Machine learning

*ePortfolio*



**Group 40**

Name: Stefan Winter

Student number: 2067606

Name: Joost Oudesluijs

Student number: 2056392

Name: Joost Schutte

Student number: 2069625

[880083-M-6](https://tilburguniversity.instructure.com/courses/5509)

Dr. Güven Ç and Dr. Önal, I.

Department of Cogntive science and Artificial intelligence

School of Humanities and Ditital sciences

Tilburg University

June 2021

|  |
| --- |
|  |

# Task 1

**Model 1**

**Support vector machine**

Support vector machines are algorithms that find a hyperplane that classifies all the datapoints to categories in an N-dimensional space (Ghandi, 2018). This hyperplane (black line) maximizes the distance or margin (light blue area) between different classes. The support vectors are the closest points to the classifier and determine the position of the hyperplane (see Figure 1).

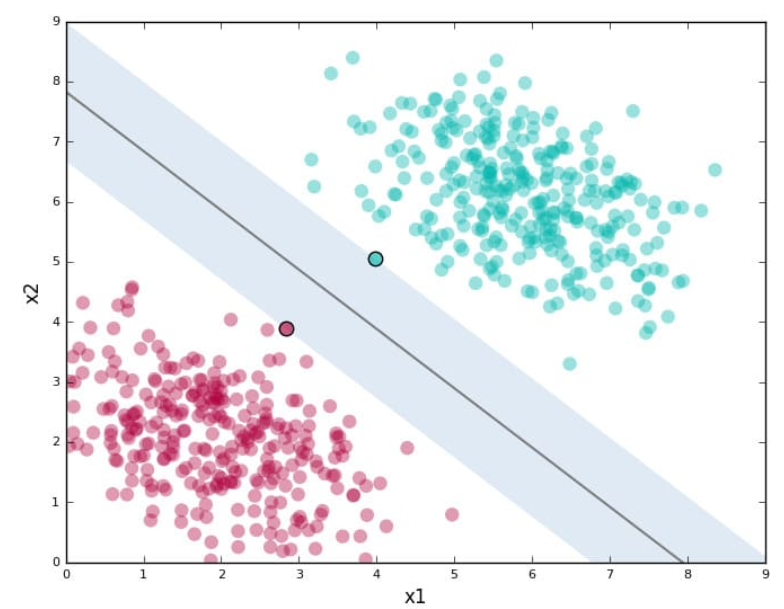


Figure 1: Visualized support vector machine.

To correctly classify the datapoints, a kernel function projects low-dimensional data into a higher-dimensional space. Enhancing the dimensions makes it possible to better divide the observations into classes. Therefore support vector machines works more effective in high dimensional spaces, or datasets with a high number of input features (Ghandi, 2018).

**Input to classifier**

To represent the image data from the train\_data\_label.npz and test\_data\_label.npz the vectors are standardized with the Standardscalar function from sklearn.preprocessing. This means that the mean is subtracted from every observation and is divided by its standard deviation. Particularly for SVM, the algorithm assumes that all features are centered around 0 and have variance in the same order (Scikit-learn, 2021). The data is not splitted into a train and a validation set because cross-validation is applied.

**Hyperparameter tuning**

To optimize the SVM (training) model a pipeline is created. Multiple hyperparameter settings are evaluated. First the Principal component analysis (PCA) is applied. PCA is often used to reduce the dimensionality and could therefore be seen as a denoising method (Husson et al., 2010). The first dimensions extract the most essential information, while the last dimensions extracts mainly noise. In this particular case the dim\_red\_\_n\_components function is applied with a range from 0.80-1 and increments of 0.01. E.g. 0.90% of all variance will be explained with in this case 58 features used, which is a reduction of 784-58 = 726 dimensions or features for training the model. To transform low dimensional input to a higher dimensional space multiple kernels are applied. For this case a polynomial, radial basis function (RBF) and a sigmoid kernel are used. Lastly to find the best parameter settings the GridsearchCV function from sklearn.model\_selection is applied with a cross-validation of five (Souza, Matwin, Japkowicz, 2002). In this particular case every part of the training data is used four times for training and one time for validation.

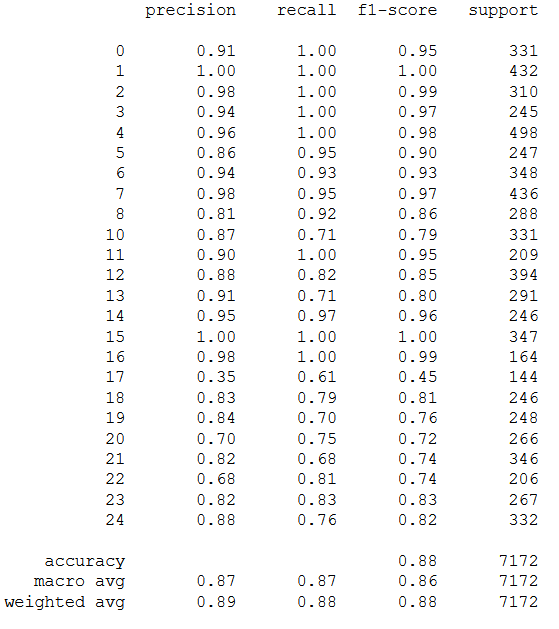
**Model training**

After hyperparameter tuning the best settings from the cross-validation are with a dimensionality of 0.91% variance explained and 65 features used. The best SVM kernel is RBF. And the training score is approximately 0.99. Since the RBF kernel and the Standardscalar both squeeze their observations to a (near) Gaussian distribution, the RBF kernel could be the best fit (Scikit-learn, 2021).

**Results**

The created hyperplane from the training set, with the appropriate hyperparameter settings, gives an accuracy of 0.88 on the test set. This means that almost 90 percent of all images are correctly classified (see Table 1). The most difficult label to classify is label 17; the letter “R”. The letters V,K,Y U and W were mostly incorrectly classified as an R with 39,39,33,31 and 18 times respectively. The other way around, R incorrectly classified U and V most often 35 and 21 times respectively. For the similarities in the pictures see Appendix 2.

Table 1: Visualized classification report with the best hyperparameter settings. (CONFUSION MATRIX)



**Model 2**

**Random forest**

Decision trees is on the basis of random forest. Decision trees is a supervised learning model and is often used for classification tasks (James et al., 2013). The architecture of the decision tree is comparable with a regular tree; roots, nodes, branches and leaves (Sato & Tsukimoto, 2001). The root node is the starting point of the tree and splits every node with recursive partitioning based on the largest information gain. At each coming leaf node, the data splits into partitions based on the feature with the second largest information gain. This process is done until a base case is reached or when the label is small enough.

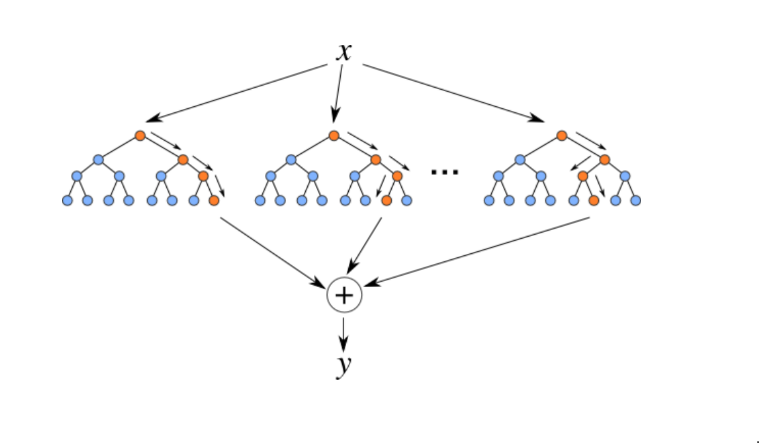
Although decision trees are transparent and easily interpretable, this model overfits the data quite often. To overcome overfitting, random forests are introduced. Random forests differentiate on decision trees by generating multiple trees. The process of generating multiple trees is done by random sampling with replacement of training data and random subsets of features when splitting nodes (Koehrsen, 2018). The random selection of features in each node allows variables to enter the model that otherwise would be overlooked (Strobl et al., 2008). The model is then trained with these random samples and subsets to decrease the total variance. When the predictions are made the bagging approach is applied. Therefore different features combinations could generate different predictions, the final prediction is based on a majority vote.

Figure 2: Visualized random forest classifier.

**Hyperparameter tuning**

The input of the classifier is the same as from the former model, therefore it will not be discussed. To increase performance of random forest hyperparameter can be tuned in multiple ways. The number of decision trees that comprise the random forest. The number of features available for splitting each tree node. The maximal size of terminal nodes.

**Model training**

RandomForestClassifier

Sklearn.tree

**Results**

**Comparison between models**

# Task 2

Description of your experiments and results including:

* features used
* learning model and algorithm used
* parameter tuning
* description of method or system built to perform classification

discussion of experiments run and performance of your solution (this may include analysis such as confusion matrix, accuracy per class, etc.)

# Reference

Breiman, L. (2001). Random forests. Machine learning, 45, 5-32.

Gandhi, R. (2018b, July 5). Support Vector Machine — Introduction to Machine Learning Algorithms. Retrieved from https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Koehrsen, W. (2018, December 1). An Implementation and Explanation of the Random Forest in Python. Retrieved from <https://towardsdatascience.com/an-implementation-and-> explanation-of-the-random-forest-in-python-77bf308a9b76

Sato, M. and H. Tsukimoto. 2001. Rule extraction from neural networks via decision tree induction. In Proceedings of the International Joint Conference on Neural Networks.

Strobl, C., Boulesteix, A. L., Kneib, T., Augustin, T., & Zeileis, A. (2008). Conditional variable importance for random forests. BMC bioinformatics, 9, 307.

Scikit-learn (2021). [sklearn.preprocessing](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing).StandardScaler. Retrieved from <https://scikit-learn.org/stable/modules/preprocessing.html>

Souza, J., Matwin, S., Japkowicz, N. (2002). *Evaluating Data Mining Models: A Pattern Language* School of Information Technology and Engineering University of Ottawa.

# Appendices

## 1 Group work

### Task 1

Stefan Winter

* Explaining Sourcetree
* Generating SVM and pipeline

Joost Oudesluijs

* Generating Neural network
* Describing task 1 on paper

Joost Schutte

* Generating random forest

### Task 2

Stefan Winter

Joost Oudesluijs

Joost Schutte

## 2 Images



Image 1: Images of example sign language letters and their labels