Assignment 2 - Group 32[*]

Bas van der Bijl (sbl206), Max de Bruijne (mbe345), and Tom de Valk (tvk550)

Vrije Universiteit Amsterdam, 1081 HV Amsterdam, The Netherlands

1 Theoretical Exercises

Exercise 5.2 In the comparison of two time series, dynamic time warping can accommodate shifts in these time series as well as time series of varying length, making it more suitable than shortest-path distance metrics [2, 8]. According to R.J. Kate, the best alignment between two time series is given by the minimum cost alignment, using the Euclidean distance, which can be calculated for time series Q and C as follows [10]:

$$DTW(Q,C) = \arg\min_{W=w1,\ldots,w_k,\ldots,w_K} \sqrt{\sum_{k=1,w_k=(i,j)}^{K} (q_i - c_j)^2}$$

A concrete example in which dynamic time warping excels, is speech recognition. In this field, time series are often not the same length and are often shifted along the time axis.

Exercise 5.7 When clustering the high-dimensional data, the addition of features results in an exponential growth of the distance matrix, which may exhaust computational resources. In general, clustering algorithms are not suitable for datasets with large amounts of features as the resulting clustering is likely one large cluster.

In the context of machine learning for the quantified self, a user could have collected data using several sensory devices. This could lead to a large amount of features such as accelerometer, gyroscope, heart rate, activity, temperature, burned calories, steps, etc. Joining the data from the different devices, results in a high dimensional space for which clustering could be problematic. The distance matrix would potentially be enormous and the results will not be very insightful. It may be possible to select a smaller subset of features for clustering without losing too much information. In this way, the distance matrix would be reduced in size and the clustering may yield better results.

Exercise 6.1 Hoogendoorn and Funk [7] refer to the best fitting function as the function yielding the best performance on the training set. Whether this is a good fit, can be determined after analyzing the performance of the same model on the validation or test set (unseen data). Whether these results are sufficient depends on the practical impact of making incorrect predictions. In the detection of COVID-19, for instance, the impact of not detecting infected individuals (false negative) may be larger than the incorrect detection of a COVID-19 infection for

[*] Part of the 2021 Machine Learning for the Quantified Self course at VU Amsterdam, The Netherlands

an uninfected individual (false positive). Therefore, it is important to understand how a function behaves on unseen data and adjust the decisions for this model to satisfy the practical requirements.

Exercise 6.7 For a binary classification problem, a model often predicts the class of an instance based on a probability. Whether a value of either 0 or 1 is assigned to an instance, is dependent on Θ_{cut}. When the classification probability is higher than Θ_{cut}, the output is 1, and 0 otherwise. Increasing Θ_{cut} indicates that the certainty of a model should be higher when classifying instances, which should result in less false positive instances. Decreasing Θ_{cut} works the other way around; the number of false positive instances increase which results in an increase in true positive rate and a false positive rate. Since there are certain costs for incorrectly predicted instances, the costs for different values for Θ_{cut} vary.

When the minimum of the cost estimate is achieved at a Θ_{cut} of 0 or 1, this is an indication of highly imbalanced data. The predictions of the model shift toward the majority class which may reduce the impact of Θ_{cut}. Furthermore, this suggests that the impact of the wrongly predicted instances of the minority class do not impact the expected cost function enough to compensate for the costs of the wrongly predicted majority class instances.

Exercise 7.3 The use of multi-layered perceptron networks depends on a number of parameters. One of these parameters is the number of nodes in the hidden layer(s). According to A. Balota et al. [16], there are three basic rules to determine the number of hidden nodes h:

- h should be between the size of the input and output layers
- h should be equal to $\frac{2}{3}(input\_size + output\_size)$
- h should be less than $2 * input\_size$

Other studies have presented other guidelines for the selection of the number of hidden nodes. However, more suitable numbers of hidden nodes may be found through trial-and-error or optimization techniques such as neural architecture search algorithms [4].

Exercise 7.6 Support vector machines use kernel functions to project low-dimensional data to a higher-dimensional space, in which the data becomes separable [14]. This separability allows for the classification of the data. Three commonly-used kernel functions are [11, 13]:

1. Linear kernel function: $K(x_i, x_j) = x_i^T x_j$
   The linear kernel function is suitable for linearly separable data and has the properties that is extremely simple to compute and that it is not likely to overfit the learned model [13].

2. Gaussian kernel function: $K(x_i, x_j) = \exp(\frac{-||x_i - x_j||^2}{2\sigma^2})$
   In this equation, σ is the parameter that determines the spread of the Guassian function, or the region of influence of an instance in the vector space. The Guassian kernel function is suitable for the non-linearly separable data. It will also always lead to the optimal solution for linearly separable data. The tuning of σ, however, can be computationally expensive and does not guarantee the discovery of a better model [13].
3. Sigmoid kernel function: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + c)$
   In this equation, c is a smoothing parameter that is used to determine the area of influence that the input vector has [6]. The parameter γ is a scaling factor for the input and is generally set to $\frac{1}{N}$ where N is the size of the dataset. The sigmoid kernel function is suitable for non-linearly separable data, but generally underperforms than the RBF kernel function [11].

Exercise 7.8 When using a nearest neighbors approach for high-dimensional data, all features have the same influence on the prediction of an instance. This means that even features that negatively affect the prediction, are used. Model-based approaches can generally add a certain weight to each feature, increasing or decreasing it's effect.

Next to underperforming with respect to model-based approaches, nearest neighbor approaches are also remarkably expensive to compute for high-dimensional data. Considering that distance metrics, such as the Euclidean distance, are calculated by computing the difference between two instances for an attribute p, this computation can get extensive for large values p.

Exercise 7.13 Overfitting is a problem for predictive models where the generalizability of the model is extremely limited. In this case, the model performs well on the training data, but performs poorly on unseen data (the validation and test sets). This is often the case when a model considers all of the features in a high-dimensional data set, even though some of these features have a limited (or even negative) effect on the output of the model [17]. This is a problem that occurs often for data with high-dimensionality and small samples [12]. A remedy for this is feature selection, where only useful features are used to train the predictive model . This feature selection can reduce the dimensionality of the data set and the complexity of the model while increasing the generizability and predictive performance of the model [17].

Exercise 8.5 The "no free lunch theorem" (NFLT) states that for all optimisation problems, every non-resampling optimisation algorithm performs equally well [9]. In other words, the error rate is the same for all solution methods when averaged over all problems. In the field of reservoir computing, there is one large reservoir consisting of hidden neurons that are fully connected with randomly assigned weights [7]. For a particular dataset, an initialisation of the random reservoir according to a specific strategy, can be most suitable. However, considering the NFLT, there is no optimal strategy to initialize the random reservoir across all datasets. All initialisation strategies perform equally well when balanced out over all datasets.

Exercise 8.6 The main difference between a feed forward neural network and recurrent neural network (RNN) is that a RNN allows for information travel between hidden nodes, where a feed forward neural network does not. This means that hidden nodes in an RNN take the output of the previous hidden node as input, in addition to the input values. RNNs take advantage of time dependence and are therefore a great tool for the prediction of time series, which are time dependent [5]. A topic in the setting of the quantified self can be diabetes management, for which it is of great importance to have a representation of the future glucose concentration of a person. The future values of the glucose concentration can be used to adjust insulin injections or insulin infusion rates. The glucose concentration depends on present and past values of activities, such as tiredness, eating and stress. Therefore, an RNN is expected to work better than a feed forward neural network for the prediction of glucose concentration.

Exercise 8.8 Hoogendoorn and Funk discuss three algorithms for parameter optimization, particularly: simulated annealing (SA), genetic algorithms (GA) and multi-criteria optimization [7]. All three algorithms are used to find the parameter values that optimize a certain function.

Simulated Annealing (SA) randomly searches through the parameter space looking for performance improvements. The SA algorithm does consider worse configurations with a particular probability. This enables the algorithm to escape local optima. Since the SA algorithm is highly dependent on a random component, the convergence to a global optimum is not guaranteed [1].

Genetic Algorithms (GA) represent individuals as encodings of parameter values, all forming a solution to a certain problem. After starting with a random initialization of these individuals, they will converge towards the optimal solution after a large number of generations. The algorithm does not guarantee to be the optimal solution due to the possibility of local optimum individuals that can start to dominate the population [3].

According to Vakhania et al., an optimization problem consisting of multiple targets can not be guaranteed to find an optimal solution using the multi-criteria optimization algorithm due to the existence of different objective criteria [15].

Exercise 9.2 The total number of steps per day can a good indicator to determine whether or not a person was active during that day. When someone has taken a lot of steps, which can vary per person, it can be concluded this person has been fairly active. For very few steps taken during the day the opposite can be concluded. In the context of a reward function, a higher number of steps will lead to a higher reward.

Exercise 9.4 According to Hoogendoorn and Funk, a state has the *Markov property* when the previous state contains all the information that is needed to compute the probability of moving forward to the next state [7]. Therefore, the Markov property does not hold if additional historic information is needed to compute the next transition of the system. An example of a setting in the quantified self where the Markov property does not hold is in the prediction of future values for the mood of a user. The mood of a person depends on a wide number of variables, which all have a different impact. For example, the mood can depend on this person's relationship status for the past months or on the amount of sleep he or she has had in the past week. This shows that the future mood of a person does not only depend on the previous interval, indicating that the Markov property does not hold.

2 Practical Exercises

Exercise 5.1 The *gyroscope* data from the *crowdsignals* dataset has been clustered using k-means, k-medoids and hierarchical (agglomerative) clustering. The silhouette coefficients of the different methods have been plotted in Figure 1, showing similar silhouette scores for different numbers of k for k-means and k-medoids, and higher silhouette scores for larger values of k for hierarchical (agglomerative) clustering. Figure 2 shows the spread of the clusters (for k-medoids with $k = 4$) in a three-dimensional space (x, y, z) and Figure 3 shows the silhouette coefficients for these clusters. It can be seen that quality of the clusters is not consistentas the silhouette score of the first cluster deviates from the silhouette scores of the other clusters. Figure 4 shows the development of the clusters using hierarchical clustering in the form of a dendrogram.

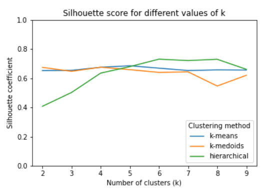

Fig. 1. Plot of the silhouette coefficient curve for k-means, k-medoids and hierarchical clustering

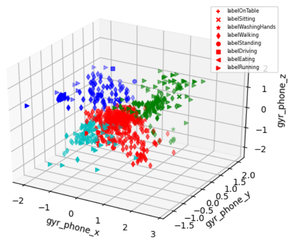

Fig. 2. Plot of cluster spread using k-medoids clustering with $k = 4$

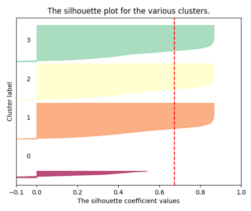

Fig. 3. Plot of the silhouette coefficient for the different clusters with $k = 4$

Considering these results, it can be stated that the clustering results are remarkably different from the accelerometer clustering, as seen in Machine Learning for the Quantified Self [7]. The three-dimensional plot of the different clusters for the accelerometer data shows much more separation between the different clusters. This is likely a result of very different acceleration values for each activity, where the gyroscope data seemed similar between activities. This is also reflected by the difference in the distances shown in the dendrograms in Figures 4 and 5.

It can also be observed that $k = 4$ led to the most suitable clustering for the gyroscope data, where $k = 6$ was most suitable for the accelerometer clustering.

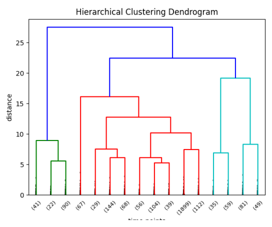

Fig. 4. Dendrogram of hierarchical (agglomerative) clustering of gyroscope data.

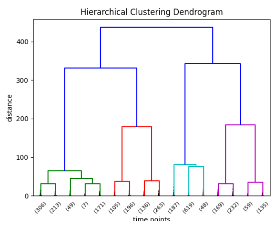

Fig. 5. Dendrogram of hierarchical (agglomerative) clustering of accelerometer data.

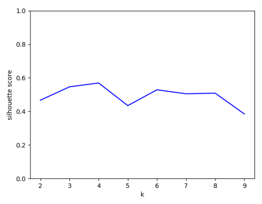

Fig. 6. Plot of silhouette coefficient using k-medoids clustering for different values of k

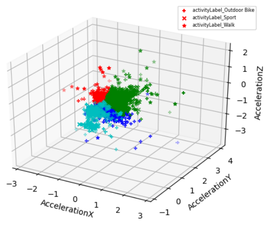

Fig. 7. Plot of cluster spread using k-medoids clustering with $k = 4$

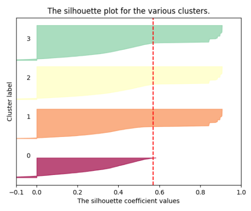

Fig. 8. Plot of the silhouette coefficient for the different clusters with $k = 4$

Exercise 5.2 Our dataset has been clustered on the x, y and z accelerometer data using k-medoids clustering. Figure 6 shows the silhouette coefficient for different values of k. It can be seen that $k = 4$ leads to the highest silhouette score. Considering $k = 4$, the clustering of the data in a three-dimensional space can be seen in Figure 7. It can be observed that the clusters are not clearly separated and that instances with different activity labels are scattered throughout the different clusters. Figure 8 shows the silhouette coefficients for these clusters. It can be seen that quality of the clusters is not consistent. The silhouette score of the first cluster deviates from the silhouette scores of the other clusters.

With respect to the clustering of the *crowdsignals* dataset on accelerometer data, it can be seen that the clustering of our dataset does not seem to be similar. The clustering of the *crowdsignals* dataset showed clear separation of the different clusters and the silhouette scores within the different clusters appeared to be consistent. The reason for these differences is likely the manner in which data was collected, where the accelerometer did not show remarkable differences in the measurements between the different activities.

Exercise 7.1 Some instances in the *crowdsignals* dataset contain more than one classification label. These instances are generally left as "undefined" as the interpreter cannot be sure which label is the true label. To be able to use these instances for classification, the "undefined" label is replaced by the last-occurring label for the given instance.

Figure 9 shows a forward selection feature selection. It can be seen that the accuracy increases remarkably with the addition of up to 10 features, after which the accuracy stabilizes. In Figure 10, it can be seen that the accuracy of the neural network model decreases as the value of the regularization parameter exponentially increases. In Figure 11, it can be seen that the random forest model performs best with a low minimum number of points per leaf. This is in concordance with expectations as adding minimum points per leaf acts as a regularization mechanism, reducing the probability of overfitting. This can also be seen as the accuracy curves for the training and test sets seem to come together slightly as the minimum points per leaf is raised. Figure 12 shows the performance of the different algorithms that were implemented for different feature sets. It can be seen that all models, except random forest and naive bayes, perform worse with the smaller subsets of features. The cause of this may be that the feature selection is done using decision tree, but this does not explain the relatively poor performance of decision trees with the feature subset. Figure 13 shows the confusion matrix for the random forest model predictions. It can be seen that a large majority of the points are gathered on the diagonal of the matrix, indicating that the points are classified correctly. Of the misclassified points, most of them are labelled with *labelWalking* and classified as *labelOnTable*.

With respect to the results of the classification without the undefined labeled instances, all of the models perform better in terms of the accuracy. This is also reflected by the confusion matrix, as more instances are shown on the diagonal of the matrix.

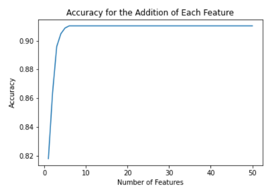

Fig. 9. Accuracy for different subsets of features

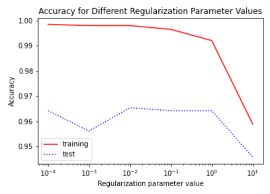

Fig. 10. Accuracy of a neural network model with different parameter values

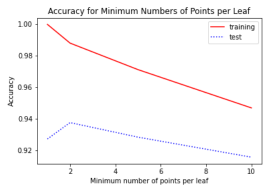

Fig. 11. Accuracy of an RF model with different minimum points per leaf

Exercise 7.3 Similarly to the *crowdsignals* dataset, the classification algorithms have implemented for the dataset that has been collected. Figure 14 shows a forward selection feature selection. It can be seen that the accuracy increases remarkably with the addition of up to 13 features, after which the accuracy stabilizes. It can be seen that the accuracy at which it stabilizes, is notably lower than the accuracy seen in Figure 9 for the *crowdsignals* data.

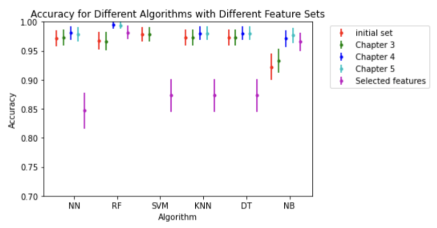

Fig. 12. Accuracy the different algorithms on different feature sets

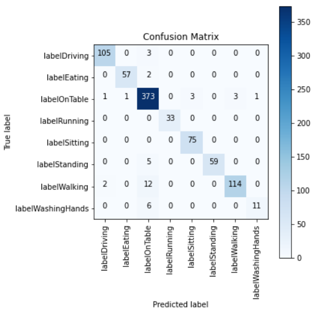

Fig. 13. Confusion matrix

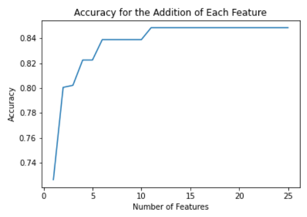

Fig. 14. Accuracy for different subsets of features

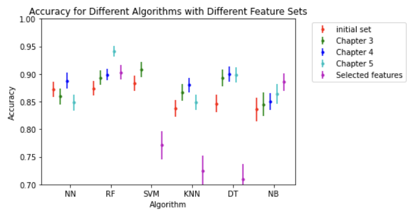

Fig. 15. Accuracy of a random forest model with different minimum number of points per leaf

In Figure 15, a plot of the accuracy for the different algorithms is given for different subsets of features. It can be seen that the models generally perform worst with the subset of selected features. It can also be seen that the random forest generally performs best, especially with the chapter 5 feature subset. This is in concordance with the results found for the *crowdsignals* data classification.

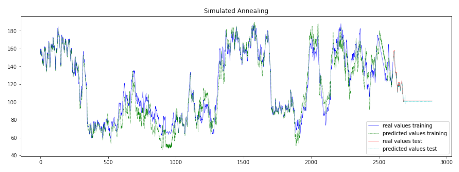

Fig. 16. Parameter Optimization using Simulated Annealing

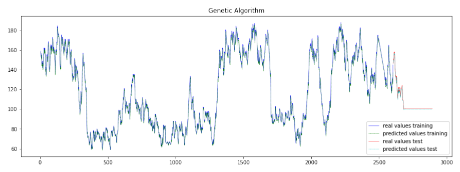

Fig. 17. Parameter Optimization using Genetic Algorithm

Exercise 8.3 A dynamical systems model was created for the *crowdsignals* data using the heart rate as the target variable. The model considers two additional features in the prediction of the heart rate, namely *acc_phone_x* and *acc_phone_y*. For the tuning of the parameters, simulated annealing and the genetic algorithm were used. The results of this models are shown in Figures 16 and 17. Overall, the predictions for the genetic algorithm seem to be better than the results for simulated annealing, as the predictions follow the observed values closely.

References

1. Austbø, B., Wahl, P.E., Gundersen, T.: Constraint handling in stochastic optimization algorithms for natural gas liquefaction processes. In: Kraslawski, A., Turunen, I. (eds.) 23rd European Symposium on Computer Aided Process Engineering, Computer Aided Chemical Engineering, vol. 32, pp. 445–450. Elsevier (2013). https://doi.org//10.1016/B978-0-444-63234-0.50075-0, https://www.sciencedirect.com/science/article/pii/B9780444632340500750
2. Begum, N., Ulanova, L., Wang, J., Keogh, E.: Accelerating dynamic time warping clustering with a novel admissible pruning strategy. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 49–58 (2015)
3. Busetti, F.: Genetic algorithms overview. Retrieved on December **1** (2007)
4. Chen, Y., Meng, G., Zhang, Q., Xiang, S., Huang, C., Mu, L., Wang, X.: Renas: Reinforced evolutionary neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4787–4796 (2019)
5. Dematos, G., Boyd, M.S., Kermanshahi, B., Kohzadi, N., Kaastra, I.: Feedforward versus recurrent neural networks for forecasting monthly japanese yen exchange rates. Financial Engineering and the Japanese Markets **3**(1), 59–75 (1996)
6. Hassan, U.K., Nawi, N.M., Kasim, S.: Classify a protein domain using sigmoid support vector machine. In: 2014 International Conference on Information Science & Applications (ICISA). pp. 1–4. IEEE (2014)
7. Hoogendoorn, M., Funk, B.: Machine learning for the quantified self. On the art of learning from sensory data (2018)
8. Izakian, H., Pedrycz, W., Jamal, I.: Fuzzy clustering of time series data using dynamic time warping distance. Engineering Applications of Artificial Intelligence **39**, 235–244 (2015)
9. Joyce, T., Herrmann, J.M.: A Review of No Free Lunch Theorems, and Their Implications for Metaheuristic Optimisation, pp. 27–51. Springer International Publishing, Cham (2018)
10. Kate, R.J.: Using dynamic time warping distances as features for improved time series classification. Data Mining and Knowledge Discovery **30**(2), 283–312 (2016)
11. Lin, H.T., Lin, C.J.: A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods. submitted to Neural Computation **3**(1-32), 16 (2003)
12. Liu, H., Dougherty, E.R., Dy, J.G., Torkkola, K., Tuv, E., Peng, H., Ding, C., Long, F., Berens, M., Parsons, L., et al.: Evolving feature selection. IEEE Intelligent systems **20**(6), 64–76 (2005)
13. McDonald, G., García-Pedrajas, N., Macdonald, C., Ounis, I.: A study of svm kernel functions for sensitivity classification ensembles with pos sequences. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 1097–1100 (2017)
14. Noble, W.S.: What is a support vector machine? Nature biotechnology **24**(12), 1565–1567 (2006)
15. Vakhania, N., Werner, F.: A brief look at multi-criteria problems: Multi-threshold optimization versus pareto-optimization. In: Multi-criteria Optimization-Pareto-optimal and Related Principles. IntechOpen (2020)
16. Vujicic, T., Matijevic, T., Ljucovic, J., Balota, A., Sevarac, Z.: Comparative analysis of methods for determining number of hidden neurons in artificial neural network. In: Central european conference on information and intelligent systems. p. 219. Faculty of Organization and Informatics Varazdin (2016)

17. Ying, X.: An overview of overfitting and its solutions. In: Journal of Physics: Conference Series. vol. 1168, p. 022022. IOP Publishing (2019)