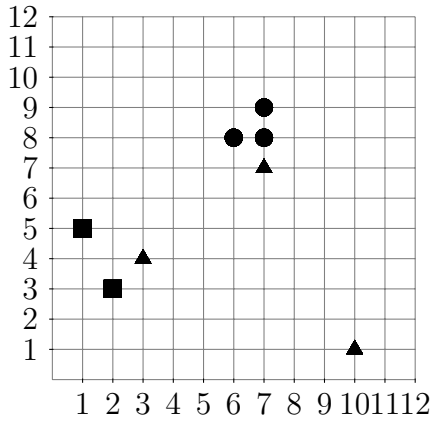
Solutions

Exercise 5 : Clustering : k-means and Silhouette, Evaluation of Classifiers, Probability

Exercise 5-1 : k-means, choice of k, and compactness

Given the following data set with 8 objects (in $\mathbb{R}^2$) as in the lecture :



Compute a complete partitioning of the data set into $k = 3$ clusters using the basic k-means algorithm (due to Forgy and Lloyd). The initial assignment of objects to clusters is given using the triangle, square, and circle markers.

Objects x are assigned to the cluster with the least increase in squared deviations $SSQ(x, c)$ where c is the cluster center.

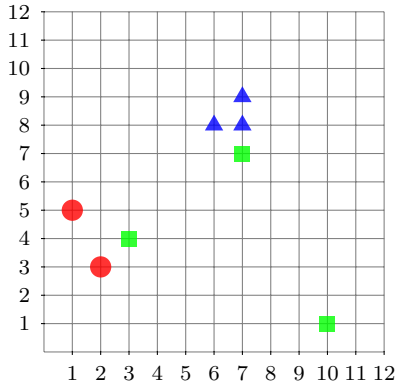
$$SSQ(x, c) = \sum_{i=1}^{d} |x_i - c_i|^2$$

Start with computing the initial centroids, and draw the cluster assignments after each step and explain the step. Remember to use the least squares assignment !

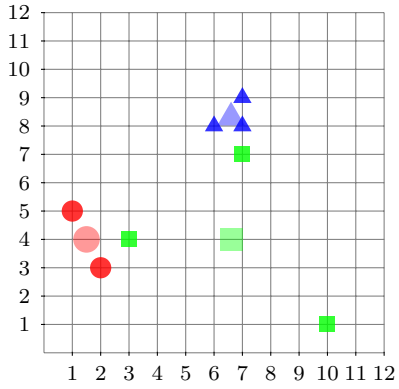
Give the final quality of the clustering (TD^2). How does it compare with the solutions for $k = 2$ discussed in the lecture ? Can we conclude on $k = 3$ or $k = 2$ being the better parameter choice on this data set ?

Also compute solutions with $k = 4$, $k = 5$, starting from some random initial assignments of objects to clusters. What do you observe in terms of the TD^2 measure ?

Suggested solution :

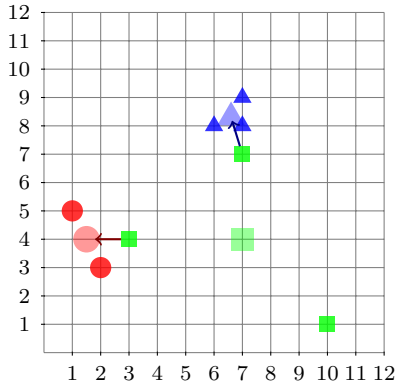


Initial clusters.

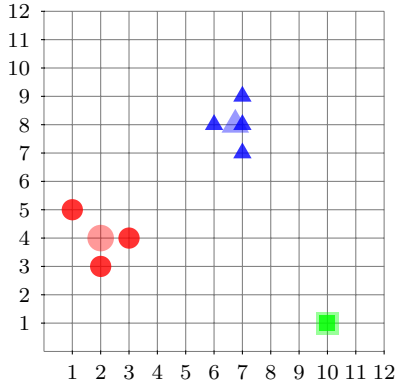


Compute centroids :

- $\mu = (1.5, 4)$
- $\mu \approx (6.6, 8.3)$
- $\mu \approx (6.6, 4)$



Reassign points to closest representant.



Recompute centroids :

- $\mu = (2, 4)$
- $\mu = (6.75, 8)$
- $\mu = (10, 1)$

Then reassignment of points : no change.
Algorithm terminates.

$SSQ(\mu_1, p_1) = |10-10|^2 + |1-1|^2 = 0$
$TD^2(C_1) = 0$

$SSQ(\mu_1, p_2) = |2-2|^2 + |4-3|^2 = 0+1 = 1$
$SSQ(\mu_1, p_3) = |2-3|^2 + |4-4|^2 = 1+0 = 1$
$SSQ(\mu_1, p_4) = |2-1|^2 + |4-5|^2 = 1+1 = 2$
$TD^2(C_2) = 4$

$SSQ(\mu_2, p_5) = |6.75-7|^2 + |8-7|^2 = \frac{1}{16} + 1 = 1\frac{1}{16}$
$SSQ(\mu_2, p_6) = |6.75-6|^2 + |8-8|^2 = \frac{9}{16} + 0 = \frac{9}{16}$
$SSQ(\mu_2, p_7) = |6.75-7|^2 + |8-8|^2 = \frac{1}{16} + 0 = \frac{1}{16}$
$SSQ(\mu_2, p_8) = |6.75-7|^2 + |8-9|^2 = \frac{1}{16} + 1 = 1\frac{1}{16}$
$TD^2(C_3) = 2\frac{3}{4}$
$TD^2 = 6\frac{3}{4}$

In terms of the compactness measure TD^2, this solution with $k = 3$ is much better than any solution with $k = 2$.

However, if we increase k further, the compactness will be even smaller. With $k = 8$, we could get a solution with $TD^2 = 0$, because each point will be identical with its cluster. Optimizing compactness alone is therefore not good enough to find the optimal number of clusters.

Exercise 5-2 : Measure for Evaluation of Classifiers

Given a data set with known class labels ($f(o)$) of the objects. In order to evaluate the quality of a classifier h, each object is additionally classified using h. The results are given in the table (all three columns) below.

ID	$f(o)$	$h(o)$
O_1	A	A
O_2	B	A
O_3	A	C
O_4	C	C
O_5	C	B

ID	$f(o)$	$h(o)$
O_6	B	B
O_7	A	A
O_8	A	A
O_9	A	A
O_{10}	B	C

ID	$f(o)$	$h(o)$
O_{11}	B	A
O_{12}	C	A
O_{13}	C	C
O_{14}	C	C
O_{15}	B	B

(a) Rewrite the definitions for precision and recall given in the lecture by using TP, TN, FP, and FN.

Suggested solution :

— True positives for class i : $TP_i = \{o \mid f(o) = i \wedge h(o) = i\}$

— False positives for class i : $FP_i = \{o \mid f(o) \neq i \wedge h(o) = i\}$

— False negatives for class i : $FN_i = \{o \mid f(o) = i \wedge h(o) \neq i\}$

— (for the sake of completeness, although we do not use this in the computations below :) True negatives for class i : $TN_i = \{o \mid f(o) \neq i \wedge h(o) \neq i\}$

Precision :

$$\text{Precision}(h, i) = \frac{|\{o \in h_i \,|\, h(o) = f(o)\}|}{|h_i|}$$

or

$$\text{Precision}(h, i) = \frac{|TP_i|}{|TP_i| + |FP_i|}$$

Recall :

$$\text{Recall}(h, i) = \frac{|\{o \in f_i \,|\, h(o) = f(o)\}|}{|f_i|}$$

or

$$\text{Recall}(h, i) = \frac{|TP_i|}{|TP_i| + |FN_i|}$$

(b) Using the table (all three columns) above, compute precision and recall for each class.

Suggested solution :
Confusion matrix :

	A	B	C	$\|f_i\|$	$\|TP\|$	$\|FP\|$	$\|FN\|$
A	4	0	1	5	4	3	1
B	2	2	1	5	2	1	3
C	1	1	3	5	3	2	2
$\|h_i\|$	7	3	5				

Solution :
$\text{Precision}(h, A) = 4/7$
$\text{Precision}(h, B) = 2/3$
$\text{Precision}(h, C) = 3/5$
$\text{Recall}(h, A) = 4/5$
$\text{Recall}(h, B) = 2/5$
$\text{Recall}(h, C) = 3/5$

(c) To get a complete measure for the quality of the classification with respect to a single class, the F_1-measure (the harmonic mean of precision and recall) is commonly used. It is defined as follows :

$$F_1(h, i) = \frac{2 \cdot \text{Recall}(h, i) \cdot \text{Precision}(h, i)}{\text{Recall}(h, i) + \text{Precision}(h, i)}$$

Compute the F_1-measure for all classes.

Suggested solution :
$F_1(h, A) = 2/3$
$F_1(h, B) = 1/2$
$F_1(h, C) = 3/5$

(d) So far, the F_1-measure is only defined for classes and not yet useful to get an overview of the overall performance of the classifiers. To achieve such an overall assessment, one commonly takes the average over all classes using one of the following two approaches :

— Micro Average F_1-Measure : The values of TP, FP and FN are added up over all classes. Then precision, recall and F_1-measure are computed using these sums.

— Macro Average F_1-Measure : Precision and recall are computed for each class individually, afterwards the average precision and average recall are used to compute the F_1-measure.

Compute the Micro- and Macro-Average F_1-measures for the example above. What do you observe ?

Suggested solution :

Micro Average F_1 :

$|TP| = 4 + 2 + 3 = 9$
$|FP| = 3 + 1 + 2 = 6$
$|FN| = 1 + 3 + 2 = 6$
Precision : $9/15$, Recall : $9/15$, **Micro Average F_1 :** $9/15 = 3/5 = 0.6$.

Observation :

The recall for TP and FN over all classes is not a particularly sensible value, and is actually the same as precision (since FP in columns without the diagonal and FN in rows without diagonal add up to the same) :

$$\begin{aligned} \text{micro precisionrecall}(h) &= \frac{|\{o \in D \,|\, h(o) = f(o)\}|}{|D|} \\ &= \frac{\sum_i |TP_i|}{|D|} \\ &= \text{Accuracy}(h) \end{aligned}$$

Micro F_1 is in terms of information retrieval the same as accuracy in machine learning :

$$\begin{aligned} \text{micro}F_1 &= \frac{\frac{2 \cdot \sum_i |TP_i| \cdot \sum_i |TP_i|}{|D| \cdot |D|}}{2 \cdot \frac{\sum_i |TP_i|}{|D|}} \\ &= \frac{2 \cdot \sum_i |TP_i| \cdot \sum_i |TP_i| \cdot |D|}{2 \cdot \sum_i |TP_i| \cdot |D| \cdot |D|} \\ &= \frac{\sum_i |TP_i|}{|D|} \end{aligned}$$

Should be obvious : the harmonic mean of accuracy and accuracy is accuracy.

Macro Average F_1 :

average precision : $1/3(4/7 + 2/3 + 3/5) \approx 0.613$
average recall : $1/3(4/5 + 2/5 + 3/5) = 0.6$
Macro Average $F_1 \approx \frac{2 \cdot 0.6 \cdot 0.613}{0.6 + 0.613} = 0.606$.

The key difference between Micro- and Macro-Average is that in Macro Average all classes have the same weight. In Micro-Average, the classes are weighted by their size, so classes with a small size play a very small role. With Macro-Average, small classes are as important as large classes, which is often a desired property.

Example : an illness affects only 1% of the population. Always predicting "healthy" is correct in 99% of the cases, but the remaining 1% of error is maybe worse than errors in the healthy 99%.

Exercise 5-3 : Procedures for Evaluation of Classifiers

Given a data set D with objects from classes A and B ($D = A \cup B$) where the class assignments are *random* (not related to the attribute values). Furthermore, let the two classes have the same size $|A| = |B|$.

(a) What *true error rate* is to be expected for an *optimal* (for this data set) classifier ?

Suggested solution :

The best we can do on random data is to always predict the majority class. As both classes have the same size, we decide to predict always one of these classes.

For two classes of the same size and with randomly assigned labels, the expected error rate is thus 50%.

(b) What error rates are to be expected when training and evaluating an optimal classifier on the given dataset using a leave-one-out test ?

Suggested solution :

Since $|A| = |B|$ and each fold is exactly 1 object, the wrong class always becomes the majority. Thus the expected error rate becomes 100%, which is very pessimistic.

(c) Remember that in Bootstrap we produce the training and test data by sampling with replacement. An object is with a probability of

$$\left(1 - \frac{1}{n}\right)^n \approx 0.368$$

not part of the n training objects, i.e. only about 63.2% of the objects are used for training. (Compare this to 10-fold cross validation, where 90% of the data are used for training.)

This implies that the error estimation is pessimistic, as the training set has size n, but actually only contains $0.632 \cdot n$ *different* examples.

To make up for this, when evaluating bootstrap it is a common practice to also include the apparent classification error (error on the training data) during evaluation :

$$\text{error rate} = 0.632 \cdot \text{Error on test set} + 0.368 \cdot \text{Error on training set}$$

This will be repeated multiple times (with different samples) and averaged.

What error rates are to be expected when evaluating an optimal classifier on the given dataset using the 0.632 Bootstrap method ? Interpret these results.

Suggested solution :

Using the bootstrap evaluation setting, the optimum classifier for the minimum error on the given training set is "memorize".

The true error rate of the optimum classifier (predicting some class at random if the example is not memorized) is 50% (averaged over enough samples). On the training set however the "memorize" approach can achieve an accuracy of 100%. Using the formula above then yields

$$\text{error rate} = 0.632 \cdot 50\% + 0.368 \cdot 0\% = 31.6\%$$

which is a too optimistic estimation.

Exercise 5-4 : Events and Sample Spaces

(a) We have a system of several fuses. We can examine each single fuse to see whether it is defective. The sample space for this experiment can be abbreviated as $\Omega = \{N, D\}$, where N represents not defective, D represents defective.

If we examine three fuses in sequence and note the result of each examination, what is the sample space Ω ?

Suggested solution :
An outcome for the entire experiment is any sequence of Ns and Ds of length 3, so

$$\Omega = \{NNN, NND, NDN, NDD, DNN, DND, DDN, DDD\}$$

(b) As an experiment, we observe the number of pumps in use at a six-pump gas-station, so simple events are the numbers 0-6 (pumps in use). Given the events $A = \{0, 1, 2, 3, 4\}$, $B = \{3, 4, 5, 6\}$, and $C = \{1, 3, 5\}$, which simple events are contained in

i) $A \cup B$?

Suggested solution :
$A \cup B = \{0, 1, 2, 3, 4, 5, 6\} = \Omega$

ii) $A \cup C$?

Suggested solution :
$A \cup C = \{0, 1, 2, 3, 4, 5\}$

iii) $A \cap B$?

Suggested solution :
$A \cap B = \{3, 4\}$

iv) $A \cap C$?

Suggested solution :
$A \cap C = \{1, 3\}$

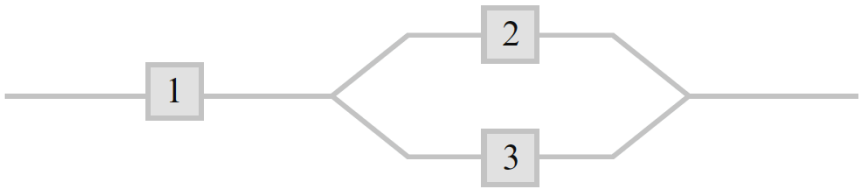
v) $\overline{A}$?

Suggested solution :
$\overline{A} = \{5, 6\}$

vi) $\overline{A \cup C}$?

Suggested solution :
$\overline{A \cup C} = \{6\}$

(c) Three components are connected to form a system as shown in this diagram :



Because the components in the 2-3 subsystem are connected in parallel, that subsystem will function if at least one of the two individual components functions. For the entire system to function, component 1 must function and so must the 2-3 subsystem. The experiment consists of determining the condition of each component (S (success) for a functioning component and F (failure) for a non-functioning component).

i) What outcomes are contained in the event D that exactly two out of the three components function ?

Suggested solution :
$D = \{SSF, SFS, FSS\}$

ii) What outcomes are contained in the event E that at least two of the components function ?

Suggested solution :
$E = \{SSF, SFS, FSS, SSS\}$

iii) What outcomes are contained in the event G that the system functions ?

Suggested solution :
$G = \{SSF, SFS, SSS\}$

iv) List the outcomes in $\overline{G}$, $D \cap G$, $D \cup G$, $E \cup G$, and $E \cap G$.

Suggested solution :
It is helpful to consider $\Omega = \{SSS, SSF, SFS, SFF, FSS, FSF, FFS, FFF\}$.
- $\overline{G} = \Omega \setminus C = \{SFF, FSS, FSF, FFS, FFF\}$
- $D \cap G = \{SSF, SFS\}$
- $D \cup G = \{SSF, SFS, FSS, SSS\}$
- $E \cup G = \{SSF, SFS, FSS, SSS\}$
- $E \cap G = \{SSF, SFS, SSS\}$

Exercise 5-5 : Tools : Over-fitting , Under-fitting

(a) Load python packages : `numpy`, `matplotlib.pyplot`, `datasets`, `metrics` from `sklearn`, `KNeighborsClassifier` from `sklearn.neighbors` and `train-test-split` from `sklearn.model-selection`.

Suggested solution :

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import datasets,metrics
import matplotlib.pyplot as plt
```

(b) Load `breast-cancer` dataset and split that into random train and test subsets and assign 40 percent of data to test dataset.

Suggested solution :

```
cancer = datasets.load_breast_cancer()
X = cancer.data
y = cancer.target
#Split dataset into random train and test subsets
X_train, X_test, y_train, y_test = train_test_split(X,
                                   y,test_size=0.4,random_state=3)
```

(c) Train the K Neighbors Classifier model, then predict and plot train and test accuracy for different number of neighbours from 1 to 100.

Suggested solution :

```
k_range=range(1,101, 1)
train_scores = []
test_scores = []
for k in k_range:
  knn = KNeighborsClassifier(n_neighbors=k)
  #Train the model using the training sets
  knn.fit(X_train, y_train)
  #Predict the response for test dataset
  train_yhat = knn.predict(X_train)
  train_acc = metrics.accuracy_score(y_train, train_yhat)
  y_pred = knn.predict(X_test)
  test_acc = metrics.accuracy_score(y_test,y_pred)
  test_scores.append(test_acc)
```

```
  train_scores.append(train_acc)
#Plot train and test accuracy
plt.plot(k_range, test_scores, label='Test accuracy')
plt.plot(k_range, train_scores, label='Train accuracy')
plt.legend()
plt.show()
```

(d) Now put the neighbours number equal to 3, and plot train and test accuracy again with 20 percent of data for test size and different train sizes from (0.1 to 0.8).

Suggested solution :

```
train_scores = []
test_scores = []
train_range = np.arange(.1,.9,.1)
for train_size in train_range:
  X_train, X_test, y_train, y_test = train_test_split(X, y,
  test_size= 0.2, train_size= train_siz, random_state=3)
  knn = KNeighborsClassifier(n_neighbors=3)
  #Train the model using the training sets
  knn.fit(X_train, y_train)
  #Predict the response for test dataset
  train_yhat = knn.predict(X_train)
  train_acc = metrics.accuracy_score(y_train, train_yhat)
  y_pred = knn.predict(X_test)
  test_acc = metrics.accuracy_score(y_test,y_pred)
  test_scores.append(test_acc)
  train_scores.append(train_acc)

plt.plot(train_range,test_scores, label='Test accuracy')
plt.plot(train_range,train_scores, label='Train accuracy')
plt.legend()
plt.show()
```

(e) Describe which "k" and "train size" for the model can cause over-fitting or under-fitting.