# Work 2. Case-based reasoning exercise

Marc Albert Garcia Gonzalo, Miquel Perelló Nieto

November 19, 2012

## 1 Introduction

In this exercise we have implemented a case-based reasoning system, specifically using the K nearest neighbor algorithm. We will use some variations of the algorithm with two different datasets, and we will analyze the results. In the first section we will evaluate the different KNN solutions using the Principal Component Analysis algorithm to create the new features. With this new features we will see the differences in accuracy with different values for the $k$ parameter for KNN and the parameter for Minkowsky distance. In the second we will see the results executing the KNN, selected KNN and weighted KNN with the original features. In that section the Minkowsky distance will be parametrised with the value two, performing the Euclidian distance. We will select in this section the $k$ parameter. In the last section we will test the Case Based Reasoning with the different KNN's with PFA, and we will extract the accuracy conclusions and computational times.

## 2 Project environment

This project has been developed using Matlab 7.12 for Linux.

These are the files of the project:

- *run* : Shell script to run PCA code from command line
- *run2* : Shell script to run PFA code from command line
- *run3* : Shell script to run CBR code from command line
- *main.m* : Main program, which executes the PCA section.
- *main2.m* : Main program, which executes the PFA section.
- *main_cbr.m* : Main program, which executes the CBR section.
- *kNN.m* : K nearest neighbor function
- *cbr.m* : CBR function with normal, selected, or weighed KNN with PFA.
- *pca.m* : Computes the PCA selecting returning the best principal components.
- *pfa.m* : Computes the PFA and returns a list with the best features and the weights for all features.
- *arffload.m* : Function to load data from ARFF to Matlab
- *arffloads.m* : Function to modify the information of the arff files to be used with the kNN algorithm
- *minkdist.m* : Function to calculate Minkowsky distances

To execute the different sections on a UNIX system, one of the next command needs to be called:

```
./run dataset_name
./run2 dataset_name
./run3 dataset_name
```

For example:

```
./run iris
```

# 3 KNN with Principal Component Analysis features

## 3.1 K nearest neigbor

In this section we describe the implementation and the results of the raw version of the K nearest neigbor algorithm.

The K nearest neigbor algorithm or KNN is a simple version of case-based reasoning algorithm, which classifies instances based on the class of the nearest known instances. The number of nearest instances is a parameter of the algorithm, and an odd number is usually used to avoid ties. Common values are 1, 3, 5 and 7. These are the values we use to compare the results of the algorithm.

The distance function used to compute the nearest instances is also a parameter of the algorithm. Euclidean distance is typically used, but in our exercice we experimented with different parameters of the Minkowsky distance. Parameters used are 1 (known as Manhattan distance), 2 (Euclidean distance), and 3.

Next are the results of the classification, showing the accuracy for each combination of the parameter of the Minkowsky distance (columns), and for the K parameter of KNN (rows):

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 0.9533 | 0.9600 | 0.9600 |
| 3 | 0.9600 | 0.9600 | 0.9600 |
| 5 | 0.9467 | 0.9667 | 0.9600 |
| 7 | 0.9467 | 0.9600 | 0.9733 |

Table 1: Results for the iris dataset

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 0.9078 | 0.9038 | 0.9009 |
| 3 | 0.9116 | 0.9102 | 0.9054 |
| 5 | 0.9113 | 0.9082 | 0.9027 |
| 7 | 0.9092 | 0.9077 | 0.9075 |

Table 2: Results for the SAT image dataset

The results are good, around 96% of accuracy in the Iris dataset, and around 90% on the SAT image dataset. But, from the obtained results we can see how there is not a significant improvement on the results, for any value of the different parameters.

## 3.2 Weighted K nearest neigbor

In this case, we have implemented a variation of the KNN algorithm. In the previous experiment we computed the distance or similarity between instances based on the original information on the datasets. In this experiment we will perform the PCA algorithm to extract new components for the data. The first of these components are the ones which maximize the variance in the data.

Then, we will weight each component, multiplying the values of each component by the eigenvalue associated to it. This eigenvalue is proportional to the variance of the component, so we will multiply by biggest values the components with higher variance, making them have even more variance.

This way, we will make components with less variance have more weight when computing the distance.

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 0.9267 | 0.9200 | 0.9200 |
| 3 | 0.9600 | 0.9467 | 0.9467 |
| 5 | 0.9533 | 0.9600 | 0.9600 |
| 7 | 0.9333 | 0.9400 | 0.9400 |

Table 3: Results for the iris dataset

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 0.8519 | 0.8398 | 0.8357 |
| 3 | 0.8586 | 0.8483 | 0.8446 |
| 5 | 0.8572 | 0.8477 | 0.8458 |
| 7 | 0.8601 | 0.8488 | 0.8465 |

Table 4: Results for the SAT image dataset

As we can see, the results for this method are worse than in the previous experiment. The computational cost of this method is also higher than the regular KNN, so, we do not see any advantage on the use of this method.

Again, we do not perceive any significant difference on the values used for the different parameters.

### 3.3 Selected K nearest neigbor

In this last experiment, we will use again PCA to extract components, but instead of setting different weights to them, we are going to discard the components with less variance. This method is called dimensionality reduction, and is useful, because the computational cost is lower, as the amount of data to process is smaller, and there is no significant loss of information, as the removed components have very small variance.

The criteria used to decide which components to use, and which to discard, it is been based on the eigenvalues of the matrix diagonalization. We have computed the mean of the eigenvalues, and we kept the components which eigenvalue was higher than this mean.

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 0.9467 | 0.9600 | 0.9600 |
| 3 | 0.9600 | 0.9600 | 0.9600 |
| 5 | 0.9667 | 0.9667 | 0.9600 |
| 7 | 0.9667 | 0.9600 | 0.9600 |

Table 5: Results for the iris dataset

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 0.8984 | 0.9038 | 0.9023 |
| 3 | 0.9057 | 0.9102 | 0.9093 |
| 5 | 0.9040 | 0.9082 | 0.9066 |
| 7 | 0.9040 | 0.9077 | 0.9065 |

Table 6: Results for the SAT image dataset

As we can see in the results, the accuracy of the classification is very similar to the one obtained with the first experiment using the regular KNN method. But in this case the method reduced the computational cost, and this is specially important when the datasets are large, like the SAT image dataset.

Once again, we do not see any significant difference on the results depending on the values used for the parameters.

# 4 Principal Feature Analysis applied to KNN

In this section we will compare the different methods to deal with original features, and we will compare the results on two datasets, Iris and Sat image. At this point we have fixed the $p$ parameter of *Minkowski distance* to the value two. This means that all the distances will be *Euclidean*. We will decide in this section wich $k$ value we will use in the CBR.

## 4.1 Selected K nearest neighbor

The methodology to select the best fatures was the same used in the article [1]. The approach is to use the results of PCA to select the new best components and project the original features in this new space. Once the projections are represented in this new space, we made a k-means to put each feature in a cluster and selected the features nearest to each new centroid.

Let $X$ be a zero mean n-dimensional random feature vector. Let $\Sigma$ be the covariance matrix of $X$. Let $A$ be a matrix whose *columns* are the orthonormal eigenvectors of the matrix $\Sigma$:

$\Sigma = A\Lambda A^T$

Where $\Lambda$ is a diagonal matrix whose diagonal elements are the eigenvalues of $\Sigma, \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. Let $A_q$ be the first $q$ *columns* of $A$ and let $V_1, V_2, \ldots V_n \in \Re^q$ be the *rows* of the matrix $A_q$.

**Step 1.** Compute the sample covariance matrix.

**Step 2.** Compute principal components and eigenvalues of covariance matrix.

**Step 3.** chose the subspace dimension $q$ and construct the new matrix $A_q$ from $A$. In this case we selected principal components with their eigenvalue larger than the average.

**Step 4.** Cluster the vectors $|V1|, |V2|, \ldots, |Vn|$ in $(p \geq q)$ clusters using K-Means algorithm. With the euclidian distance. The number of clusters $p$ that we selected is $q+1$ (In the article of [1] they recomend a value between $q+1 \leq p \leq q+5$).

**Step 5.** For each cluster find the corresponding vector $V_i$ wich is closest to the mean of the cluster. Choose the corresponding feature, $x_i$, as a principal feature.

## 4.2 Weighted K nearest neighbor

In the weighted approach instead of selecting the most significant features, we will give weights to all the features with this approach:

**Step 1.** Compute the sample covariance matrix.

**Step 2.** Compute principal components and eigenvalues of covariance matrix.

**Step 3.** In this point we can see that as bigger is the eigenvalue, bigger is the relevance of the principal component associated. In the rows of $A_q$ (eigenvectors) we have the $V_1, V_2, \ldots, V_n$ vectors that corresponds to the projections of the $X_i$ original features in the new space. And each eigenvalue $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ correspond to one new principal component.

**Step 4.** If we weight each column of the eigenvector matrix by the corresponding eigenvalue, we will give most relevance to the most important projections.

**Step 5.** Once all this values are weighted we can compute the sum of each weighted projection for each feature:

$W_i = \Sigma_{j=0..n} |V_{ij} * \lambda_j|$

**Step 6.** In this step we will normalize all this new weights to sum 1.

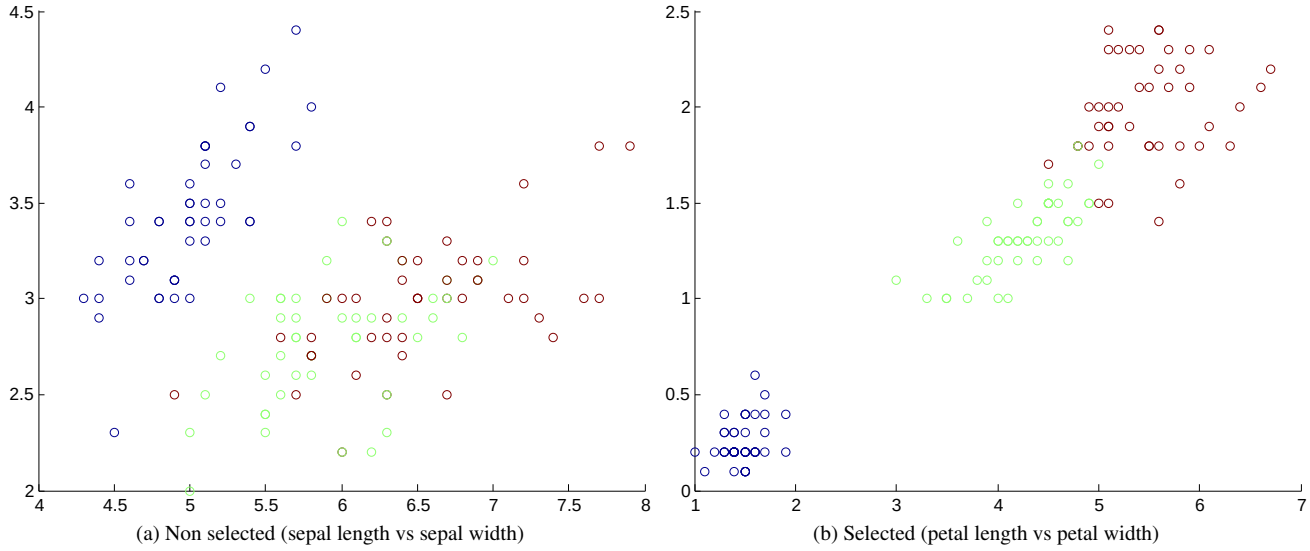
## 4.3 Evaluation

### 4.3.1 Iris dataset

With the Iris dataset the results of the weight compute in one of the fold training data can be seen in the table 7. And the selection have been the petal length and petal width (see picture 1). The weights and the selection are not related because in the selection we want only one element for each cluster in the new space, but in the weights we weight the features according to the projection to the best principal components.

| sepallength | sepalwidth | petallength | petalwidth |
|---|---|---|---|
| 0.2201 | 0 | **0.5862** | **0.1938** |

Table 7: Weights for each feature and selected features in bold.




(a) Non selected (sepal length vs sepal width)

(b) Selected (petal length vs petal width)

Figure 1: Selected and non selected features with Principal Feature Analysis.

This is the result of computing the mean for each 10-fold cross validation, with the accuracy for each value of $K$ in the KNN algorithm and the different kNN's (table 8). With this results it seems that the selected KNN presents worse results in average but the difference does not seem to be relevant. In the other side the average results for the value of $k$ parameter points at 7 value like the best, but another time it does not seem significant. And referencing to the execution time it does not seems to be a significant difference with this dataset (table 9).

| **K** | **KNN** | **SelectedKNN** | **WeightedKNN** | **mean** |
|---|---|---|---|---|
| 1 | 0.9600 | 0.9533 | 0.9533 | 0.955 |
| 3 | 0.9600 | 0.9333 | 0.9667 | 0.953 |
| 5 | 0.9667 | 0.9400 | 0.9667 | 0.957 |
| 7 | 0.9600 | 0.9667 | 0.9600 | **0.962** |
| **mean** | **0.9617** | 0.9483 | **0.9617** | |

Table 8: Accuracy results for the iris dataset.

| **K** | **KNN** | **SelectedKNN** | **WeightedKNN** |
|---|---|---|---|
| 1 | 0.002 | 0.007 | 0.006 |
| 3 | 0.008 | 0.005 | 0.006 |
| 5 | 0.005 | 0.004 | 0.008 |
| 7 | 0.007 | 0.005 | 0.003 |
| **mean** | 0.0055 | 0.0052 | 0.0057 |

Table 9: Mean times for 10-fold with different K values (in seconds).

### 4.3.2 Sat image dataset

With the SAT image dataset there are thirty-six features and six classes, in this case getting the eigenvalues we can see that this dataset have almost all the information in three principal components (this are largest than the average; see table 10). For that reason we have four selected features that try to explain all the information contained in the other features. These four selected can be seen in the table of weights (table 11) and are the first, twenty-eight, thirty and thirty-one. We can see in the figure 2 two of the selected features with the classification for the training data.

| [0,5] | [6,10] | [11,15] | [16,20] | [21,25] | [26,30] | [31,35] | [36] |
|---|---|---|---|---|---|---|---|
| **0.7065** | 0.0205 | 0.0029 | 0.0015 | 0.0010 | 0.0006 | 0.0005 | 0.0003 |
| **0.3987** | 0.0115 | 0.0023 | 0.0015 | 0.0009 | 0.0006 | 0.0004 | |
| **0.0689** | 0.0066 | 0.0021 | 0.0014 | 0.0008 | 0.0005 | 0.0004 | |
| 0.0326 | 0.0046 | 0.0021 | 0.0011 | 0.0007 | 0.0005 | 0.0004 | |
| 0.0234 | 0.0044 | 0.0016 | 0.0010 | 0.0007 | 0.0005 | 0.0003 | |

Table 10: Eigenvalues of the Principal Component Analysis for SAT image dataset.

| [0,5] | [6,10] | [11,15] | [16,20] | [21,25] | [26,30] | [31,35] | [36] |
|---|---|---|---|---|---|---|---|
| **0.0437** | 0.0359 | 0.0258 | 0.0016 | 0.0416 | 0.0407 | **0.0265** | 0.0034 |
| 0.0376 | 0.0267 | 0.0021 | 0.0410 | 0.0416 | 0.0289 | 0.0025 | |
| 0.0328 | 0.0014 | 0.0425 | 0.0390 | 0.0257 | **0.0035** | 0.0425 | |
| 0.0048 | 0.0428 | 0.0367 | 0.0240 | 0.0016 | 0.0413 | 0.0425 | |
| 0.0420 | 0.0414 | 0.0259 | 0 | 0.0426 | **0.0410** | 0.0266 | |

Table 11: Different weights for all original features for SAT image.

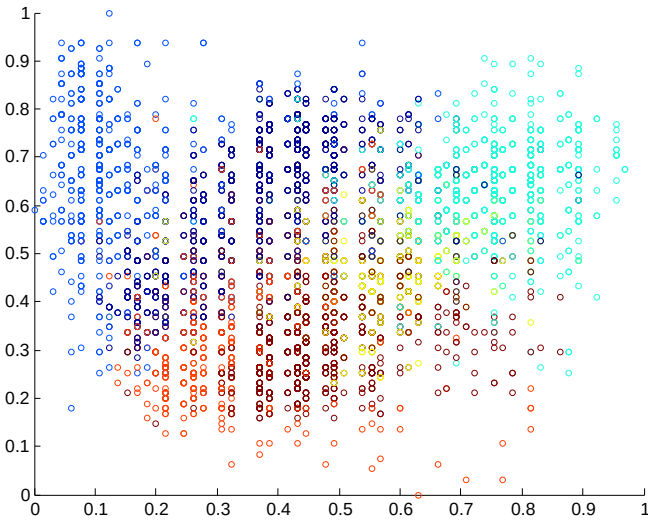



Figure 2: Selected features 1 and 31.

The results of accuracy in this case are very significant for the selected version (see table 12). With only four of the thirty-six features it has 10% less accuracy. May be would be interesting to select the number of features according to the original number of features. But in the other side, the computation time is five times lower (see table 13) and it will reduce the number of sensors in some applications. Then, in some cases could be interesting to think in this kind of improvements.

| **K** | **KNN** | **SelectedKNN** | **WeightedKNN** | **mean** |
|---|---|---|---|---|
| 1 | 0.9038 | 0.7776 | 0.8934 | 0.858 |
| 3 | 0.9102 | 0.8078 | 0.9049 | 0.874 |
| 5 | 0.9082 | 0.8106 | 0.9021 | 0.873 |
| 7 | 0.9077 | 0.8204 | 0.9016 | **0.876** |
| **mean** | **0.9075** | 0.8041 | 0.9005 | |

Table 12: Accuracy results for the satimage dataset.

| K | KNN | SelectedKNN | WeightedKNN |
|---|---|---|---|
| 1 | 5.843 | 1.441 | 5.879 |
| 3 | 5.534 | 1.267 | 5.355 |
| 5 | 5.622 | 1.265 | 5.291 |
| 7 | 5.623 | 1.303 | 5.507 |
| **mean** | 5.6555 | **1.319** | 5.508 |

Table 13: MAverage times for 10-fold with different K values (in seconds).

# 5 Case Based Reasoning

In this section we will see an approach to resolve the classification problem with Case Based Reasoning using algorithms and parameters discussed in sections before. And we will analyze the different KNN algorithms with selection and weighted features to extract conclusions. In this point we have fixed the $p$ parameter of *Minkowski distance* to the value two. This means that all the distances will be *Euclidean*. About the $k$ parameter for KNN selection, we will use the value seven. We do not see a real difference betwen values of $k$, but on average the value seven was slightly better.

In Case Based Reasoning there are four important points that needs to be defined, there are a lot of different improvements to do this steps and we decided to use one simple configuration to test, but it is possible to modify all the steps by more complex aproaches. This is the configuration implemented in our work:

1. **Retrive :** We will use the training dataset like memory cases.
2. **Reuse :** In this step with a new case we will see wich are the most similar cases in our memory cases, and we will predict the corresponding class for this new element.
3. **Revise :** Once we have decided wich class corresponds to this new instance, we can prove if this classification is correct or not.
4. **Retain :** With this new case correctly classified we want to classify all the next cases near this point with the same solution. To do this, we will add this new instance to the memory cases.

## 5.1 Iris dataset

These are the average accuracy for all ten-fold cross validation with CBR and the training data used as memory cases (see table 14). In this case the number of individuals in test data is about fourteen. For that reason the retain process occurs about half times in average (table 15). This is reflected in the accuracy values that do not seem to improve.

| | **KNN** | **SelectedKNN** | **WeightedKNN** |
|---|---|---|---|
| CBR | 0.960 | 0.966 | 0.960 |
| Without CBR | 0.960 | 0.966 | 0.960 |

Table 14: Results for the iris dataset with Euclidian distance and parameter $k = 7$.

| **KNN** | **SelectedKNN** | **WeightedKNN** |
|---|---|---|
| 0.6000 | 0.5000 | 0.6000 |

Table 15: Average of cases addeds to CBR.

## 5.2 Sat image dataset

In this case with the SAT image dataset the number of individuals is six thousand. It makes much probable to see the effect of increasing the knowledge of case based reasoning. Anyway the improvement performed with this approach is not evident, it is because the retained cases have not been used to classify the posterior cases.

| | **KNN** | **SelectedKNN** | **WeightedKNN** |
|---|---|---|---|
| CBR | 0.9077 | 0.8272 | 0.9016 |
| Without CBR | 0.9077 | 0.8204 | 0.9016 |

Table 16: Results for the SAT image dataset.

| **KNN** | **SelectedKNN** | **WeightedKNN** |
|---|---|---|
| 59.4 | 111.2 | 63.3 |

Table 17: Average of cases addeds to CBR.

# References

[1] Lu, Y., Cohen, I., Zhou, X., & Tian, Q. (2007). Feature selection using principal feature analysis. ... of the 15th international conference on .... Retrieved from http://dl.acm.org/citation.cfm?id=1291297

[2] Hall, M. A. (n.d.). Feature Selection for Discrete and Numeric Class Machine Learning (pp. 1–16). Hamilton.