Department of Computer Science
Umeå Universitet

October 22, 2018

5DV121
Fundamentals of Artificial Intelligence
Assignment 3
k-NN analysis

Teachers
Carl-Anton Anserud, Anders Broberg
Adam Dahlgren Lindström, Monika Jingar
Thomas Johansson, Jenny Nilsson, Ola Ringdahl

Emil Hallberg id16ehg@cs.umu.se
Jonas Sjödin id16jsn@cs.umu.se

October 22, 2018
Emil Hallberg & Jonas Sjödin

Prerequisites

We tested the diffrent datasets with the seed number 5.

Iris

We chose to use the following features since it gave the highest average value:

```
2 — petal length (cm)
3 — petal width (cm)
```

Breastcancer

We chose to use the following features:

```
20 — worst radius
21 — worst texture
22 — worst perimeter
23 — worst area
24 — worst smoothness
25 — worst compactness
26 — worst concavity
27 — worst concave points
28 — worst symmetry
29 — worst fractal dimension
```

Question 0

For the given dataset, what seems to be an optimal k value? Are there more than one? If so, which should you choose?

Iris

We calculated the optimal k-value by cross validating since we have more than one unknown parameter. We do this by fitting our model as well as possible to the training data. The test scores gave us that the optimal number of neighbors is 3 which is as close to the value itself, meaning that it is almost not underfitted or overfitted. We found that there were no more than one optimal k-value.

Breastcancer

We calculated the optimal k-value by cross validating since we have more than one unknown parameter. We do this by fitting our model as well as possible to the training data. The test scores gave us that the optimal number of neighbors is 9 which is as close to the value itself, meaning that it is almost not underfitted or overfitted. We found that there were no more than one optimal k-value.

October 22, 2018
Emil Hallberg & Jonas Sjödin

Question 1

What type of learning is this? (Super-/unsupervised or reinforcement learning, multi-class, regression or ranking, et c.)

Both the Iris and Breast Cancer tests are supervised learning since we give the program a controlled set of data to work with. The learning is multiclass since we have more than two classes. Having only two classes is called binary classification.

Question 2

Based on your plots, describe the relationship between the training and test error.

Iris

The KNN results (see Figure 1) show that the test score stabilizes at around 98% accuracy and the training score stabilizes at around 95% accuracy. If there are fewer neighbors the relationship between training and test error decreases.

Breastcancer

The KNN results (see Figure 3) show that the test score stabilizes at around 97% accuracy and the training score stabilizes at around 90% accuracy. If there are fewer neighbors the relationship between training and test error decreases.

Question 3

Run the entire procedure twice. Once using a test set of 33% and once using a test set of 66%. Describe the results and how they relate to the terms under-/overfitting.

If we compare the results of the Iris data set in figure 1 and 2 we can see that figure 1 is underfitted since it does not find the relationship between the input and the target values. We can see that on the graph since the test score curve doesnt follow the training score curve. Figure 2 is balanced since the two curves follow each other well. If we compare the results of the Breastcancer data set in figure 3 and 4 we can see clearly the the training results have a lower average score than the test score in both cases. This results in the conclusion that the breast cancer score is underfitting because it doesn't find the relationship between the input and the target values. We can see that on the graph since it doesn't follow the given training score curve at all.

Iris

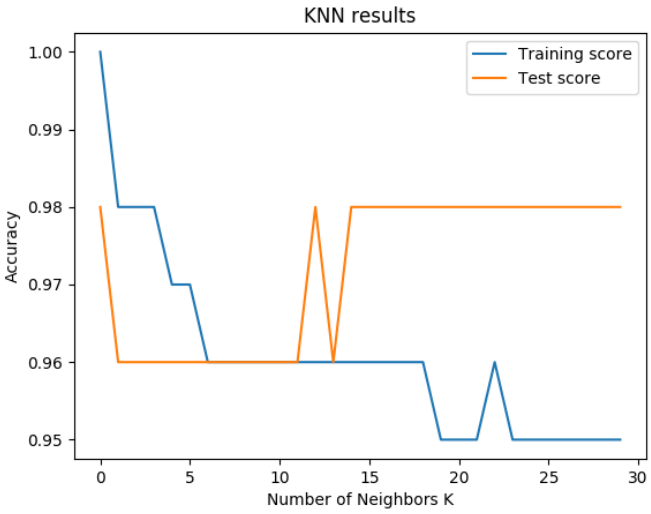

Figure 1 – Iris Analysis Basis 33%

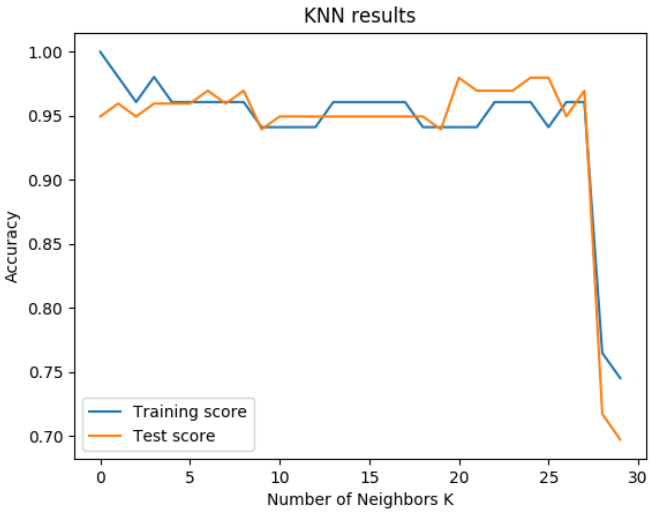

Figure 2 – Iris Analysis Basis 66%

Breastcancer

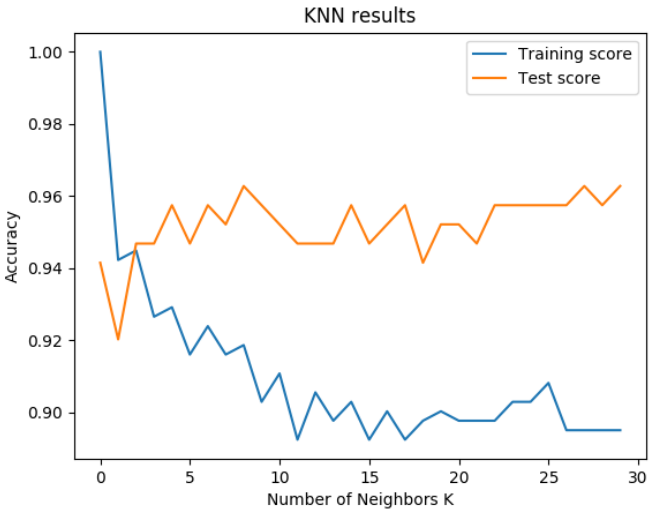

Figure 3 – Breastcancer Analysis Basis 33%

Figure 4 – Breastcancer Analysis Basis 66%

Bilagor

Attachment 1 Source Code

```
# Author: dali@cs.umu.se & Emil Hallberg: id16ehg@cs.umu.se & Jonas Sjodin: id16jsn@cs.umu.se
# Given as part of the course Fundamentals of Artificial Intelligence 2018.

# Import classifier, dataset and data formatting
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.datasets import *

# Import matlab stuff to produce plots
import matplotlib.pyplot as plt


def train_knn(x_train, y_train, k):
    """
    Given training data (input and output), train a k-NN classifier.

    Input:    x/y_train - Two arrays of equal length, one with input data and
              one with the correct labels.
              k - number of neighbors considered when training the classifier.
    Returns:  The trained classifier
    """
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(x_train, y_train)
    return knn

def evaluate_knn(knn, x_train, y_train, x_test, y_test):
    """
    Given a trained classifier, its training data, and test data, calculate
    the accuracy on the training and test sets.

    Input:    knn - A trained k-nn classifier
              x/y_train - Training data
              x/y_test  - Test data

    Returns:  A tuple (train_acc, test_acc) with the resulting accuracies,
              obtained when using the classifier on the given data.
    """
    train_score = knn.score(x_train, y_train)
    test_score = knn.score(x_test, y_test)
    return (train_score, test_score)

def load_dataset(name, features, test_size):
    """
    Loads the iris or breast cancer datasets with the given features and
    train/test ratio.

    Input:    name - Either "iris" or "breastcancer"
              features - An array with the indicies of the features to load
              test_size - How large part of the dataset to be used as test data.
                          0.33 would give a test set 33% of the total size.
    Returns:  Arrays x_train, x_test, y_train, y_test that correspond to the
              training/test sets.
    """
```

```
54      # Load the dataset
55      if name == "iris":
56          dataset = load_iris()
57      elif name == "breastcancer":
58          dataset = load_breast_cancer()
59
60      print('You are using the features:')
61      for x in features:
62          print x,"—", dataset.feature_names[x]
63
64      X = dataset.data[:,features]
65      Y = dataset.target
66
67      # Split the dataset into a training and a test set
68      # generator.
69      return train_test_split(X, Y, test_size=test_size, random_state=5)
70
71  if __name__ == '__main__':
72
73      # Choose features to train on
74      features = range(20, 30)
75      print features
76
77      # The maximum value of k
78      k_max = 30
79
80      # Load the dataset with a test/training set ratio of 0.33
81      x_train, x_test, y_train, y_test = load_dataset('breastcancer', features, 0.66)
82
83      # Lists to save results in
84      train_scores = []
85      test_scores = []
86      te_cv_scores = []
87      tr_cv_scores = []
88
89      # Train the classifier with different values for k and save the accuracy
90      # # achieved on the training and test sets
91      neighbors = range(1, k_max + 1)
92      for k in neighbors:
93          knn = train_knn(x_train, y_train, k)
94          train, test = evaluate_knn(knn, x_train, y_train, x_test, y_test)
95          # new
96          te_scores = cross_val_score(knn, x_test, y_test, cv=10, scoring='accuracy')
97          te_cv_scores.append(te_scores.mean())
98          # end new
99          train_scores.append(train);
100         test_scores.append(test);
101
102     # Construct plot
103     plt.title('KNN results')
104     plt.xlabel('Number of Neighbors K')
105     plt.ylabel('Accuracy')
106
107     # Create x—axis
108     xaxis = [x for x in range(k_max)]
109     # Plot the test and training scores with labels
110
111     te_MSE = [1 —x for x in te_cv_scores]
```

```
112     tr_MSE = [1 -x for x in tr_cv_scores]
113
114     s = sum(test_scores) / len(test_scores)
115     print s
116     te_optimal_k = neighbors[te_MSE.index(min(te_MSE))]
117     print "The optimal number of neighbors for the test score is %d" % te_optimal_k
118
119     plt.plot(xaxis, train_scores, label='Training score')
120     plt.plot(xaxis, test_scores, label='Test score')
121
122     # Show the figure
123     plt.legend()
124     plt.show()
```