Traffic Signs classification

Andrea Laruelo, Nov 2017

Introduction

There is a high probability that a driver misses some of the traffic signs on the road, due to many different reasons: overcrowding of neighbouring vehicles, lack of concentration,… This may be problematic and can, eventually, lead to car accidents. With the continuous growth of vehicle numbers around the world, this problem is only expected to grow.
Traffic Sign Recognition Systems aim to detect and identify road signs. These systems can be implemented on the automobile in order to ensure safe traffic flow by informing road users about regulations, directions, warnings or potential risks on the road. The types of signs that could be displayed to warn the driver could be for example, stop, "closed road", or speed limits. ,
This field is getting considerable interest, as shown by the vast amount of publications (see Table below). The interest is driven by the emerging market for intelligent applications such as advanced driver assistance systems (ADAS) [1] and autonomous driving [2]. In addition, the recent releases of large traffic signs datasets such as Belgian [3] or German [4] datasets have allowed the comparison/benchmarking of different approaches.
Traffic sign recognition covers two problems: traffic sign detection (TSD) and traffic sign classification (TSC). TSD is meant for the accurate localization of traffic signs in the image space, while TSC handles the labeling of such detections into specific traffic sign types or subcategories. For TSD and TSC numerous approaches have been developed. A recent survey of such methods and existing datasets is given in [5]. This study is focused on traffic signs classification.

Data

Traffic sign recognition is a well-studied problem, and there are several datasets available. We choose the German Traffic Sign Dataset because it is very complete and small enough to be feasible to work with. The dataset can be downloaded from http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset.
This dataset is provided by the Institut fuer Neuroinformatik of the Ruhr-Universitaet Bonn. It contains over 50.000 RGB images (40.000 for training and 12.000 for testing) of 43 different street signs. The images are in .ppm format and contain one traffic sign each. Image sizes vary from 15x15 to 250x250 pixels and images are not necessarily square.
The images and the related features are delivered in 43 subdirectories, named sequentially from 00000 to 00042. The directory names represent the labels, and the images inside each directory are samples of each label.
Image annotations are provided in CSV files containing the filename, the size, the coordinates of the bounding box containing the sign and the class label.

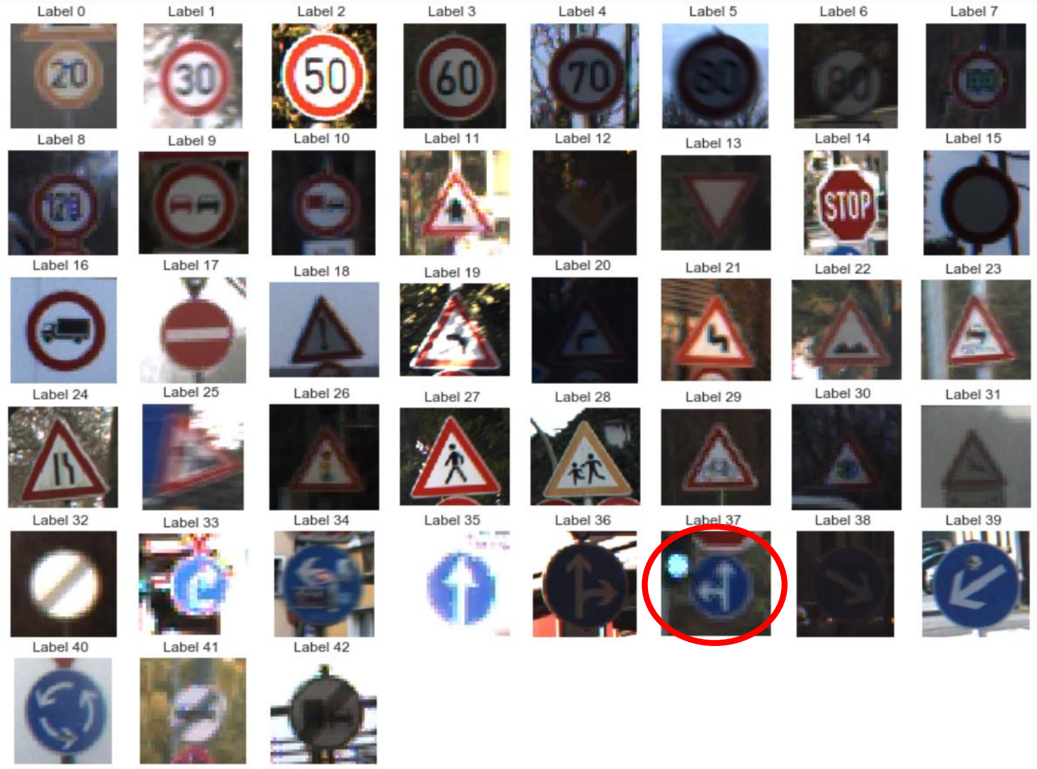
Read/Handle data

Functions to read the training and test images and the pre-processed features have been implemented in a separate notebook (read_capstone_project_data.ipnb).
Reading data and features is a quite long process since it involves the opening and reading of some milliards of files. In order to speed the access to the data, the arrays containig the main features have been stored in HDF5 files. HDF5 is a convenient format to store arrays. In addition, there are software libraries in different languages to handle HDF5 files. For this project, the Python package PyTables has been used. PyTables is built on top of the HDF5 library and uses the NumPy package.

The package can be downloaded here: http://www.pytables.org/downloads.html.

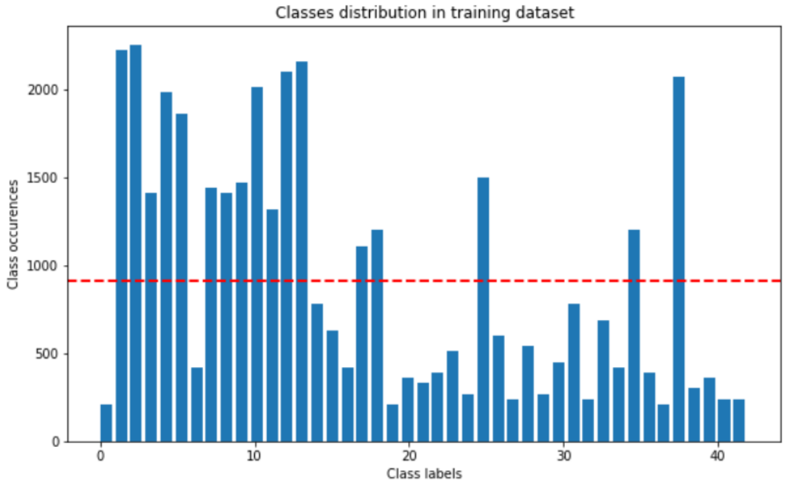
Exploratory Data Analysis

The .ppm images can be read using the imread function from the matplotlib.pyplot module. It returns an MxNx3 numpy.array (RGB images) where MxN is the image size.
In the figure below, we can see a sample image of each label.



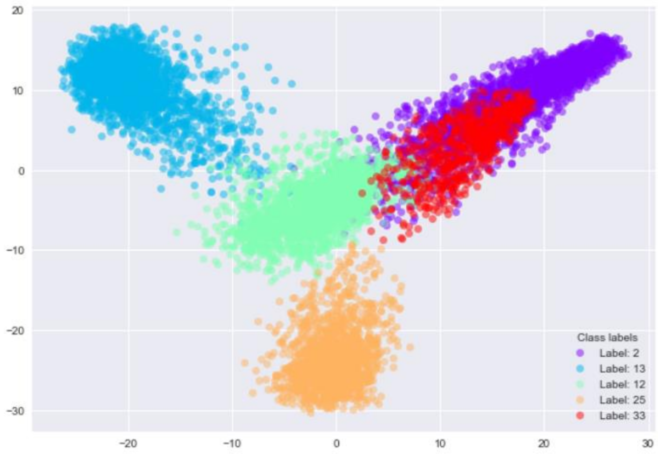
As we can see, the image quality is quite good, and there are a variety of angles and lighting conditions. An important point is that, in most of the cases, the traffic signs occupy most of the area of the image.

As shown in the figure below, the dataset is very unbalanced. The number of images per class ranges from 210 to 2250. This may bias the classifier towards the better represented classes (classes having more samples).



Images are delivered with pre-computed features: HOG histograms, HAAR-like features and HUE histograms.

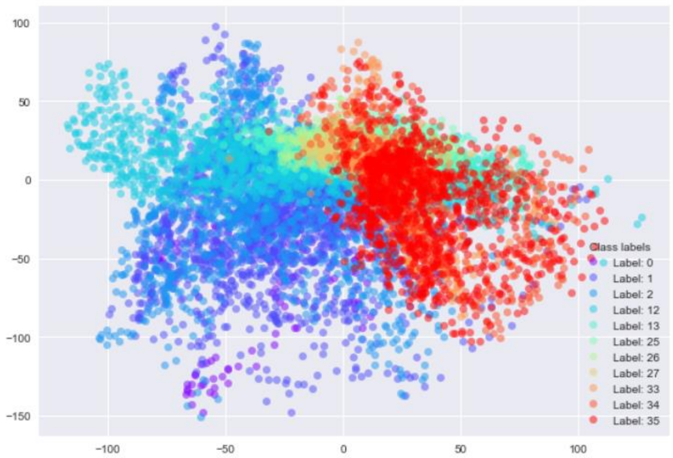
Histograms of Oriented Gradient (HOG) descriptors have been proposed by Dalal and Triggs (2005) for pedestrian detection. Based on gradients of color images, different weighted and normalized histograms are calculated: first for small nonoverlapping cells of multiple pixels that cover the whole image and then for larger overlapping blocks that integrate over multiple cells. In order to visualize and explore the features it is necessary to reduce the dimensionality of the feature space. A basic technique well-suited for this problem is the Principal Component Analysis which finds the directions of most variation in your data set. HOG features seem to provide a good representation of the traffic signs data set. As can be seen in the figure below, the first two principal components provide already a quite clear separation of different sign shapes.



- Labels 0,1,2: speed limit (blue)
- Label 12: diamond (light blue)
- Label 13: inverted down (turquoise)
- Labels 25,26,27: up triangle (green)
- Labels 33,34,35: round blue signals (red)

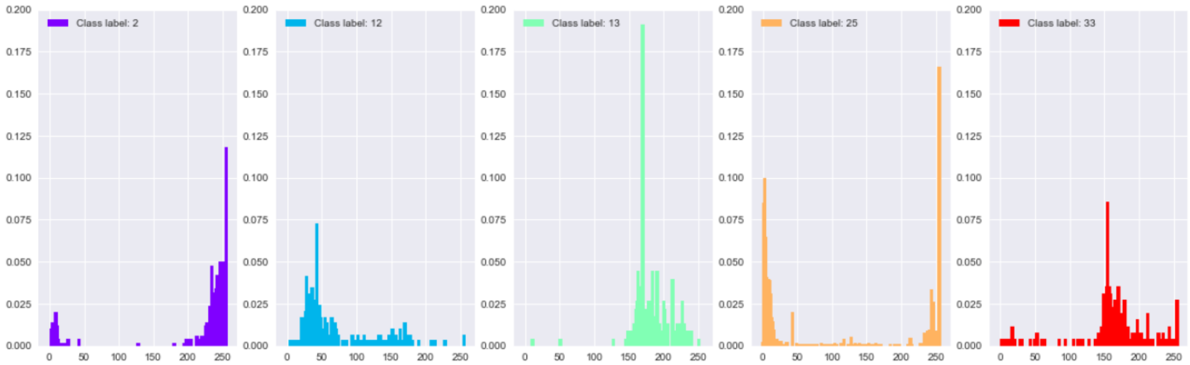
HOG1 features projected on its first two PCA components.

The data set includes the pre-computed Haar-like features. For each image, 5 different types of Haar-like features were computed in different sizes for a total of 12 different features. The overall feature vector contains 11,584 features. The figure below that HAAR features also provide a certain degree of class separation.



HAAR features projected on its first two PCA components.

In addition, for each image in the training set, a 256-bin histogram of hue values (HSV color space) is delivered.



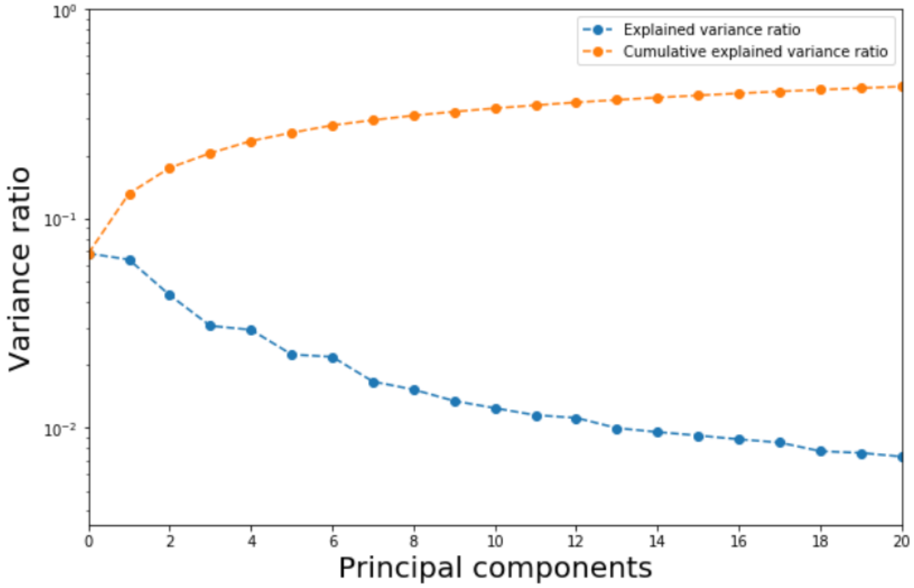
HUE histograms.

Features

In order to be able to train the studies models in a reasonable time (the original number of features is 17892), we have reduced the number of features using a PCA approach. IN addition, we have tested the proposed classification models using the HOG2 features, due to the good classification performance provided by these features in previous works [7].

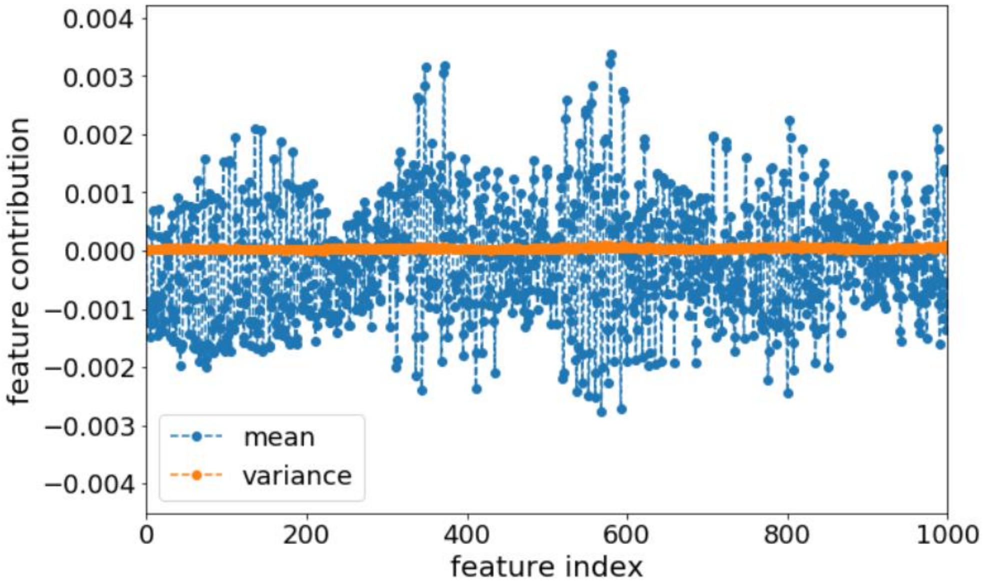
Feature selection using PCA:

The pre-computed features provided within the data set have shown to provide useful insights about the data. However, the total number of features is around 17892. In the previous section we have used PCA in order to reduce the dimensionality of the feature space. PCA can also be used for feature selection purposes, i.e., to identify the 'most important' variables in the original feature space. Within the PCA context, the 'most important' features are the ones that contribute most to the most important principal components. Intuitively, we can imagine that a dimension that has not much variability cannot explain much of the data. However, we should notice that original dimensions that have a much larger variance will have more impact on the principal component. To prevent that features having higher values dominate the PCA, it is highly recommended to normalize the data before applying PCA. This can be easily be done using sklearn.



According to the cumulative explained variance shown in the previous figure, it seems that by keeping only 20 principal components we can describe more than the 80% of the full variance. This drastically reduces the size of the feature space while compromising only 20% of potential information.

In order to identify and keep only features that have big influence in the PCA space, we can apply PCA inversion the identity matrix. We can then measure the contribution of each feature over all principal components by looking at its mean and variance (see figure below).



Given this result we only keep features with largest absolute mean for the classification task.

HOG2 features:

In a second step, we have chosen to perform classification using the HOG2 features. This selection is based in the good results reported using HOG2 features in previous studies [7]. Results from PCA and HOG2 are compared in the next section.

Data (features) standardization

Standardizing the features so that they are centred around 0 with a standard deviation of 1 is an important step when comparing measurements that have different units. It is also a general requirement for many machine learning algorithms, since having features on different scales, certain weights may update faster than others. For example, it is recommended to apply data standardization before applying PCA in order to prevent dimensions with high variance from dominating the PCA. On the other hand, feature standardization does **not** change the overall results of other methods, as it can be the case of Linear Discriminant Analysis, and thus standardization is optional.

Classification models

Linear Discriminant Analysis:

Linear Discriminant Analysis (LDA) is a linear classification technique. The general LDA approach is very similar to a PCA, but in addition to finding the component axes that maximize the variance of our data, it finds the axes that maximize the separation between multiple classes (see figure below).

We have used LDA as a classification model for baseline for comparison. As we can see below, LDA provides surprisingly good results in practice despite its simplicity (Hastie et al., 2001).

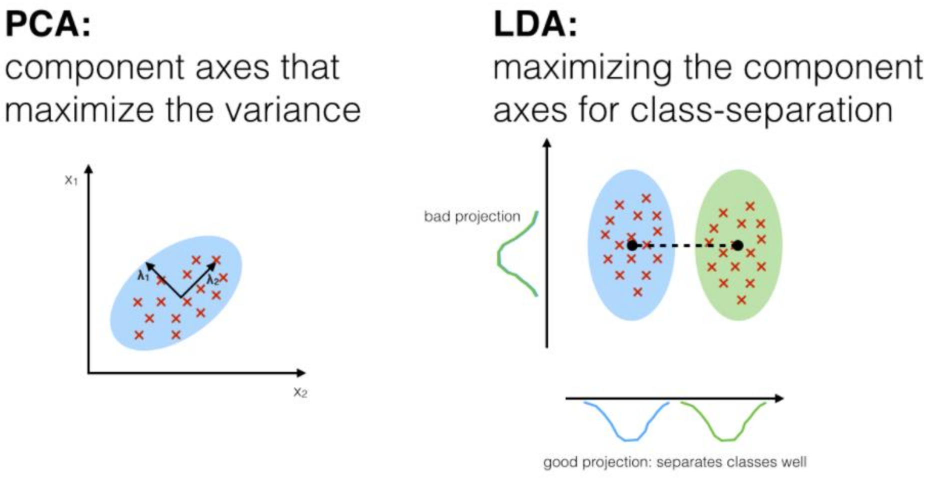
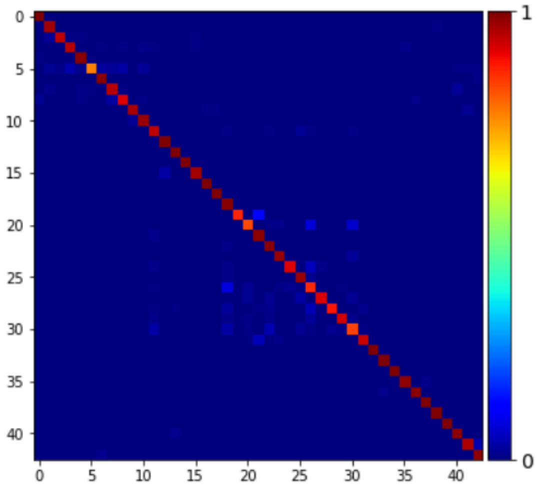

PCA:
component axes that maximize the variance
x_1
λ_1
λ_2
x_2
LDA:
maximizing the component axes for class-separation
bad projection
good projection: separates classes well

Figure from http://sebastianraschka.com/Articles/2014_python_lda.html

Results using HOG2 features:

- Training data set: 98.96%
- Test data set: 95.68%

The good results of the LDA method are shown below by means of a confusion matrix. A confusion matrix summarizes the performance of a classification algorithm. The number of correct and incorrect predictions are shown for each class.



Logistic regression:

Logistic Regression (LR) has shown to be an efficient model for multiclass classification while being much simpler and faster than other classifiers used in state-of-the art approaches, such as convolutional networks (CNN). The results from this approach are comparable to the results from LDA on the HOG2 features. Using the PCA features the results are very bad. This can be explained by the fact PCA finds out which features are important for best describing the variance in a data set, so it may hide features that contribute little to the variance but that are significant class differentiators. In view of this poor results using PCA selected features, the following classification model are also tested on HOG2 features.

Results using PCA:

- Training data set: 12%
- Test data set: 10.6%

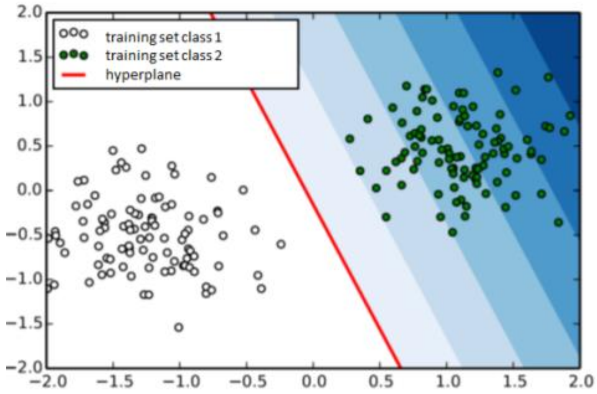
Results using HOG2 features:

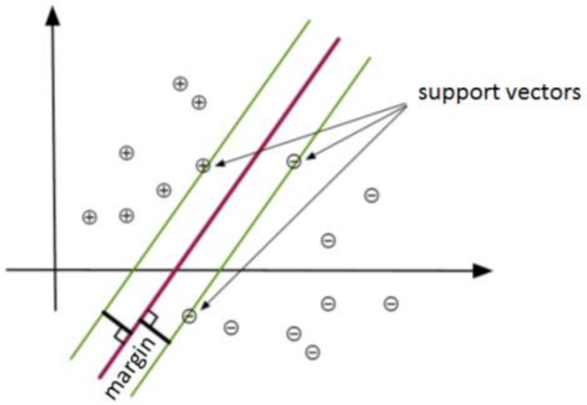
- Training data set: 99.89%
- Test data set: 95.64

In [6] it was already shown that LR based on HOG2 features has an accuracy very similar to the LDA classifier.

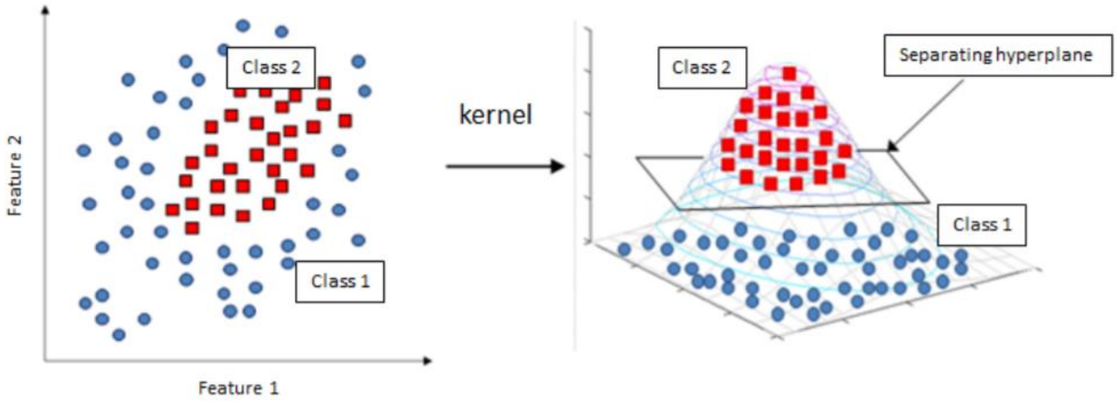
Support Vector Machines:

The basic idea behind support vector machines is illustrated with the example shown in the figure below. Given a training set with points (samples) that belong to two classes, it finds the best hyperplane to differentiate between the two types of points (classes) (i.e. the distance between the hyperplane and the closest training points is maximized). In this example the data is assumed to be linearly separable. Therefore, there exists a linear hyperplane (or decision surface) that separates the points into two different classes. In the two-dimensional case, the hyperplane is simply a straight line. The minimal distance from the separating hyperplane to the closest training point is called margin. The training samples that lie on the margin define the border between the two classes and are referred to as support vectors.





If training data are not linearly separable in the original space, they can be mapped to a higher dimensional space by using what is called a kernel function. By employing kernel transformations to map the samples from their original space into a higher-dimensional feature space, SVM can separate objects which are not linearly separable (Figure 6.8). Kernel-based techniques are less sensitive to the amount of data and the input dimension than other methods (like K-Nearest Neighbour [Samadzadegan et al., 2010] or Linear Discriminant Analysis [Devos et al., 2005]) and are able to detect automatically important characteristics independently of the input pattern [Devos et al., 2005].



Results using HOG2 features:

- Training data set: 85.12%
- Test data set: 80.92%

Results using **standardized** HOG2 features:

- Training data set: 99.98%
- Test data set: 95%

The previous results show that SVM methods are (as expected) very sensitive to the magnitude of the features. Just be standardizing the features, we improve the performance from 80.92% to 95%. The reason behind these results is that SVM maximizes the 'margin' between classes, and thus relies on the concept of between different points. If there are features having different scales/magnitudes, the measured 'distance' may not be meaningful (for classification purposes). Even after standardization of the features, SVM provides slightly worse results than LDA and Logistic Regression. The reason may be that, compared to those approaches, SVM requires a careful hyperparameters estimation. The results could be improved with a more suitable selection of the parameters.

Random Forests:

Random Forests were introduced by Breiman and Cutler. To classify a sample, the classification of each random tree in the forest is taken into account. The class label of the sample is the one with the majority of the votes.

The random forests achieve state-of-the-art performance in many multi-class classification applications. In [Zaklouta et al. 2011], the performance of random forest classifiers is studied for different parametrizations.

Results using HOG2 features:

- Training data set: 99.63%
- Test data set: 96.01%

Random Forests offers the best results compared to the previous methods. The reason for this may be that Random Forests are intrinsically suited for multiclass problems, while SVM is intrinsically two-class. For multiclass problem you will need to reduce SVM into multiple binary classification problems. In addition, there are no really sensitive hyperparameters to tune in Random Forests (except the number of trees; typically, the more trees we have the better). On the contrary, there are a lot of very sensitive parameters to be turned in SVMs, like the kernel or the regularization penalties. Summarizing, SVM requires a much more careful analysis and tend to provide good results if there are fewer outliers in the dataset, while Random forests are less sensitive to hyperparameters and (if there is enough data available) usually come up with a pretty good model.

Future work: Deep learning

As we have seen, traffic sign detection is a challenging task. In this study we have used well known machine learning algorithms that, in some cases, are able to provide quite good results. All these methods rely pre-computed features that may be latter on selected according to different parameters. This approach has shown to be not good enough for complex problems. This has lead to big rise of deep learning approaches. Deep leaning models circumvent the feature extraction step and learn by themselves what features they should focus on. Deep learning has shown to be able to provide very good performance in complex problems such as image classification. Therefore a natural next step would be to use a deep learning approach to classify our traffic signs dataset.

References:

[1] Timofte et al., “Chapter 3.5: Combining traffic sign detection with 3d tracking towards better driver assistance,” in Emerging Topics in Computer Vision and its Applications, C. H. Chen, Ed. World Scientific Publishing, 2011.

[2] Levinson et al., “Towards fully autonomous driving: Systems and algorithms,” in Intelligent Vehicles
Symposium (IV). IEEE, 2011.

[3] Timofte et al., “Multi-view traffic sign detection, recognition, and 3d localisation,” Machine Vision and Applications, 2011.

[4] Stallkamp et al., “Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition,” Neural Networks, no. 0, pp. –, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608012000457

[5] Song et al., “Detection of Stop Signs”, 2008

[6] Pei et al., “Low-Rank Matrix Recovery for Traffic Sign Recognition in Image Sequences” , 2013.

[7] Stallkamp et al., “The German Traffic Sign Recognition Benchmark: A multi-class classification competition”, 2011