Report for Week 5 & 6

Create your own "Google Similar Images" system

Patrick de Kok (5640318) Georgios Methenitis (10407537)

December 18, 2013

1 Implement k-NN classifier

As asked for, the most similar images to the four query images `goal/15.jpg`, `bicycle/37.jpg`, `beach/31.jpg` and `mountain/37.jpg` are drawn in the report. The images can be found in Figure 1.

2 Different k-NN strategy

A possible alternative to this simple voting mechanism will be by weighing each vote based on its rank. Each vote with rank r will be assigned a weight

$$w_r = \frac{1}{1+r}.$$

Images with a higher rank should be considered as more important than lower ranked images. The addition of 1 to the rank in the denominator lets the first vote have less relative significance to the second and third votes, such that $w_1 < w_2 + w_3$.

Another weighing mechanism would be based on the global label frequency. Votes for each label would be weighed by:

$$w_\ell = \frac{1}{\#\ell}$$

where $\#\ell$ represents the frequency of label ℓ occurring among the training images. One might want to normalize these weights before they are used, such that

$$\bar{w}_\ell = \frac{w_\ell}{\sum_{\ell \in L} w_\ell}.$$

Here, L is the set of all labels. This ensures that

$$\sum_{\ell \in L} w_\ell = 1.$$

By weighing votes of very frequent labels less and very infrequent labels more, we tell the system that when a rare (or common) label is suspected to occur in the image this is very (or less) significant than expected.

With training sets of limited size, one is often interested in applying a smoothing method over these weights, such as Good-Turing smoothing. This

(a) Query image `goal/15.jpg`.

(b) Query image `bicycle/37.jpg`.

(c) Query image `beach/31.jpg`.

(d) Query image `mountain/37.jpg`.

Figure 1: Query images and their $k = 9$ nearest neighbours with labels.

will even out the pure chance occurance of very frequent labels and very infrequent labels. Because this dataset has equal frequencies for all labels, this method is not applicable, and only the first method is implemented.

The accuracy for the original k-NN implementation over the four queries is 0.5, while the alternative implementation has an accuracy of 0.25. When inspecting the nearest neighbours of the four queries visually, it becomes clear that the histogram intersection distance is not appropriate for this domain. The best label is most often not the first returned by k-NN.

3 Implement mean class accuracy

The accuracy evaluation measure has been implemented, and both k-NN label selection mechanisms are evaluated with $k = 9$. The accuracies of both methods per image class and the mean accuracy can be found in Table 1. Overall both methods have difficulties classifying images correctly.

4 Cross validation of k

The mean class accuracy for each value of $k \in \{1, 3, 5, 7, 9, 15\}$ is reported in Table 2. This has only been computed for the label selection mechanism as meant in the first question. With $k = 9$, you will get the highest mean accuracy.

Class	Original k-NN	Alternative k-NN
Airplane	0.5	0.2
Beach	0.1	0.1
Bicycle	0.8	0.7
Boat	0.2	0.3
Car	0.4	0.2
Flower	0.4	0.3
Goal	0.1	0.2
Mountain	0.3	0.2
Temple	0.3	0.3
Train	0.0	0.1
Mean accuracy	0.31	0.26

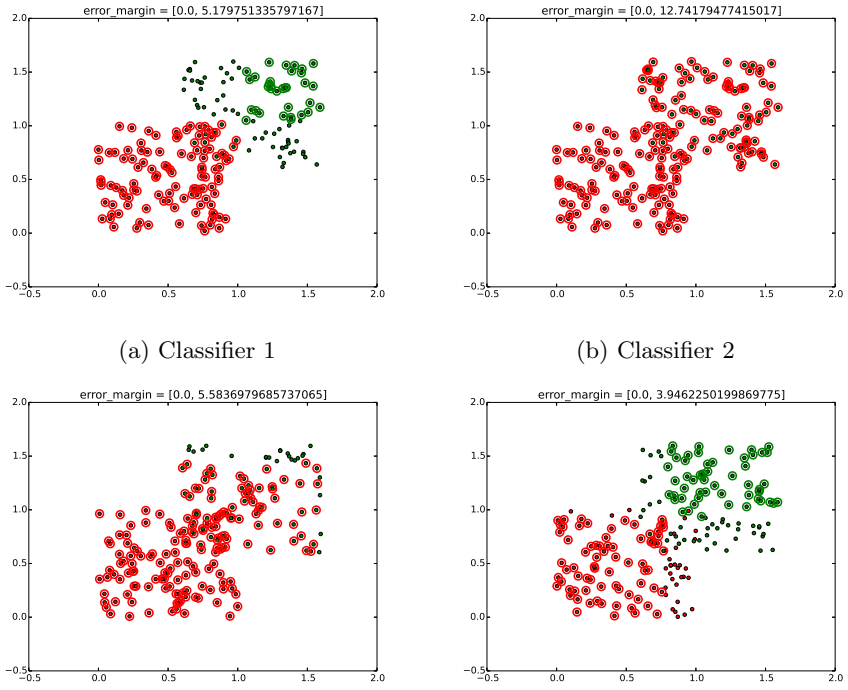
Table 1: Class accuracy per class and the mean accuracy over the complete test set with the original k-NN.

Label	$k = 1$	$k = 3$	$k = 5$	$k = 7$	$k = 9$	$k = 15$
Airplane	0.120	0.262	0.201	0.181	0.222	0.181
Beach	0.312	0.348	0.332	0.257	0.173	0.201
Bicycle	0.459	0.685	0.597	0.623	0.645	0.702
Boat	0.348	0.363	0.363	0.458	0.423	0.423
Car	0.261	0.244	0.283	0.350	0.311	0.333
Flower	0.486	0.399	0.418	0.381	0.401	0.374
Goal	0.156	0.062	0.130	0.124	0.084	0.068
Mountain	0.308	0.153	0.281	0.329	0.446	0.467
Temple	0.160	0.080	0.197	0.192	0.210	0.234
Train	0.097	0.013	0.013	0.013	0.000	0.000
Mean	0.274	0.266	0.284	0.292	0.294	0.292

Table 2: Class accuracies and mean accuracy for different values of k with threefold cross validation. As can be concluded from the mean, the classifier performs best with $k = 9$.

5 Evaluation of 4 possible classifiers

In this part of the assignment we visualize the data obtained by the four different classifiers. Figure 2 shows the resulted predicted labels and the true labels of the four classifiers, as well as the error margin of each one of the classifiers. We can see that the fourth classifier for $classifier = 3$, we obtain the minimum margin error, so we can say that this classifier is the support vector machines.



(a) Classifier 1

(b) Classifier 2

(c) Classifier 3

(d) Classifier 4

Figure 2: Visualization of the four different classifiers. The color of the center of a point indicates its true class. The color of the outer circle indicates the predicted class by the classifier. Error margins are reported above the image for both classes.

6 Use scikit

Now we apply the linear classifier from the `sklearn` package to train it on the provided data and the labels. Using a linear kernel, and $C = 1$ as it comes from default, the resulting predicted labels are shown in Figure 3.

We can see that the predicted class for each data point is in most cases correct, with some faulty prediction only near the boundary between the two classes. This is happening because our classifier uses only one vector, and data from different classes are inside the area of the other class. We cannot overcome this problem with a linear classifier. The data is not linearly separable; there is no straight line to separate both classes perfectly.

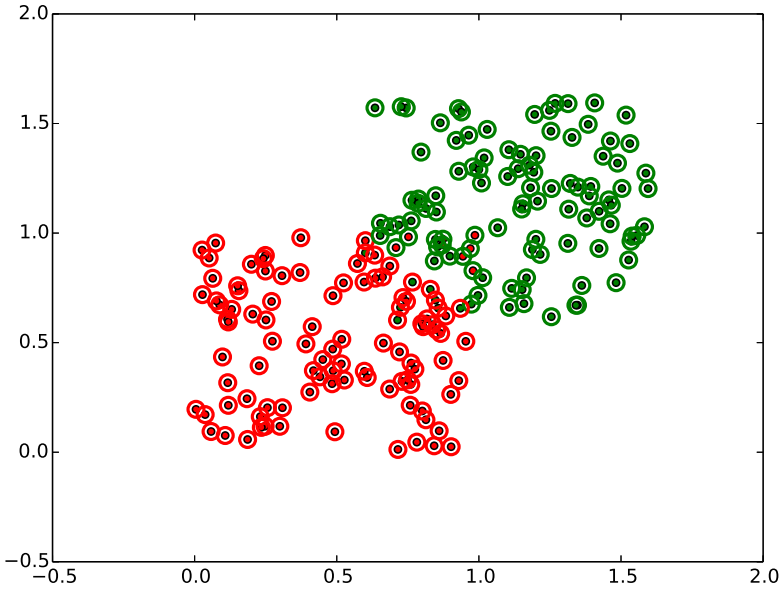


Figure 3: Visualization of the predicted label of the model fitted by a linear SVM classifier.

7 Ring data – linear SVM

We run a linear SVM classifier on the ring data and we have visualized the predicted classes as well as the true class of each data point in Figure 4. The prediction accuracy of about 0.5 is really bad as each predicted class contains only half of the correct class and half of the data of the wrong predicted class. We cannot predict the classes correctly as a linear vector classifier cannot distinguish over data that follow this patterns. This is again because the data cannot be linearly separated. We tried also a non-linear classifier over these data points and the predicted classes were correct as non-linear classifiers can work with this type of data.

The classification results can be seen in Figure 4.

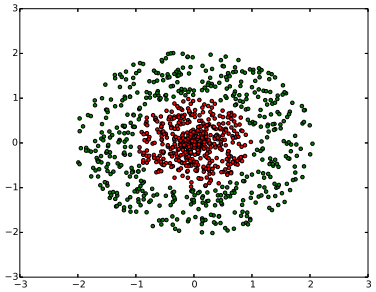
8 Non-linear transformations

The first approach to a non-linear transformation of the data is to represent points by their polar coordinates. Each data point can be transformed as follows:

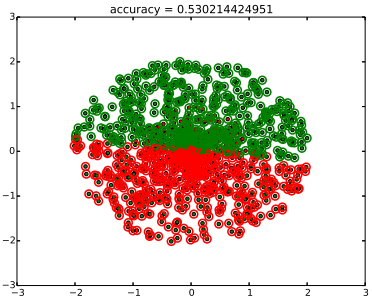
$$p(x, y) \rightarrow p'(\sqrt{x^2 + y^2}, \arctan(y, x)) \tag{1}$$

In our case in which angle does not help us distinguish between the classes as all data points are equally distributed over all angles we could do the same without using the second term like:

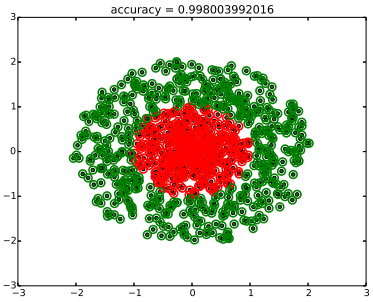
$$p(x, y) \rightarrow p'(\sqrt{x^2 + y^2}) \tag{2}$$



(a) Ring data.



(b) Predicted classes with a linear classifier.



(c) Predicted classes with a non-linear classifier.

Figure 4: Visualization of the predicted label of the ring data model fitted by a linear and a non-linear SVM classifier. The same visualization technique has been used as described at Figure 2.

Now we transformed the two-dimensional dataset to a one-dimensional space, make it feasible and easier than with Eq. 1 to work with a linear classifier with one-dimensional data. As the only thing that we care about is the distance from the point $\langle 0, 0 \rangle$, we only use that piece of information. Another non-linear transformation that we could use in general, but would not be advisable in this case, is the following:

$$p(x, y) \rightarrow p'(x^2 - y^2, 2xy) \tag{3}$$

9 Transform to polar coordinates

In this step we transformed all data into polar coordinates. Figure 5 illustrates the resulted data points after the transformation.

10 Non-linear SVM

Transforming the data into polar coordinates really improves the accuracy from ~ 0.53 to ~ 1.00, which we saw in Figure 4, and has better results than perform-

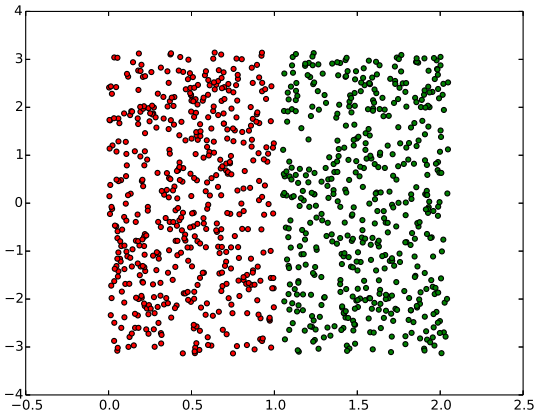


Figure 5: Ring data transformed into polar coordinates.

ing a non-linear classifier over the same data ~ 0.99. Figure 6 shows the resulted predicted classes over the polar data representation and over the original data representation.

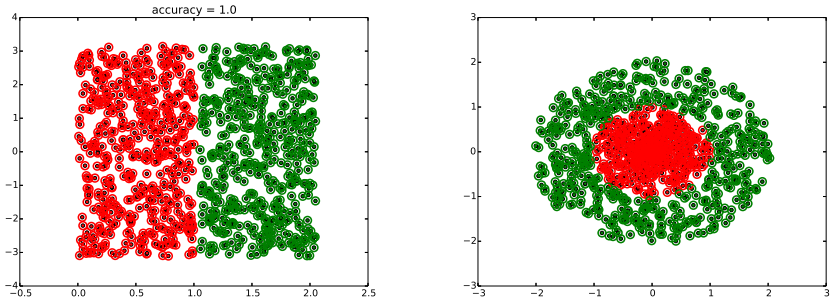


Figure 6: Predicted classes on polar data using a linear SVM classifier, plotted also at their original coordinates.

11 Image classification – linear SVM

The result of threefold crossvalidation on training SVMs with different values for C on the dataset are presented in Table 3. While lower values for C give slightly better accuracies, enlarging the book size has a more drastic effect. The highest mean accuracy is obtained with a booksize of 4000 and $C = 0.1$.

12 Image classification – discussing the results

Class accuracies are shown in Table 4a. A ranking is presented in Table 4b as visualization. One can see that the `mountain` class is classified most often

Book size	$C = 0.1$	$C = 1$	$C = 10$	$C = 1000$
10	0.132	0.126	0.134	0.140
100	0.242	0.240	0.234	0.214
500	0.332	0.326	0.328	0.328
1000	0.352	0.340	0.340	0.340
4000	0.370	0.368	0.366	0.362

Table 3: Mean accuracies for different codebook sizes and different values of the SVM parameter C. The results are from training with threefold crossvalidation on the bag-of-words representation of the image dataset.

Label	Class accuracy
airplane	0.1
beach	0.5
bicycle	0.2
boat	0.2
car	0.5
flower	0.5
goal	0.4
mountain	0.6
temple	0.4
train	0.1
Mean accuracy	0.35

(a) Class accuracies.

Rank	Class
1	mountain
2	beach
2	car
2	flower
5	goal
5	temple
7	bicycle
7	boat
9	airplane
9	train

(b) Class ranking.

Table 4: Class accuracies and their ranking of the SVM trained on the complete training set, on 4000 words per data point and with $C = 0.1$.

correctly. The `airplane` and `train` classes seem to be hardest to classify. While inspecting the data sets, it becomes clear that the airplanes and trains are all on heavily varying backgrounds, whereas the mountains have similar colors and shapes in their environment.

13 Image classification – comparison k-NN vs SVM

Table 5 presents the mean accuracies for the optimally tuned SVM (with $C = 0.1$) and k-NN (with $k = 9$). While the k-NN classifier does not benefit from large code books, the SVM does seem to benefit from this.

Book size	SVM with $C = 0.1$	k-NN with $k = 9$
10	0.22	0.22
100	0.31	0.31
500	0.32	0.34
1000	0.37	0.28
4000	0.35	0.26

Table 5: Mean accuracies for the optimally tuned SVM and k-NN.