Foundations of Data Mining
Assignment 2
fdm-ws18-p1

Stefan Kraatz

1 Task description

This experiment targets the influence of the amount of training data on the accuracy of a predefined classification algorithm. For this purpose the nearest neighbour algorithm was chosen. The parameter for this algorithm, besides the selection of training data are:

- the amount of neighbours considered
- the cost function, that determines the weight of the nearest neighbours

The amount of training data to be used in this experiment is 3, 5, 10 members of each class. The feature vector consists of the image edge histogram data for each image and thus represents the shapes displayed on each image. Therefore using it as a class discriminator seems like a logical choice and has been attempted previously (e.g. in [1])

2 Experiment setup

Implementation laguage and libraries

As an implementation language for this experiment, python was used. The choice was mainly influenced by the availability of many data mining functions, implemented in the well known scikit learn library. Besides scikit learn, the following auxiliary libraries where used, mainly due to dependencies of scikit learn and for ease of use in data handling: pandas and numpy.

Test data selection

The strategy for selecting the test data is yet to be improved, as of now, the first n representatives of each class where selected. However, removing all class representatives, that do not have enough (at least 3) nearest neighbours in the same class prior to the selection of the first n, greatly improved the result (see 2.1).

Execution

The number of neighbours was gradually increased for each given amount of training data and the effect of distance based weight for the neighbour class was tested. Thus the parameter set for each test run was.

- the amount of training data
- the amount of neighbours

Each test run was initially executed 10 times. Due to the nature of the chosen algorithm, this did not yield varying results per repetition, and thus results for them were not recorded individually. The effect of the cost function (distance based bias or unbiased) was tested first and the more "attractive" was chosen as default from then on.

2.1 Variations and tuning

After initial test runs, it was decided to also evaluate the influence of the amount of feature points, that were used for testing. For reducing the feature vectors to the optimal set of feature points, the principal component analysis was used [2]. The useful number of features was evaluated with 10 members per class for training the classifier and the achieved accuracy for this observed. The idea behind this experiment is to check, if some feature values do not contribute to the classification (because they may be simply random) and *may* be removed (e.g. to save computation time and memory for the classifier) or if some features may actually decrease the classification accuracy (because the data is unrelated but not random) and *should* be removed to improve classification. However the latter case may not be detectable easily using this method.

test data selection After identifiying the relationship between the amount of training data and the value for `k` and therefore determining the best `k` for each `n` the highest accuracy achieved was around 50%. However this result was only achieved by increasing `n` far beyond the numbers given in the task description. Therefore a suitable strategy for selecting the best `n` class members was desired. Research into this topic revealed, that there exists a broad range of instance selection techniques [3], that aim to improve the quality of the test data selection and therefore also the classification accuracy. Since the classification technique of choice was the KNN Algorithm, it seemed logical to use the Edited Nearest Neighbour Algorithm for the instance selection. This algorithm only considers those data points as representative for a class, that have their closest (configurable) neighbours in the same class. The results for those experiments can be seen in 3.3, 3.4 and 3.5

3 Results

The test results present the achieved accuracy score, given the corresponding parameters, described in each experiment. The number of neighbours was increased by two in each run and the effect observed for each number of class members

3.1 Experiment 1

This setup observes the effect of the (distance)weight function to a otherwise not tuned classifier.

3.1.1 Experiment 1a

description: This experiment simply uses the first `n` class members for the training. No further tuning was attempted. See figure 1 and table 1

results: It is clearly visible, that the accuracy increases, when more training data is provided.

	number of neighbours (`k`)						
`n`	2	4	6	8	10	12	14
3	0.209	0.203	0.205	0.205	0.21	0.206	0.207
5	0.256	0.259	0.261	0.262	0.263	0.259	0.254
10	0.328	0.351	0.36	0.363	0.366	0.365	0.36

Table 1: no tuning, no weight function

$acc_{min} = 0.203$
$acc_{max} = 0.366$

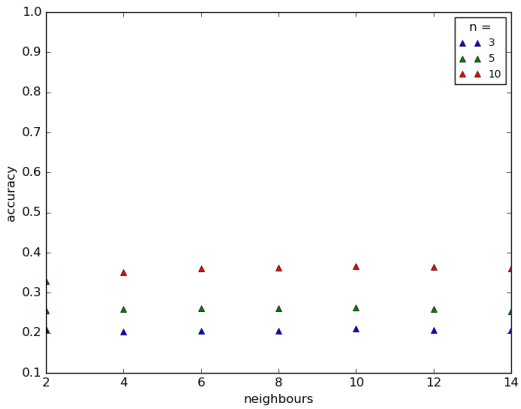

Figure 1: no tuning, no weight function

3.1.2 Experiment 1b

description: This experiment adds the weight function to the classifier. See figure 2 and table 2

results: Adding the weight function clearly raises the accuracy score compared to experiment 1a. Therefore all following experiments will have the distance weight function enabled by default.

	number of neighbours (k)						
n	2	4	6	8	10	12	14
3	0.227	0.229	0.231	0.231	0.233	0.23	0.23
5	0.293	0.301	0.304	0.303	0.302	0.298	0.292
10	0.367	0.383	0.39	0.39	0.392	0.392	0.393

Table 2: no tuning, with weight function

$acc_{min} = 0.227$
$acc_{max} = 0.393$

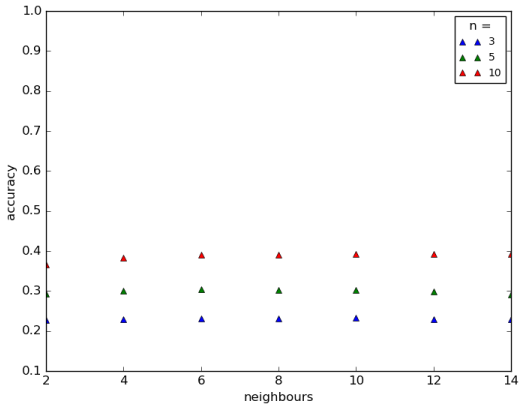

Figure 2: no tuning, with weight function

3.2 Experiment 2

description In this experiment, principal component analysis (PCA) was enabled prior to training the classifier. The previously identified number of features (45) was used here.

3.2.1 Experiment 2a

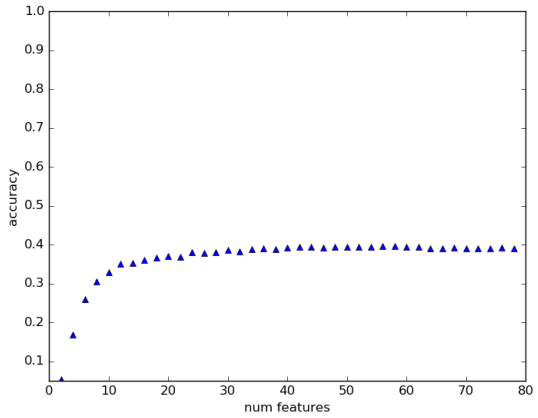

Figure 3: influence of # num features

feature reduction Here we are evaluating the effect of gradually increasing the amount of features used for training the classifier. The PCA implementation in scikit-learn will identify the features with the lowest entropy between the classes and will only return data, with a configurable amount of best features remaining. Prior to using PCA in the next experiment, the effect was observed for identifying the optimal value.

3.2.2 Experiment 2b

results: In experiment 2a, the number of remaining features was found best to be around 45. There was no noticeable increase, when using more features for training the classifier (see figure 3). When enabled however, there was no noticeable difference in the achieved accuracy score, when applying PCA, prior to training the classifier, compared to not using it at all. (see table 3 and figure 4)

	number of neighbours (k)						
n	2	4	6	8	10	12	14
3	0.222	0.223	0.226	0.227	0.228	0.226	0.225
5	0.291	0.299	0.301	0.297	0.296	0.293	0.289
10	0.367	0.389	0.393	0.395	0.392	0.393	0.391

Table 3: pca enabled

$acc_{min} = 0.222$
$acc_{max} = 0.395$

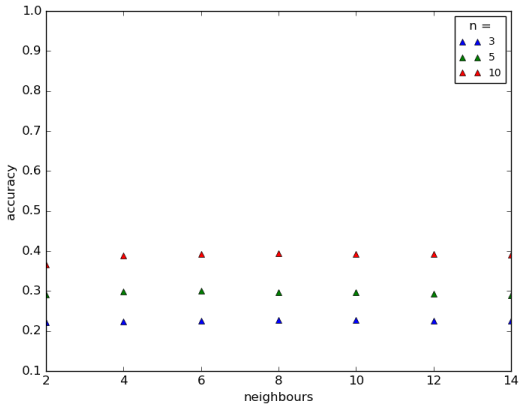

Figure 4: pca enabled

3.3 Experiment 3

description In this experiment, PCA and the edited nearest neighbour (ENN) instance selection were enabled. However, only the indices list of "good" class representatives was used. The test data also contains data points, which would have been removed by the ENN.

results: The results are at the first glance unchanged from those found in experiment 2. However, there are some subtle differences, the values seem more compressed and generally slightly higher than previously except for the maximum values. It is also notable, that the accuracy is now inversely sensitive to a growing k.

	number of neighbours (k)						
n	2	4	6	8	10	12	14
3	0.274	0.264	0.255	0.252	0.251	0.249	0.246
5	0.338	0.321	0.305	0.289	0.275	0.26	0.246
10	0.385	0.375	0.368	0.358	0.348	0.341	0.328

Table 4: with enn with pca

$acc_{min} = 0.246$
$acc_{max} = 0.385$

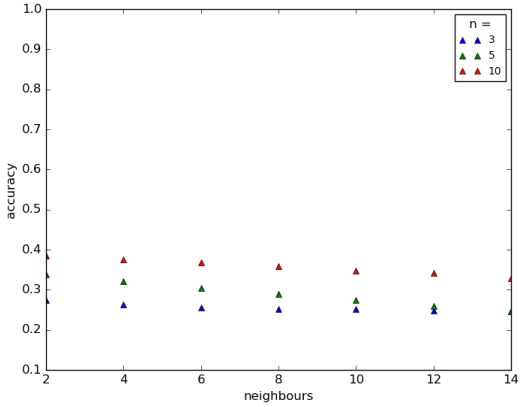

Figure 5: with enn with pca

3.4 Experiment 4

description: for cross checking the effect of using ENN without the principal component analysis was also tested.

results: These results are practically unchanged from those found in experiment 3.

	number of neighbours (k)						
n	2	4	6	8	10	12	14
3	0.277	0.265	0.258	0.253	0.252	0.25	0.247
5	0.336	0.317	0.304	0.285	0.264	0.249	0.236
10	0.374	0.365	0.358	0.348	0.338	0.326	0.313

Table 5: with enn without pca

$acc_{min} = 0.236$
$acc_{max} = 0.374$

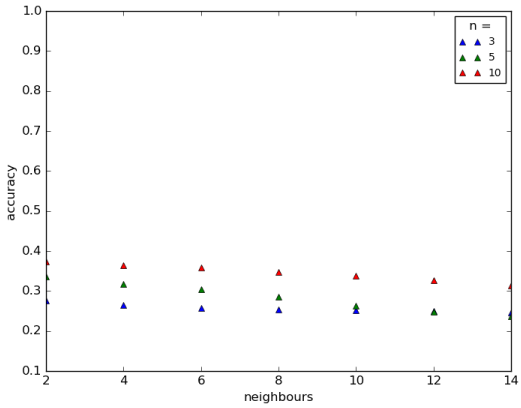

Figure 6: with enn without pca

3.5 Experiment 5

description: In this experiment, PCA was enabled, ENN was enabled and the output of ENN was used for testing as well. This tests how good data points from the "denoised" data set are classified, after using a small subset from it as training data. The overall amount of data was reduced to around 3.500 from almost 10.000 data points. This already shows, that the biggest amount of data is not easily classifiable using the KNN classifier.

results: At the first glance, the results seem very promising, reaching accuracy scores of almost 90% for the series with 10 training candidates. However, when considering the conditions for this result, this is easily explainable. One observation can also be made here: Increasing the amount of k for the KNN classifier decreases the accuracy score more strongly than it did in the previous experiments. This may be the result of the removal of the "noise", making it more likely that more neighbours may be from a different class.

	number of neighbours (k)						
n	2	4	6	8	10	12	14
3	0.772	0.662	0.596	0.517	0.479	0.45	0.425
5	0.834	0.819	0.729	0.685	0.655	0.627	0.603
10	0.881	0.885	0.87	0.857	0.829	0.806	0.795

Table 6: with enn with pca denoised

$acc_{min} = 0.425$
$acc_{max} = 0.885$

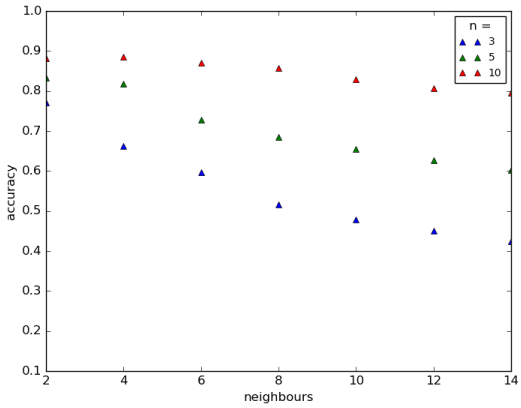

Figure 7: with enn with pca denoised

4 Evaluation

All experiments show, that increasing the amount of training data effectively increases the accuracy of the KNN classifier. The biggest impact, though, has the de-noising of the data, which makes sense insofar, that hardly classifiable data points are already removed by the ENN algorithm. However, practically, this method is probably not an option, since the aim of classification should be to classify unknown data, ideally with a high accuracy. In conclusion it can be said, that the KNN algorithm is not very efficient on the edge histogram data. This seems logical, as shape data may be similar locally but different in another location, potentially making it difficult to decide purely on distance in a large dimensional space, if images are similar or not.However, an accuracy of almost 50%, may under these circumstances be quite sufficient results, though, considering the amount of classes.

References

[1] H. Frigui and P. Gader, "Detection and discrimination of land mines in ground-penetrating radar based on edge histogram descriptors and a possibilistic*k*-nearest neighbor classifier," *IEEE Transactions on Fuzzy Systems*, vol. 17, pp. 185–199, Feb 2009.

[2] S. WOLD, K. ESBENSEN, and P. GELADI, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, pp. 37–52, 1987.

[3] Á. Arnaiz González *et al.*, *Estudio de métodos de selección de instancias*. PhD thesis, Universidad de Burgos. Departamento de Ingeniería Civil, 2018.