Machine Learning
Exam 2016

Nikolaj Dybdahl Rathcke (rfq695)

January 28, 2016

1 The Traffic Sign Recognition Data

1.1 Question 1

The method `makeFreqHistogram` implemented in python in the file `src/q1.py` produces a histogram showing the distribution of class frequencies given the number of bins and the classes of a dataset. Running `main.py`, will give us the histogram over the dataset `ML2015TrafficSignsTrain.csv` found in Figure 1.

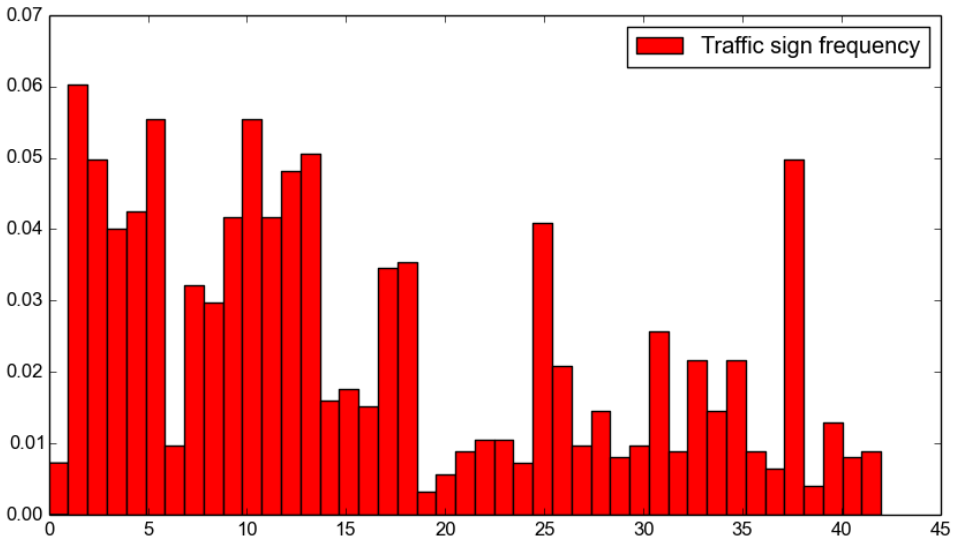

Figure 1: Histogram showing the distribution of class frequencies for the dataset `ML2015TrafficSignsTrain.csv`

The histogram shows us that the number of observations for each class varies quite a bit.

1.2 Question 2

PCA has been implemented in python in `src/q1.py`. The function `pca` takes 2 arguments: The data and the classes corresponding to each entry in the data. Running `main.py` will produce the eigenspectrum and scatter plot for the dataset `ML2015TrafficSignsTrain.csv`. This produces Figure 2 and 3. As a preprocessing step, the data has been normalized to zero mean and variance one.

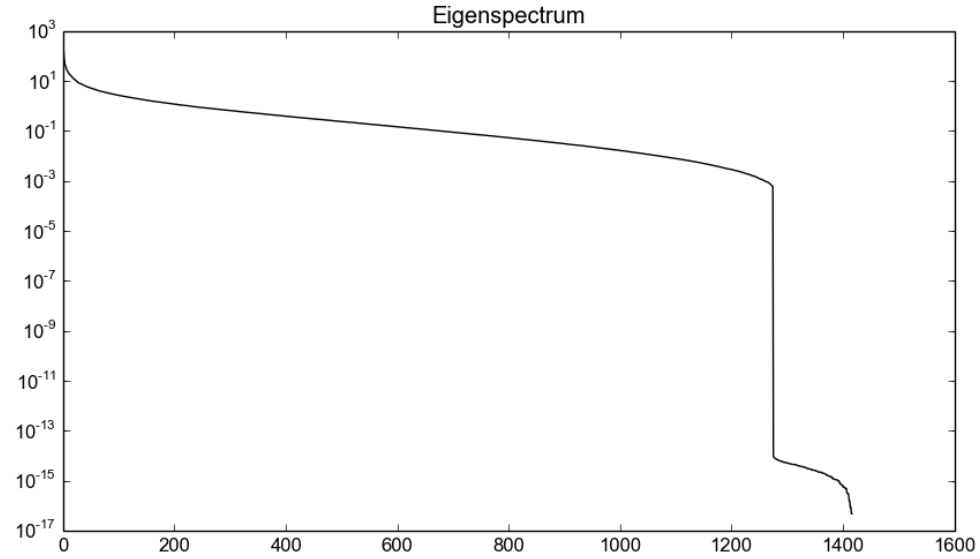

Figure 2: Eigenspectrum after performing PCA on the dataset `ML2015TrafficSignsTrain.csv`

Besides the eigenspectrum, running the main file will also output how many principal components we need to explain 90% of the variance to the terminal:

```
Number of principal components to explain 90% of the variance:
262
```

So we need the 262 most influential principal components to explain 90% of the variance.

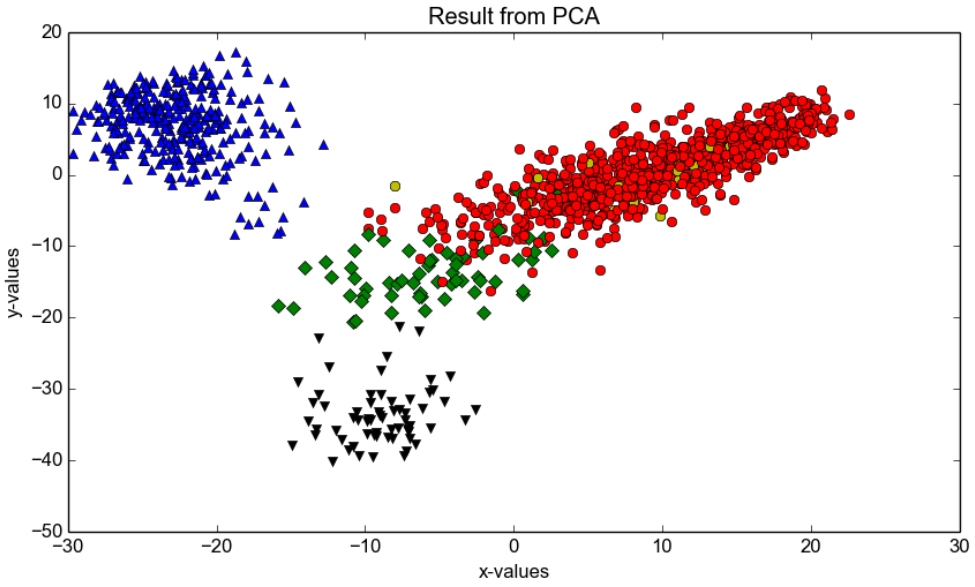

Figure 3: Scatter plot for the first two principal components where the colors (and shape) represent the shape of traffic signs. The scatter plot was generated from the dataset `ML2015TrafficSignsTrain.csv`

The scatter plot shows us somewhat separable data. The yellow octagons are hard to separate from the red circles. This makes sense as a circle and an octagon have a close resemblance to each other. Some of the green square signs are mixed together with some of the circles, but all in all the data is divided quite well using only two principal components.

1.3 Question 3

K-means clustering has been implemented in python and is found in `src/q1.py`, which given a dataset, a k representing the number of centroids and the classes, it will return the k centroids after performing k-means. The function returns when the euclidean distance between the centroids in two consecutive iterations is less than the tolerance set to 0.0001. Running `src/main.py` will produce Figure 4 obtained from the dataset `ML2015TrafficSignsTrain.csv` and $k = 4$. Again, the data has been normalized to zero mean and variance one.

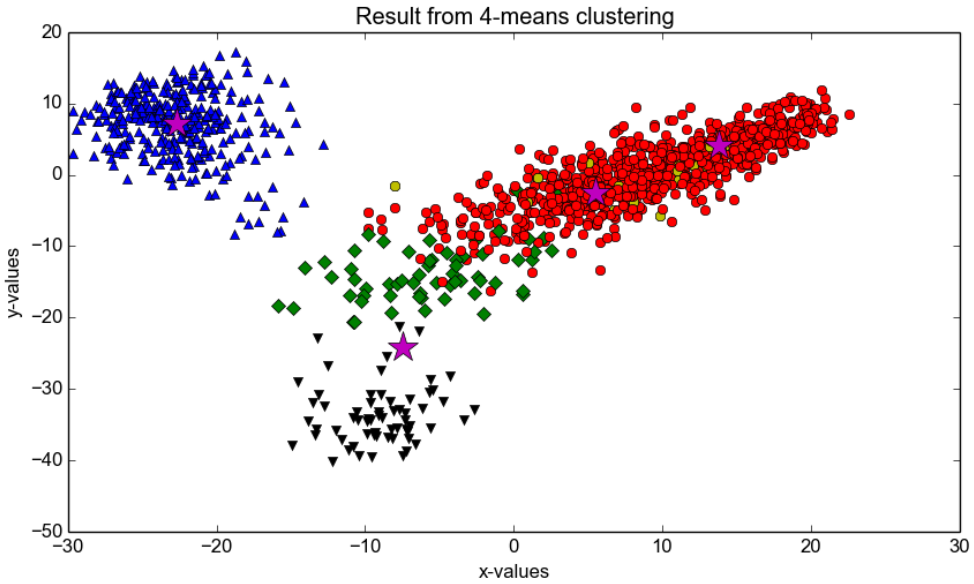

Figure 4: Scatter plot for the first two principal components with the 4 centroids shown as purple stars. The plot is generated from the dataset `ML2015TrafficSignsTrain.csv`

The centroids are represented by the purple stars. If we consider the reduction to two dimensions from PCA as an accurate measure for mutual distances, the centroids makes sense. Obviously, we would like to have a centroid for each shape (and one for circles and octagons), but there are two centroids for the circles/octagons and only one for both downward pointing triangles and diamonds. However, we have a lot of data points for circle shaped traffic signs, which means two of them might have a greater distance to each other than a diamond and a downward pointing triangle. This is also what the scatter plot reflects. Reducing the tolerance did not seem to affect the final centroids.

2 Stars vs. Galaxies

2.1 Question 4

The implementation is made in python and can be found in `src/q2.py` making use of the library `sklearn`. We first compute $\sigma_{Jaakkola}$ using the Jaakkola heuristic. We use $b = 10$ for grid-search as this provided the best results, and we do 5-fold cross validation. The training data has been normalized to zero mean variance one for the training data. The test data has been preprocessed, so that we use the mean and variance from the training data to normalize the test data. Running `main.py` will output the following to the terminal:

```
Gamma value suggested by the Jaakkola Heuristic:
0.461513538631

Best parameters:
{'kernel': 'rbf', 'C': 1000.0, 'gamma': 0.0046151353863122715}

Success rate for training data:
```

```
0.998666666667

Success rate for test data:
0.991333333333
```

So our initial estimate γ from Jaakkola's heuristic is roughly 0.46. The optimal C and γ returned by grid-search is 1000 and 0.0462 respectively. With these parameters, the classification accuracy on the training data is 99.83% and it is 99.13% for the test data. This is quite a small error for both datasets.

3 Normalization

3.1 Question 5

Normalizing to zero mean and variance one before doing logistic regression is not needed, though it can help on the convergence for gradient descent. However, the regression should not change as we only scale the data.
Normalization to zero mean and variance one does nothing for random forests. The ranges do not really matter as a feature is never compared to another feature in terms of magnitude. So normalizing is unnecessary as it will not influence classification accuracy or choice of hyperparameters.

4 Sparse Classifiers

4.1 Question 6

4.1.1

We are working with an infinite amount of hypotheses. Let $|S| = n$, let the expected loss of a hypothesis be $L(h)$ and the empirical error be $\hat{L}(h, S)$. Then by theorem 2.6 in the lecture notes, we have that:

$$\mathbb{P}\left\{\exists h \in \mathcal{H} : L(h) \geq \hat{L}(h,S) + \sqrt{\frac{8\ln(2((2n)^{d_{VC}}+1)/\delta)}{n}}\right\} \leq \delta \tag{1}$$

Now, we know that the VC-dimension for the hypothesis class of linear classifiers in $\mathbb{R}^d$ is $d+1$, so we get the following bound:

$$\mathbb{P}\left\{\exists h \in \mathcal{H} : L(h) \geq \hat{L}(h,S) + \sqrt{\frac{8\ln(2((2n)^{d+1}+1)/\delta)}{n}}\right\} \leq \delta$$

Another way to read it is that we with probability at least $1-\delta$ for all $h \in \mathcal{H}$ have that:

$$L(h) \leq \hat{L}(h,S) + \sqrt{\frac{8\ln(2((2n)^{d+1}+1)/\delta)}{n}}$$

This means that no matter what h is returned from the algorithm, we can with high probability have a guarantee on the expected performance.

4.1.2

The coefficient d should be relatively small compared to n so the bound is not too loose. The smaller d compared to n, the tighter the bound.
To avoid non-trivial information, we let $\delta = 0.01$ to predict something with probability 99%. If we want a non-trivial information about the generalization error, we want the square root term to be less than 1. If we ignore the constants (including δ) in the square root term, we can derive the following relation between d and n:

$$\mathcal{O}\left(\sqrt{\frac{8\ln(2((2n)^{d+1}+1)/\delta)}{n}}\right) = \sqrt{\frac{\ln(n^d)}{n}} \qquad \text{(Removing all constants)}$$

We want this to be less than 1, so:

$$\begin{aligned}
&\sqrt{\frac{\ln(n^d)}{n}} < 1 \\
&\Leftrightarrow \frac{\ln(n^d)}{n} < 1 && \text{(Squaring both sides)} \\
&\Leftrightarrow \ln(n^d) < n \\
&\Leftrightarrow d\ln(n) < n && \text{(As we can assume } d, n > 0\text{)} \\
&\Leftrightarrow d < \frac{n}{\ln n}
\end{aligned}$$

So we want d to be smaller than $\frac{n}{\ln n}$.

4.1.3

Again we will use theorem 2.6 from the lecture notes with $d_{VC} = 2$ as we are working in one dimension. However, there is a slight difference as we need to select a dimension as well. Thus, we have to use a union bound to provide an upper bound. Since we believe any dimension is as good as any other, we use δ_i to denote a distribution of our confidence budget δ among all subspaces $\mathcal{H}_{\{i\}}$, so $\delta_i = \frac{\delta}{d}$:

$$\begin{aligned}
&\mathbb{P}\left\{\exists h \in \bigcup_{i=1}^{d} \mathcal{H}_{\{j\}} : L(h) \geq \hat{L}(h,S) + \sqrt{\frac{8\ln(2((2n)^2+1)/(\delta/d))}{n}}\right\} \\
&\leq \sum_{i=1}^{d} \mathbb{P}\left\{\exists h \in \mathcal{H}_{\{i\}} : L(h) \geq \hat{L}(h,S) + \sqrt{\frac{8\ln(2((2n)^2+1)/\delta_i)}{n}}\right\} \\
&\leq \sum_{i=1}^{d} \delta_i && \text{(Using inequality 1)} \\
&= \sum_{i=1}^{d} \frac{\delta}{d} && \text{(Substitute back for } \delta_i\text{)} \\
&= \delta
\end{aligned}$$

Last step is since we can replace the sum with d and remove both d terms in the fraction. This gives us a very nice and simple bound for learning with $\mathcal{H}_1^d$.

4.1.4

Again we want the square root term to be less than 1. We start by removing all the constants:

$$\mathcal{O}\left(\sqrt{\frac{8\ln(2((2n)^2+1)/(\delta/d))}{n}}\right) = \sqrt{\frac{n^2 d}{n}}$$

And find the relation between n and d:

$$\begin{aligned}
&\sqrt{\frac{\ln(n^2 d)}{n}} < 1 \\
&\Leftrightarrow \frac{\ln(n^2 d)}{n} < 1 \\
&\Leftrightarrow \ln(dn^2) < n \\
&\Leftrightarrow dn^2 < e^n \\
&\Leftrightarrow d < \frac{e^n}{n^2}
\end{aligned}$$

Which means we want d to be at least less than e^n/n^2.

4.1.5

The proof follows the same lines as the proof from (4.1.3), but with a slightly different selection process. Like before, we want to distribute our confidence budget. Since we now pick 2 dimension, we must distribute it to $\binom{d}{2}$ subspaces. Therefore we let $\delta_i = \frac{\delta}{\binom{d}{2}}$ and now we have $d_{VC} = 3$. We also use a nested union bound now, so we get:

$$\begin{aligned}
&\mathbb{P}\left\{\exists h \in \bigcup_{i=1}^{d}\bigcup_{j=i+1}^{d} \mathcal{H}_{\{i\}} : L(h) \geq \hat{L}(h,S) + \sqrt{\frac{8\ln(2((2n)^3+1)/(\delta/\binom{d}{2}))}{n}}\right\} \\
&\leq \sum_{i=1}^{d}\sum_{j=i+1}^{d} \mathbb{P}\left\{\exists h \in \mathcal{H}_{\{i\}} : L(h) \geq \hat{L}(h,S) + \sqrt{\frac{8\ln(2((2n)^3+1)/\delta_i)}{n}}\right\} \\
&\leq \sum_{i=1}^{d}\sum_{j=i+1}^{d} \delta_i && \text{(Using inequality 1)} \\
&= \sum_{i=1}^{d}\sum_{j=i+1}^{d} \frac{\delta}{\binom{d}{2}} && \text{(Rewrite } \delta_i\text{)} \\
&= \delta
\end{aligned}$$

Last step follows from simple combinatorics, as the double sum is the same as the triangular number to $d-1$ which is equal to $\binom{d}{2}$, and this gives us a bound for learning with $\mathcal{H}_2^d$.

4.1.6

We want the square root term to be less than 1. We start by removing the constants:

$$\mathcal{O}\left(\sqrt{\frac{8\ln(2((2n)^3+1)/(\delta/\binom{d}{2}))}{n}}\right) = \sqrt{\frac{n^3d^2}{n}}$$

The d^2 term comes from $\mathcal{O}\left(\binom{d}{2}\right) = d^2$. We can now find the relation between n and d:

$$\begin{aligned}
&\sqrt{\frac{\ln(n^3d^2)}{n}} < 1 \\
&\Leftrightarrow \frac{\ln(n^3d^2)}{n} < 1 \\
&\Leftrightarrow \ln\left(n^3d^2\right) < n \\
&\Leftrightarrow n^3d^2 < e^n \\
&\Leftrightarrow d < \sqrt{\frac{e^n}{n^3}}
\end{aligned}$$

Which means we want d to be at least less than $\sqrt{\frac{e^n}{n^3}}$.

4.1.7

The proof follows the same lines as the proof (4.1.5). This time, however, we do not know how many features we are looking at. To distribute our confidence budget, we let $\delta_i = \frac{\delta}{\binom{d}{\|h\|_0}}$. This means our VC-dimension is $\|h\|_0 + 1$. The union bound is a bit tricky. Since the bound essentially comes down to a counting problem, let $\bigcup_{\|h\|_0}$ denote the $\|h\|_0$ nested unions $\bigcup_{i=1}^{d}\bigcup_{j=i+1}^{d}\dots$ and so on. Similarly, let

$\sum_{\|h\|_0}$ denote the nested sums. Then we can derive the following bound:

$$\begin{aligned}
&\mathbb{P}\left\{\exists h \in \bigcup_{\|h\|_0} \mathcal{H}_{\{i\}} : L(h) \geq \hat{L}(h,S) + \sqrt{\frac{8\ln(2((2n)^{\|h\|_0+1}+1)/(\delta/\binom{d}{\|h\|_0}))}{n}}\right\} \\
&\leq \sum_{\|h\|_0} \mathbb{P}\left\{\exists h \in \mathcal{H}_{\{i\}} : L(h) \geq \hat{L}(h,S) + \sqrt{\frac{8\ln(2((2n)^{\|h\|_0+1}+1)/\delta_i)}{n}}\right\} \\
&\leq \sum_{\|h\|_0} \delta_i && \text{(Using inequality 1)} \\
&= \sum_{\|h\|_0} \frac{\delta}{\binom{d}{\|h\|_0}} && \text{(Rewrite } \delta_i\text{)} \\
&= \delta
\end{aligned}$$

The last step follows from combinatorics again as $\binom{d}{\|h\|_0}$ is the same as counting the nested sums, so they cancel each other out.

4.1.8

Again, we start by taking O-notation of the square root term:

$$\mathcal{O}(\sqrt{\frac{8\ln(2((2n)^{\|h\|_0+1}+1)/(\delta/\binom{d}{\|h\|_0}))}{n}}) = \sqrt{\frac{\ln(n^{\|h\|_0}\binom{d}{\|h\|_0})}{n}}$$

We want it to be less than 1, so we get:

$$\begin{aligned}
&\sqrt{\frac{\ln\left(n^{\|h\|_0}\binom{d}{\|h\|_0}\right)}{n}} < 1 \\
&\Leftrightarrow \ln\left(n^{\|h\|_0}\binom{d}{\|h\|_0}\right) < n \\
&\Leftrightarrow \ln\left(n^{\|h\|_0} d^{\|h\|_0}\right) < n && \text{(We use that } \tbinom{m}{k} \leq m^k\text{)} \\
&\Leftrightarrow (dn)^{\|h\|_0} < e^n \\
&\Leftrightarrow \|h\|_0 \ln(dn) < e^n \\
&\Leftrightarrow \|h\|_0 < \frac{e^n}{\ln(dn)}
\end{aligned}$$

So $\|h\|_0$ has to be less than $e^n/\ln(dn)$.

4.1.9

We start by taking O-notation of the square root term assuming $\|h\|_0 = 1$:

$$\mathcal{O}(\sqrt{\frac{8\ln(2((2n)^2+1)/(\delta/\binom{d}{1}))}{n}}) = \sqrt{\frac{\ln(n^2\binom{d}{1})}{n}}$$

We now find a relation between n and d to ensure it is smaller than 1:

$$\begin{aligned}
&\sqrt{\frac{\ln\left(n^2\binom{d}{1}\right)}{n}} < 1 \\
&\Leftrightarrow \ln\left(n^2\binom{d}{1}\right) < n \\
&\Leftrightarrow \ln\left(n^2 d\right) < n && \text{(Since } \tbinom{d}{1} = d\text{)} \\
&\Leftrightarrow n^2 d < e^n \\
&\Leftrightarrow d < \frac{e^n}{n^2}
\end{aligned}$$

Which means d must be smaller than the e^n/n^2 for the information on the generalization error to be non-trivial.

4.1.10

Comparing the previous result with the result from (4.1.2), we see that $n/\ln(n)$ is a lot smaller than e^n/n^2. The bound on $\|h\|_0$ also enables d to be larger than $n/\ln(n)$. This means that by working with sparse classifiers, we can work with more features (d) and still get a bound that is not trivial.

4.1.11

Looking at the bound from (4.1.7), we see that increasing both n and d will drive number of features we can pick down. This is because the entire square root term will increase as well. Increasing δ however, will drive the number of features up as the square root term will be smaller. This makes sense, as δ is an indication of how certain we want to be. So if we want to be less certain that the bound is correct, we can make a tighter bound, which in turn allows us to pick more features out of d, while a looser bound will limit how many features we can pick.