, Manhattan)	91.52 %	90 %
(9, MaxAbs, Manhattan)	94.91 %	91.67 %
(7, MinMax, Euclidean)	94.06 %	86.67 %
(7, Robust, Euclidean)	98.31 %	95 %

Table 1. KNN: Model Selection Summary.

A summary of the in-sample (cross-validation) and out-of-sample (test) accuracies for the different combinations is provided at Table 1.

We highlight the fact that the datapoint-wise transformations work best with $k = 1$ and $\mathbf{d}$ = Chi-Square, while the feature-wise transformations need more neighbours (i.e. $k > 1$) and $\mathbf{d}$ = Manhattan or Euclidean. The hyperparameters $(k, \mathbf{g}, \mathbf{d})$ are chosen according to the Cross-Validation Score (**CV Score**). As a result from Table 1, we choose:

$$(k, \mathbf{g}, \mathbf{d}) = (1, L_2, \text{Chi-Square})$$

The confusion matrix for this model is provided in Figure 1 and its classification accuracy is **95%**.

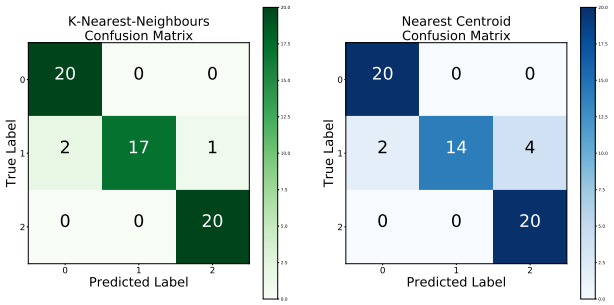

Figure 1. Confusion Matrices: (left) KNN and (right) CN.

4. Nearest Centroid Classification

The K-Nearest-Neighbour Classification algorithm is a non-parametric model with constant fitting (training) time[2]. Nevertheless, during prediction (testing) there is a two-dimensional iteration space along the D_{train} and D_{test} sets, leading to $\mathcal{O}(N_{train}N_{test})$ time complexity. From a computational perspective, this can be quite intensive, especially if the training set is sufficiently large. In this section, we introduce the idea of Centroids $\mathbf{C_l}$ for classes $\mathbf{l} \in \mathcal{Y}$. The resulting classifier is called Nearest Centroid (NC) Classifier. The motivation behind it is the representation of all datapoints $\mathbf{x_i^l}$ in class $\mathbf{l}$ by a single point, the Centroid $\mathbf{C_l}$. Then, a KNN classifier is trained on the Centroids $\mathbf{C_l}$, resulting in a prediction time complexity $\mathcal{O}(|\mathcal{Y}|N_{test})$, where $|\mathcal{Y}|$ the number of classes. Under the, usually realistic, assumption $|\mathcal{Y}| << N_{train}$ the prediction time is significantly improved. However, the calculation of the Centroids is a function of N_{train}, increasing the fitting time, but the memory complexity is also improved since after the calculation of $\mathbf{C_l}$, the training points can be discarded. The Centroids are calculated using K-Means Clustering, where K = $|\mathcal{Y}|$, and the procedure is covered in Algorithm 2

Algorithm 2: Centroid Calculation using K-Means Clustering

input: transformation $\mathbf{g} : x \mapsto \mathbf{g}(x)$

$\mathcal{G} \leftarrow |\mathcal{Y}|$-Means Clustering in $\mathbf{g}(D_{train})$
foreach cluster $\mathbf{c} \in \mathcal{G}$ **do**
  $\mathbf{l} \leftarrow$ mode of classes from datapoints in $\mathbf{c}$
  $\mathbf{C_l} \leftarrow$ center of cluster $\mathbf{c}$
end

4.1. Model Selection & Results

The calculation of the Centroids using K-Means clustering is parametrised by the transformation $\mathbf{g}$, while the KNN classifier is parametrised by the distance metric function $\mathbf{d}$. A 3-fold grid-search cross-validation is performed on the same grid with Section 3.1[3]. The optimal hyperparameters are found to be:

$$(\mathbf{g}, \mathbf{d}) = (\text{MinMax}, \text{Correlation})$$

The corresponding confusion matrix is provided in Figure 1 and the classification accuracy of the selected model is **90%**. As expected the NC classifier is more efficient but less accurate than the KNN classifier, since the compression (K-Means clustering) operation results in information loss.

5. Multi-Layer-Perceptron Classifier

For both NC and KNN classifiers we are manually choosing the distance metric function $\mathbf{d}$ and selecting

[2]storage of training samples to memory.

[3]Note that $k = 1$ since each class is represented by a single point, its Centroid.

$(\mathbf{h}, \phi, \mathbf{a}, \mathbf{b}, \mathbf{d})$	**Val Score**	**Accuracy**
$((50,), \texttt{sigmoid}, 0.1, 0.4, \text{'Raw'})$	98.31 %	96.67 %
$((25,), \texttt{sigmoid}, 0.01, 0.2, \text{'MinMax'})$	100 %	100 %
$((100,), \texttt{tanh}, 0.01, 0.2, \text{'}\mathcal{N}(0,1)\text{'})$	93.22 %	90 %

Table 2. KNN: Model Selection Summary.

the most suitable for our dataset with grid-search cross-validation. This is a valid approach, but its performance is only empirically proven. A more data-driven approach would be to do Distance Metric Learning, using a Neural Network. Specifically, we train Multi-Layer-Perceptron (MLP) Classifiers on raw data and **g**-transformed data[4]. We use Adam Optimizer[1] with initial learning rate $\eta = 0.01$, L_2 Regularization[2] and Dropout[2].

5.1. Model Selection & Results

Because of the large number of hyperparameters of MLP classifier, grid-search cross-validation is very inefficient and we will use a validation set instead, where $N_{validation} = 20$:

- number of hidden layers $\mathbf{h} \in \{(25,), (50,), (100,), (10, 10), (10, 25)\}$
- activation function $\phi \in \{\text{'}\texttt{tanh}\text{'}, \text{'}\texttt{sigmoid}\text{'}, \text{'}\texttt{relu}\text{'}\}$
- L2-regularization strength $\mathbf{a} \in [10^{-3}, 10^{3}]$
- Dropout strength $\mathbf{b} \in \{0.2, 0.4, 0.6, 0.8\}$
- $\mathbf{d} \in \{$ 'Manhattan Distance', 'Euclidean Distance', 'Chebyshev Distance', 'Chi-Square Distance', 'Mahalanobis Distance' $\}$

According to the validation summary at Table 2, the optimal model, based on Validation Score (**Val Score**) is:

$$(\mathbf{h}, \phi, \mathbf{a}, \mathbf{b}, \mathbf{d}) = ((25,), \texttt{sigmoid}, 0.01, 0.2, \text{'MinMax'})$$

As expected the preprocessing led to a more accurate model, achieving **100%** classification accuracy.

References

[1] A, Karpathy. Learning, 2015. [Online]. Available: `http://cs231n.github.io/neural-networks-3/`. [Accessed: 16- Dec- 2017].

[4] Theoretically, given enough capacity MLP Classifiers can implement feature-wise transformations as a part of their features extraction process. Nonetheless, we showed practically that data preprocessing leads to more accurate models.

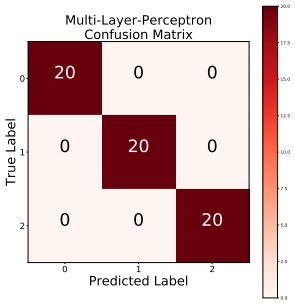

Figure 2. MLP: Confusion Matrix.

[2] A, Karpathy. Setting up the data and the model, 2015. [Online]. Available: `http://cs231n.github.io/neural-networks-2/`. [Accessed: 16- Dec- 2017].

[3] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, Stuart Russell. Distance metric learning, with application to clustering with side-information, 2015. [Online]. Available: `http://ai.stanford.edu/~ang/papers/nips02-metric.pdf`. [Accessed: 16- Dec- 2017].

[4] scikit-learn developers. Preprocessing data, 2015. [Online]. Available: `http://scikit-learn.org/stable/modules/preprocessing.html`. [Accessed: 16- Dec- 2017].

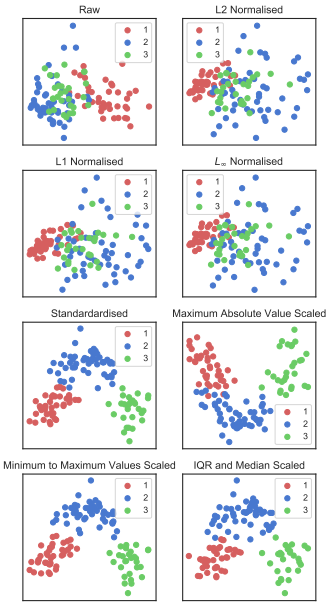

Figure 3. Visualisation of training data with PCA and Normalisation Transformations.