MACHINE LEARNING FROM DATA
Fall 2018

Report: Lab Session 3 – K-Nearest Neighbors and Parzen windows

Names: Santagiustina Francesco, Simonetto Adriano
Group:

1. Instructions

- Answer the questions
- Save the report and upload the file

2. Questions

Q1. Complete a table with training and test errors obtained for kNN and discuss the results. What is the value of k? Analyze the confusion matrices and identify the two most challenging classes.

P_e	Training set	Test set
k_{NN}	0.0311	0.0638

According to the notes of the authors an error rate below 2.5% percent is excellent. As we can see kNN is performing pretty close to that value, showing a really good accuracy on both training and test data. This confirms that kNN keeps a good performance even in the presence of an amount of data that is low with respect to the amount of features we are considering (7291 training samples against 256 features).

k = 10

From the confusion matrices we can see that the number 4 is the one getting the biggest amount of misclassifications, and is mostly confused with number 9. The second worst appears to be 5, that gets misclassifies in a more even manner as 0, 3 and 9.

Q2. Run again the script, using PCA to reduce the dimensionality of the feature space, selecting d'=64 features. Observe the eigenvectors and the images reconstructed using only the first d' eigenvectors (those with the highest eigenvalues). Discuss. Complete a table with training and test errors. Discuss the results and compare with the previous case (no PCA).

The eigenimages clearly present the most common shapes appearing in the various numbers. As an example the first one shows a pattern closely related to the shape of numbers as 0, 8, 9 etc.
The reconstruction shows images that are more blurred than before, but that still clearly show the numbers.

P_e	Training set	Test set
PCA	0.0269	0.0648

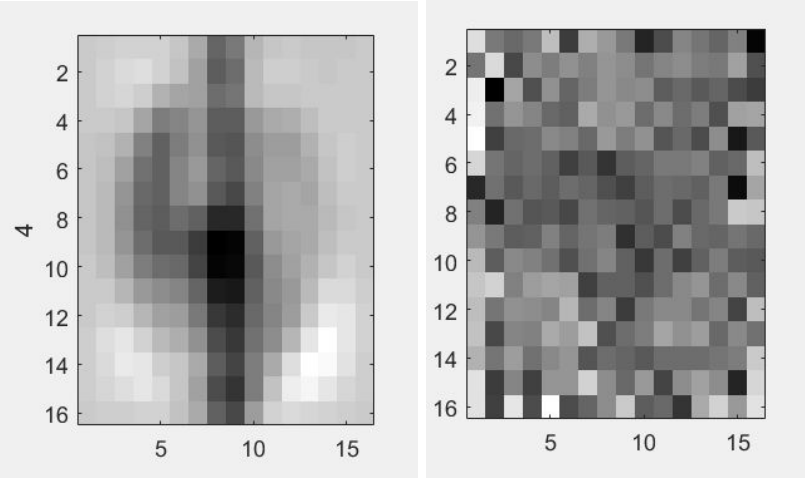
From the table we can see that the performance does not change much even with a reduced amount of features; this means that the information is mostly contained in the first 64 features.

Q3. Repeat the previous analysis using PCA with d'=9 features, and MDA with d'=9 features. Discuss which method is the best for image reconstruction and which one is preferable for classification.

P_e	Training set	Test set
PCA	0.0695	0.1196
MDA	0.0417	0.0992

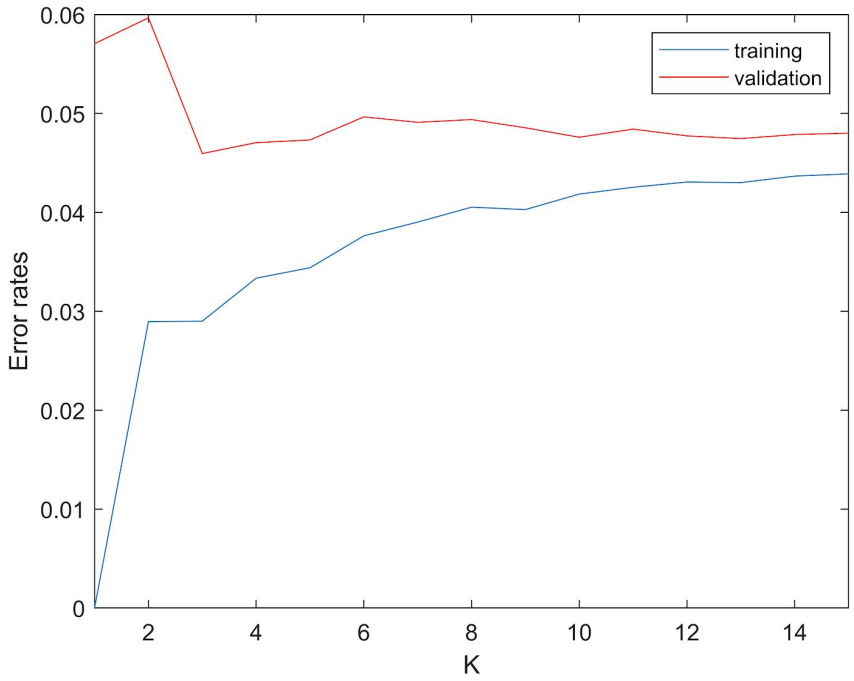
From the table above it is quite easy to notice how MDA outperforms PCA on both the training and test set and should be therefore preferred in a classification framework. Instead, if the aim is image reconstruction, PCA is a much better choice as, using MDA, the original shape gets completely lost; underneath we show as an example, the reconstruction of number 4 for both methods.

PCA MDA



Q4. Edit the script and modify the code to find the optimal value of k by K-fold cross validation on the training set. Use K=10 folds. Plot the average train and validation errors (average over the folds) as a function of k. Use the optimal value of k to compute the error on the test set.

As we can see from the plot below, the value of k giving us the lowest validation error is k=3. For that value of k we get an error on the test set equal to 0.1011, which is slightly better than the result obtained before for k = 10.



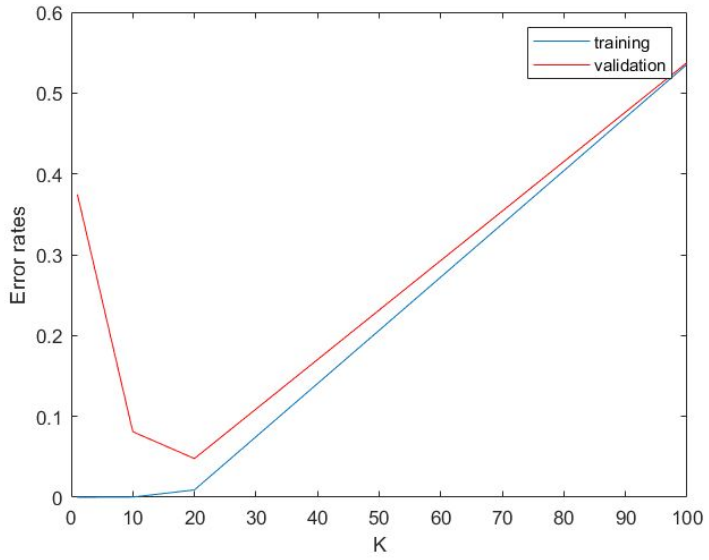
Q5. Copy the new code

```
% MC 2017
close all;
clear;
clc;
N_classes=10;
N_feat = 256;
N_dim=16;
%% Read training BD
X_train=[];              % matrix of Nx256 containing all vectors
Labels_train=[];         % labels (samples are initially ordered from
class 0 to class 9)

for k=0:N_classes-1
    nombre=sprintf('train%d.txt',k);
    [data] = textread(nombre,'','delimiter',',');
    %data=round(data);  %optional

    X_train=[X_train;data];
    N_size=size(data);
    Labels_train=[Labels_train;k*ones(N_size(1),1)];
end
clear nombre data k N_size
%% Read test BD
```

```
nombre=sprintf('zip.test');
[data] = textread(nombre,'','delimiter',' ');
Labels_test =data(:,1);
X_test=data(:,2:size(data,2));
clear nombre data
%% Perform MDA feature selection with d' = 9
N_feat = 9;
COEFF= mda_ml(X_train,Labels_train+1,N_classes);
      W=COEFF(:,1:N_feat);
      X_train=X_train*W;
      X_test=X_test*W;
%% Create a knn classifier:
pace = 1;
start = 1;
ending = 10;
res = start:pace:ending;  %different values of k we are going to test
n_kfold = 10;
X_train_copy = X_train;
Labels_train_copy = Labels_train;
X_test_copy = X_test;
Labels_test_copy = Labels_test;
train_err = zeros(length(res),1);
val_err = zeros(length(res),1);
aux = 1; %counter needed for train_err and val_err
for iter1 = res
    cp = cvpartition(size(X_train_copy,1),'Kfold',n_kfold);
    avg_train_err = 0;
    avg_val_err = 0;
    for iter2=1:n_kfold
        %X_train and Labels_train now contain only the values decided by
        %cvpartition. X_test contains the validation ones
        X_train = X_train_copy(find(training(cp,iter2)),:);
        Labels_train = Labels_train_copy(find(training(cp,iter2)));
        X_test = X_train_copy(find(test(cp,iter2)),:);
        Labels_test = Labels_train_copy(find(test(cp,iter2)));

        knnclass = fitcknn(X_train,Labels_train,'NumNeighbors',iter1);
        knn_out = predict(knnclass,X_train);
        knn_Pe_train=sum(Labels_train ~= knn_out)/length(Labels_train);
        knn_out = predict(knnclass,X_test);
        knn_Pe_test=sum(Labels_test ~= knn_out)/length(Labels_test);

        avg_train_err = avg_train_err + knn_Pe_train;
        avg_val_err = avg_val_err + knn_Pe_test;
    end
    train_err(aux) = avg_train_err/n_kfold;
    val_err(aux) = avg_val_err/n_kfold;
    aux = aux + 1;
end
%find and test the best value of k
[min_err, best_k] = min(val_err);
```

```
best_k = res(best_k);
knnclass = fitcknn(X_train_copy,Labels_train_copy,'NumNeighbors',best_k);
knn_out = predict(knnclass,X_train_copy);
knn_Pe_train=sum(Labels_train_copy ~= knn_out)/length(Labels_train_copy);
knn_out = predict(knnclass,X_test_copy);
knn_Pe_test=sum(Labels_test_copy ~= knn_out)/length(Labels_test_copy);
%plot
figure
grid on
plot(res,train_err)
hold on
plot(res,val_err,'r')

xlabel('K')
ylabel('Error rates')
legend('training','validation')
```

Q6. Use K-fold cross validation (K=10) to select the best parameter h (Parzen windows), where possible values of h are 1, 10, 20 and 100.
Plot the average train and validation errors (average over the folds) as a function of k. Use the optimal value of h to compute the error on the test set.



The best value for h is 20 : we obtain over the full dataset a training error of 0.0081 and a test error of 0.1046 which is very similar to the one obtained with the 3-nearest neighbours algorithm.

Q7 Copy the new code

```
% MC 2017
```

Lab 3: kNN and Parzen windows

Machine Learning from Data

```
close all;
clear;
clc;
N_classes=10;
N_feat = 256;
N_dim=16;
%% Read training BD
X_train=[];              % matrix of Nx256 containing all vectors
Labels_train=[];         %  labels  (samples  are  initially  ordered  from
class 0 to class 9)

for k=0:N_classes-1
      nombre=sprintf('train%d.txt',k);
      [data] = textread(nombre,'','delimiter',',');
      %data=round(data);  %optional

      X_train=[X_train;data];
      N_size=size(data);
    Labels_train=[Labels_train;k*ones(N_size(1),1)];
end
clear nombre data k N_size
%% Read test BD
nombre=sprintf('zip.test');
[data] = textread(nombre,'','delimiter',' ');
Labels_test =data(:,1);
X_test=data(:,2:size(data,2));
clear nombre data
%% Perform MDA feature selection with d' = 9
N_feat = 9;
COEFF= mda_ml(X_train,Labels_train+1,N_classes);
      W=COEFF(:,1:N_feat);
      X_train=X_train*W;
      X_test=X_test*W;
%% Create a fit the parameter h of a parzen classifier:

res = [1, 10, 20, 100];  %different values of k we are going to test
n_kfold = 10;
X_train_copy = X_train;
Labels_train_copy = Labels_train;
X_test_copy = X_test;
Labels_test_copy = Labels_test;
train_err = zeros(length(res),1);
val_err = zeros(length(res),1);
aux = 1; %counter needed for train_err and val_err
for iter1 = res
```

```
      cp  =  cvpartition(size(X_train_copy,1),'Kfold',n_kfold);  %%  is  it
optimal to compute a new partition for each k ?
      avg_train_err = 0;
      avg_val_err = 0;
      for iter2=1:n_kfold
      %X_train and Labels_train now contain only the values decided by
      %cvpartition. X_test contains the validation ones
      X_train = X_train_copy(find(training(cp,iter2)),:);
      Labels_train = Labels_train_copy(find(training(cp,iter2)));
      X_test = X_train_copy(find(test(cp,iter2)),:);
      Labels_test = Labels_train_copy(find(test(cp,iter2)));

      parzen_out                                                          =
predict_parzen(X_train,Labels_train,N_classes,iter1,X_train);
                                    parzen_Pe_train=sum(Labels_train    ~=
parzen_out)/length(Labels_train);
      parzen_out                                                          =
predict_parzen(X_train,Labels_train,N_classes,iter1,X_test);
      parzen_Pe_test=sum(Labels_test ~= parzen_out)/length(Labels_test);

      avg_train_err = avg_train_err + parzen_Pe_train;
      avg_val_err = avg_val_err + parzen_Pe_test;
      end
      train_err(aux) = avg_train_err/n_kfold;
      val_err(aux) = avg_val_err/n_kfold;
      aux = aux + 1;
end
%find and test the best value of h
[min_err, best_h] = min(val_err);
best_h = res(best_h);

parzen_out                                                              =
predict_parzen(X_train_copy,Labels_train_copy,N_classes,best_h,X_train_co
py);
parzen_Pe_train=sum(Labels_train_copy                                   ~=
knn_out)/length(Labels_train_copy);

parzen_out                                                              =
predict_parzen(X_train_copy,Labels_train_copy,N_classes,best_h,X_test_cop
y);
parzen_Pe_test=sum(Labels_test_copy ~= knn_out)/length(Labels_test_copy);

%plot
figure
grid on
```

```
plot(res,train_err)
hold on
plot(res,val_err,'r')

xlabel('K')
ylabel('Error rates')
legend('training','validation')
```

Q8. Use the Microarray dataset. Complete a table with the training and test errors for different values of k=1, 2, 3, 4. Discuss the results.

k	1	2	3	4
P_error knn train	0	0.208333	0.24	0.25
P_error knn test	0.4	0.28	0.44	0.44

On the training set the error probability obviously increase when we increase k as the model is less fitted to the training data (as we are considering more neighbours, thus enlarging decision regions). Concerning the test set, the above table shows us that the best results are obtained for a value of k=2, we can consider that for k=1 the model is overfitted while for k=3 and k=4 the model is underfitted. Even for the best k the results are quite poor as we wrongly classify almost 30% of the test samples, but we need to keep in mind that the classification was very challenging given the low number of samples and the high dimensionality of data.