Machine Learning(Assignment1)

Farhad M. Kazemi MUN ID:201576196

1-

Application description	**Approach description**	**Reference**	**Type of learning**
1-medical image segmentation. Image segmentation, a process to divide a given image into meaningful regions with homogeneous properties, is an essential step in image analysis and recognition.	Image segmentation is a process for dividing a given image into meaningful regions with homogeneous properties. A new two step approach is proposed for medical image segmentation using a fuzzy Hopfield neural network based on both global and local gray-level information. The membership function simulated with neuron outputs is determined using a fuzzy set, and the synaptic connection weights between the neurons are predetermined and fixed to improve the efficiency of the neural network. The proposed method needs initial cluster centers. The initial centers can be obtained from the global information about the distribution of the intensities in the image, or from prior knowledge of the intensity of the region of interest. It is shown by experiments that the proposed fuzzy Hopfield neural network approach is better than most previous approaches. We also show that the global information can be used by applying the hard c-means to estimate the initial cluster centers.	Chang, Chwen-Liang, and Yu-Tai Ching. "Fuzzy Hopfield neural network with fixed weight for medical image segmentation." *Optical Engineering* 41.2 (2002): 351-358.	Unsupervised Learning
2-Individual recognition by Walking	In this paper, Jinguang proposed a new spatio-temporal gait representation, called Gait Energy Image (GEI), to characterize human walking properties for individual recognition by gait. To address the problem of the lack of training templates, He also proposed a novel approach for human recognition by combining statistical gait features from real and synthetic templates. Hee directly computed the real templates from training silhouette sequences, while He generated the synthetic templates from training sequences by simulating silhouette distortion. He used a statistical approach for learning	Han, Jinguang, and Bir Bhanu. "Individual recognition using gait energy image." *Pattern Analysis and Machine Intelligence, IEEE Transactions on*28.2 (2006): 316-322.	Supervised Learning

	effective features from real and synthetic templates. He compared the proposed GEI-based gait recognition approach with other gait recognition approaches on USF HumanID Database. Experimental results show that the proposed GEI is an effective and efficient gait representation for individual recognition, and the proposed approach achieves highly competitive performance.		
3-Human ear recognition	Human ear is a new class of relatively stable biometrics that has drawn researchers' attention recently. In this paper, Chen proposed a complete human recognition system using 3D ear biometrics. The system consists of 3D ear detection, 3D ear identification, and 3D ear verification. For ear detection, He proposed a new approach which uses a single reference 3D ear shape model and locates the ear helix and the antihelix parts in registered 2D color and 3D range images. For ear identification and verification using range images, two new representations are proposed. These include the ear helix/antihelix representation obtained from the detection algorithm and the local surface patch (LSP) representation computed at feature points. A local surface descriptor is characterized by a centroid, a local surface type, and a 2D histogram. The 2D histogram shows the frequency of occurrence of shape index values versus the angles between the normal of reference feature point and that of its neighbors. Both shape representations are used to estimate the initial rigid transformation between a gallery-probe pair. This transformation is applied to selected locations of ears in the gallery set and a modified iterative closest point (ICP) algorithm is used to iteratively refine the transformation to bring the gallery ear and probe ear into the best alignment in the sense of the least root mean square error. The experimental results on the UCR data set of 155 subjects with 902 images under pose variations and the	Chen, Hui, and Bir Bhanu. "Human ear recognition in 3D." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 29.4 (2007): 718-737.	Supervised Learning

	University of Notre Dame data set of 302 subjects with time-lapse gallery-probe pairs are presented to compare and demonstrate the effectiveness of the proposed algorithms and the system		
4-nonlinear discrete-time MIMO systems	Based on the neural network (NN) approximator, an online reinforcement learning algorithm is proposed for a class of affine multiple input and multiple output (MIMO) nonlinear discrete-time systems with unknown functions and disturbances. In the design procedure, two networks are provided where one is an action network to generate an optimal control signal and the other is a critic network to approximate the cost function. An optimal control signal and adaptation laws can be generated based on two NNs. In the previous approaches, the weights of critic and action networks are updated based on the gradient descent rule and the estimations of optimal weight vectors are directly adjusted in the design. Consequently, compared with the existing results, the main contributions of this paper are: (1) only two parameters are needed to be adjusted, and thus the number of the adaptation laws is smaller than the previous results and (2) the updating parameters do not depend on the number of the subsystems for MIMO systems and the tuning rules are replaced by adjusting the norms on optimal weight vectors in both action and critic networks. It is proven that the tracking errors, the adaptation laws, and the control inputs are uniformly bounded using Lyapunov analysis method. The simulation examples are employed to illustrate the effectiveness of the proposed algorithm.	Liu, Yan-Jun, et al. "Reinforcement learning design-based adaptive tracking control with less learning parameters for nonlinear discrete-time MIMO systems." *Neural Networks and Learning Systems, IEEE Transactions on*26.1 (2015): 165-176.	Reinforcement Learning
5-Iris recognition	A new set of features for personal verification and identification based on iris image is proposed in this paper. The method consists of three major components: image pre-processing, feature extraction and classification. During image pre-processing, the iris segmentation is carried out using	Umer, Saiyed, Bibhas Chandra Dhara, and Bhabatosh Chanda. "Iris recognition using multiscale morphologic features." *Pattern*	Supervised Learning

	Restricted Circular Hough transformation (RCHT). Then only two disjoint quarters of the segmented iris pattern are normalized which is used to extract features for classification purposes. Here, method for feature extraction from iris pattern is based on multiscale morphologic operator. In this approach, the iris features are represented by the sum of dissimilarity residues obtained by applying morphologic top-hat transform. For classification purposes the multi-class problems is transformed to two-class problem using dichotomy method. The performance of the proposed system is tested on four benchmark iris databases UPOL, MMU1, IITD, and UBIRIS and is compared with well known existing methods.	*Recognition Letters* 65 (2015): 67-74.	
6-dialogue system for human–robot interactions with socially-inspired rewards	This paper investigates some conditions under which polarized user appraisals gathered throughout the course of a vocal interaction between a machine and a human can be integrated in a reinforcement learning-based dialogue manager. More specifically, we discuss how this information can be cast into socially-inspired rewards for speeding up the policy optimisation for both efficient task completion and user adaptation in an online learning setting. For this purpose a potential-based reward shaping method is combined with a sample efficient reinforcement learning algorithm to offer a principled framework to cope with these potentially noisy interim rewards. The proposed scheme will greatly facilitate the system's development by allowing the designer to teach his system through explicit positive/negative feedbacks given as hints about task progress, in the early stage of training. At a later stage, the approach will be used as a way to ease the adaptation of the dialogue policy to specific user profiles. Experiments carried out using a state-of-the-art goal-oriented dialogue management framework, the Hidden Information State (HIS), support our	Ferreira, Emmanuel, and Fabrice Lefevre. "Reinforcement-learning based dialogue system for human–robot interactions with socially-inspired rewards."*Computer Speech & Language* 34.1 (2015): 256-274.	Reinforcement Learning

	claims in two configurations: firstly, with a user simulator in the tourist information domain (and thus simulated appraisals), and secondly, in the context of man–robot dialogue with real user trials.		

2-a

The definition of the distance function.

Actually I used two types of distances. Eventually the Experimental results of the first kind of distance was satisfying. **You can see the results of the first one in diagram in section d.**

1)
dists = sum(abs(trainx - ones(n1,1)*test(i,:)),2); absolute value

2)
dist= sqrt(sum((trainx - ones(n1,1)*test(i,:)).^2,2)); Euclidean distance

2-b

My pseudocode:

```
kNN (dataset, sample){

  1. Go through each item in my dataset, and calculate the "distance"

  from that data item to my specific sample.

  2. Classify the sample as the majority class between K samples in

  the dataset having minimum distance to the sample.

}
```

This pseduocode has been illustrated in the following figure.

```
function result = knnclassifier(trainx, trainy,test, k)
class= unique(trainy);
N= size(test,1);
n1=length(trainy);
if ( n1 < k)
   error('You specified more neighbors than existed points.')
end

%// Use distance
for i=1:N
newpoint=test(i,:);
dists = sum(abs(trainx - ones(n1,1)*test(i,:)),2);
%dists= sqrt(sum((trainx - ones(n1,1)*test(i,:)).^2,2));
Euclidean distance

[d,ind] = sort(dists);
ind_closest = ind(1:k);
x_closest = trainx(ind_closest,:);
x_closest_class=trainy(ind_closest,:);
x_closesthist=hist(x_closest_class,class);
[c, best]= max(x_closesthist);

result(i,1)= class(best);
end
```

2-c

Crossvalidation

```
%% Insert Data

load GlassA1data;

X = GlassA1data;
dataRowNumber = size(GlassA1data,1);

crossValidationFolds = 5;
numberOfRowsPerFold = dataRowNumber / crossValidationFolds;

crossValidationTrainData = [];
```

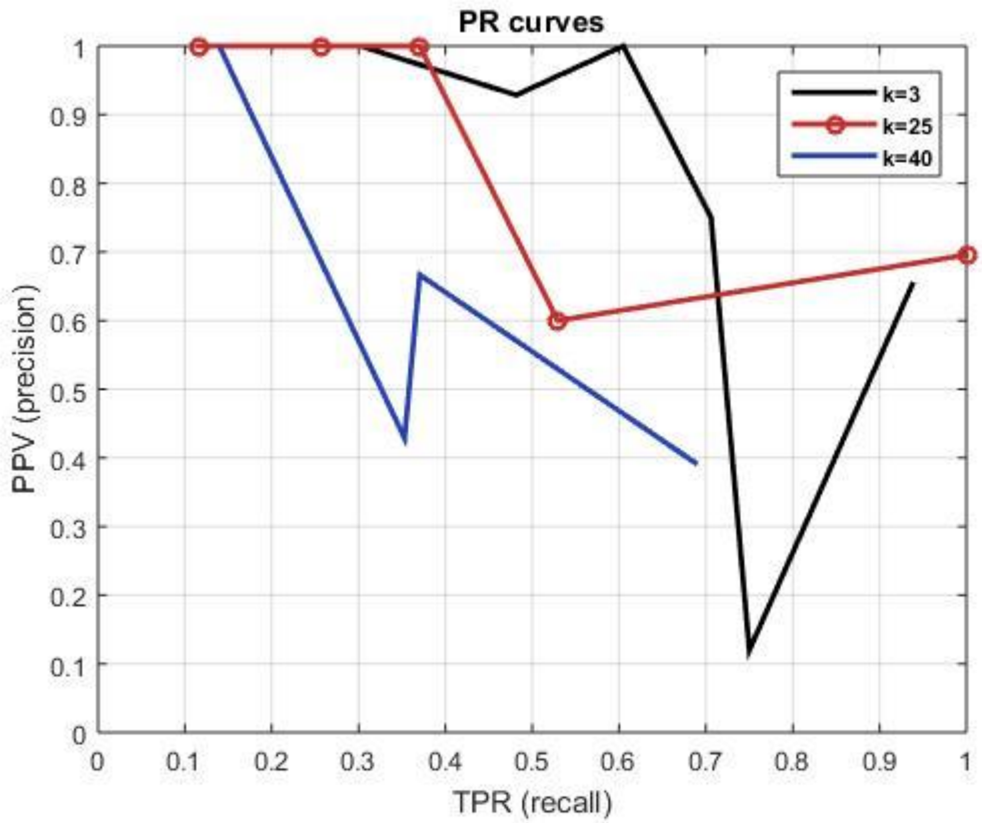
```
crossValidationTestData = [];
for startOfRow = 1:numberOfRowsPerFold:dataRowNumber
    testRows = startOfRow:startOfRow+numberOfRowsPerFold-1;
    if (startOfRow == 1)
        trainRows = [max(testRows)+1:dataRowNumber];
        else
        trainRows = [1:startOfRow-1
max(testRows)+1:dataRowNumber];
    end

crossValidationTrainData = [crossValidationTrainData ;
X(trainRows ,:)];
crossValidationTestData = [crossValidationTestData ;X(testRows
,:)];
end
```

2-d

In this question, at first I encountered with a problem that includes multiclass. As you know Precision -Recall curve is based on binary classification. So I implemented this subject for multiclass problem. **For details you can refer to Source Code Section.**

Precision-recall curves for three different values of *k*.

k1=3;
k2=25;
k3=40; As you can see, if we consider k=40, the result is better.



You'll also notice that precision and recall are inversely related. As such, if precision increases, then recall decreases. Similarly, if precision decreases, then recall will increase.

- The first part makes sense because if you don't retrieve that many samples in the beginning, you have a greater chance of not including irrelevant samples in your results but at the same time, the amount of relevant samples is rather small. This is why recall would decrease when precision would increase

- The second part also makes sense because as you keep trying to retrieve more samples in your database, you'll inevitably be able to retrieve all of the relevant ones, but you'll most likely start to include more irrelevant samples, which would thus drive your precision down.

Source Code (knn) main function

```
clc
clear
close all
format shortG
%% Insert Data

load GlassA1data;

X = GlassA1data;
dataRowNumber = size(GlassA1data,1);

crossValidationFolds = 5;
numberOfRowsPerFold = dataRowNumber / crossValidationFolds;

crossValidationTrainData = [];
crossValidationTestData = [];
for startOfRow = 1:numberOfRowsPerFold:dataRowNumber
    testRows = startOfRow:startOfRow+numberOfRowsPerFold-1;
    if (startOfRow == 1)
        trainRows = [max(testRows)+1:dataRowNumber];
        else
        trainRows = [1:startOfRow-1 max(testRows)+1:dataRowNumber];
    end

    %crossValidationTrainData = [crossValidationTrainData ;
SortedData(trainRows ,:)];
    crossValidationTrainData = [crossValidationTrainData ; X(trainRows ,:)];
    %crossValidationTestData = [crossValidationTestData ;SortedData(testRows
,:)];
    crossValidationTestData = [crossValidationTestData ;X(testRows ,:)];
end
k=[3 25 40];
r=172;
t=43;

result1=knnclassifier(crossValidationTrainData(1:r,1:9),crossValidationTrainD
ata(1:r,10),crossValidationTestData(1:t,1:9), k(1));
%prec_rec(result1, crossValidationTestData(1:t,10));
%%%%%conf1=confusionmatStats(crossValidationTestData(1:t,10),result1);
%Eval1=Evaluate(crossValidationTestData(1:t,10),result1);
%conf11=CopconfusionmatStats(crossValidationTestData(1:t,10),result1);

% Compute empirical curves
```

```
[TPR_emp, FPR_emp, PPV_emp] =
prc_stats_empirical(crossValidationTestData(1:t,10)', result1');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
result2=knnclassifier(crossValidationTrainData(r+1:r*2,1:9),crossValidationTr
ainData(r+1:r*2,10),crossValidationTestData(t+1:t*2,1:9), k(1));
%prec_rec(result2, crossValidationTestData(t+1:t*2,10));
%%conf2=confusionmatStats(crossValidationTestData(t+1:t*2,10),result2);
%Eval2=Evaluate(crossValidationTestData(t+1:t*2,10),result2);
%conf22=CopconfusionmatStats(crossValidationTestData(t+1:t*2,10),result2);

% Compute empirical curves
[TPR_emp2, FPR_emp2, PPV_emp2] =
prc_stats_empirical(crossValidationTestData(t+1:t*2,10)', result2');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

result3=knnclassifier(crossValidationTrainData(r*2+1:r*3,1:9),crossValidation
TrainData(r*2+1:r*3,10),crossValidationTestData(t*2+1:t*3,1:9), k(1));
%prec_rec(result3, crossValidationTestData(t*2+1:t*3,10));
%%conf3=confusionmatStats(crossValidationTestData(t*2+1:t*3,10),result3);
%Eval3=Evaluate(crossValidationTestData(t*2+1:t*3,10),result3);
%%conf33=CopconfusionmatStats(crossValidationTestData(t*2+1:t*3,10),result3);

% Compute empirical curves
[TPR_emp3, FPR_emp3, PPV_emp3] =
prc_stats_empirical(crossValidationTestData(t*2+1:t*3,10)', result3');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

result4=knnclassifier(crossValidationTrainData(r*3+1:r*4,1:9),crossValidation
TrainData(r*3+1:r*4,10),crossValidationTestData(t*3+1:t*4,1:9), k(1));
%prec_rec(result4, crossValidationTestData(t*3+1:t*4,10));
%%conf4=confusionmatStats(crossValidationTestData(t*3+1:t*4,10),result4);
%Eval4=Evaluate(crossValidationTestData(t*3+1:t*4,10),result4);
%conf44=CopconfusionmatStats(crossValidationTestData(t*3+1:t*4,10),result4);

% Compute empirical curves
[TPR_emp4, FPR_emp4, PPV_emp4] =
prc_stats_empirical(crossValidationTestData(t*3+1:t*4,10)', result4');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

result5=knnclassifier(crossValidationTrainData(r*4+1:r*5,1:9),crossValidation
TrainData(r*4+1:r*5,10),crossValidationTestData(t*4+1:t*5,1:9), k(1));
%prec_rec(result5, crossValidationTestData(t*4+1:t*5,10));
%%conf5=confusionmatStats(crossValidationTestData(t*4+1:t*5,10),result5);
%Eval5=Evaluate(crossValidationTestData(t*4+1:t*5,10),result5);
%conf55=CopconfusionmatStats(crossValidationTestData(t*4+1:t*5,10),result5);

% Compute empirical curves
[TPR_emp5, FPR_emp5, PPV_emp5] =
prc_stats_empirical(crossValidationTestData(t*4+1:t*5,10)', result5');
```

```
%recallvector=[conf11.recall conf22.recall conf33.recall conf44.recall
conf55.recall];
%precisionvector=[conf11.precision conf22.precision conf33.precision
conf44.precision conf55.precision];

recallvector=[TPR_emp TPR_emp2 TPR_emp3 TPR_emp4 TPR_emp5];
precisionvector=[PPV_emp PPV_emp2 PPV_emp3 PPV_emp4 PPV_emp5];
FPRvector=[FPR_emp FPR_emp2 FPR_emp3 FPR_emp4 FPR_emp5];

[FPRvector_sorted indFPR]=sort(FPRvector);
recallvectoradapt=recallvector(indFPR);

[recallvector_sorted indrecall]=sort(recallvector);
precisionvectoradapt=precisionvector(indrecall);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Second k
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
result11=knnclassifier(crossValidationTrainData(1:r,1:9),crossValidationTrain
Data(1:r,10),crossValidationTestData(1:t,1:9), k(2));
%prec_rec(result1, crossValidationTestData(1:t,10));
%%%%%conf1=confusionmatStats(crossValidationTestData(1:t,10),result1);
%Eval1=Evaluate(crossValidationTestData(1:t,10),result1);
%conf11=CopconfusionmatStats(crossValidationTestData(1:t,10),result1);

% Compute empirical curves
[k2TPR_emp, k2FPR_emp, k2PPV_emp] =
prc_stats_empirical(crossValidationTestData(1:t,10)', result11');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
result22=knnclassifier(crossValidationTrainData(r+1:r*2,1:9),crossValidationT
rainData(r+1:r*2,10),crossValidationTestData(t+1:t*2,1:9), k(2));
%prec_rec(result2, crossValidationTestData(t+1:t*2,10));
%%conf2=confusionmatStats(crossValidationTestData(t+1:t*2,10),result2);
%Eval2=Evaluate(crossValidationTestData(t+1:t*2,10),result2);
%conf22=CopconfusionmatStats(crossValidationTestData(t+1:t*2,10),result2);

% Compute empirical curves
[k2TPR_emp2, k2FPR_emp2, k2PPV_emp2] =
prc_stats_empirical(crossValidationTestData(t+1:t*2,10)', result22');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

result33=knnclassifier(crossValidationTrainData(r*2+1:r*3,1:9),crossValidatio
nTrainData(r*2+1:r*3,10),crossValidationTestData(t*2+1:t*3,1:9), k(2));
%prec_rec(result3, crossValidationTestData(t*2+1:t*3,10));
%%conf3=confusionmatStats(crossValidationTestData(t*2+1:t*3,10),result3);
%Eval3=Evaluate(crossValidationTestData(t*2+1:t*3,10),result3);
%%conf33=CopconfusionmatStats(crossValidationTestData(t*2+1:t*3,10),result3);

% Compute empirical curves
```

```
[k2TPR_emp3, k2FPR_emp3, k2PPV_emp3] =
prc_stats_empirical(crossValidationTestData(t*2+1:t*3,10)', result33');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

result44=knnclassifier(crossValidationTrainData(r*3+1:r*4,1:9),crossValidatio
nTrainData(r*3+1:r*4,10),crossValidationTestData(t*3+1:t*4,1:9), k(2));
%prec_rec(result4, crossValidationTestData(t*3+1:t*4,10));
%%conf4=confusionmatStats(crossValidationTestData(t*3+1:t*4,10),result4);
%Eval4=Evaluate(crossValidationTestData(t*3+1:t*4,10),result4);
%conf44=CopconfusionmatStats(crossValidationTestData(t*3+1:t*4,10),result4);

% Compute empirical curves
[k2TPR_emp4, k2FPR_emp4, k2PPV_emp4] =
prc_stats_empirical(crossValidationTestData(t*3+1:t*4,10)', result44');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

result55=knnclassifier(crossValidationTrainData(r*4+1:r*5,1:9),crossValidatio
nTrainData(r*4+1:r*5,10),crossValidationTestData(t*4+1:t*5,1:9), k(2));
%prec_rec(result5, crossValidationTestData(t*4+1:t*5,10));
%%conf5=confusionmatStats(crossValidationTestData(t*4+1:t*5,10),result5);
%Eval5=Evaluate(crossValidationTestData(t*4+1:t*5,10),result5);
%conf55=CopconfusionmatStats(crossValidationTestData(t*4+1:t*5,10),result5);

% Compute empirical curves
[k2TPR_emp5, k2FPR_emp5, k2PPV_emp5] =
prc_stats_empirical(crossValidationTestData(t*4+1:t*5,10)', result55');


%recallvector=[conf11.recall conf22.recall conf33.recall conf44.recall
conf55.recall];
%precisionvector=[conf11.precision conf22.precision conf33.precision
conf44.precision conf55.precision];

k2recallvector=[k2TPR_emp k2TPR_emp2 k2TPR_emp3 k2TPR_emp4 k2TPR_emp5];
k2precisionvector=[k2PPV_emp k2PPV_emp2 k2PPV_emp3 k2PPV_emp4 k2PPV_emp5];
k2FPRvector=[k2FPR_emp k2FPR_emp2 k2FPR_emp3 k2FPR_emp4 k2FPR_emp5];

[k2FPRvector_sorted k2indFPR]=sort(k2FPRvector);
k2recallvectoradapt=k2recallvector(k2indFPR);

[k2recallvector_sorted k2indrecall]=sort(k2recallvector);
k2precisionvectoradapt=k2precisionvector(k2indrecall);




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Third k
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
result111=knnclassifier(crossValidationTrainData(1:r,1:9),crossValidationTrai
nData(1:r,10),crossValidationTestData(1:t,1:9), k(3));
%prec_rec(result1, crossValidationTestData(1:t,10));
%%%%%conf1=confusionmatStats(crossValidationTestData(1:t,10),result1);
```

```
%Eval1=Evaluate(crossValidationTestData(1:t,10),result1);
%conf11=CopconfusionmatStats(crossValidationTestData(1:t,10),result1);

% Compute empirical curves
[k3TPR_emp, k3FPR_emp, k3PPV_emp] =
prc_stats_empirical(crossValidationTestData(1:t,10)', result111');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
result222=knnclassifier(crossValidationTrainData(r+1:r*2,1:9),crossValidation
TrainData(r+1:r*2,10),crossValidationTestData(t+1:t*2,1:9), k(3));
%prec_rec(result2, crossValidationTestData(t+1:t*2,10));
%%conf2=confusionmatStats(crossValidationTestData(t+1:t*2,10),result2);
%Eval2=Evaluate(crossValidationTestData(t+1:t*2,10),result2);
%conf22=CopconfusionmatStats(crossValidationTestData(t+1:t*2,10),result2);

% Compute empirical curves
[k3TPR_emp2, k3FPR_emp2, k3PPV_emp2] =
prc_stats_empirical(crossValidationTestData(t+1:t*2,10)', result222');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

result333=knnclassifier(crossValidationTrainData(r*2+1:r*3,1:9),crossValidati
onTrainData(r*2+1:r*3,10),crossValidationTestData(t*2+1:t*3,1:9), k(3));
%prec_rec(result3, crossValidationTestData(t*2+1:t*3,10));
%%conf3=confusionmatStats(crossValidationTestData(t*2+1:t*3,10),result3);
%Eval3=Evaluate(crossValidationTestData(t*2+1:t*3,10),result3);
%%conf33=CopconfusionmatStats(crossValidationTestData(t*2+1:t*3,10),result3);

% Compute empirical curves
[k3TPR_emp3, k3FPR_emp3, k3PPV_emp3] =
prc_stats_empirical(crossValidationTestData(t*2+1:t*3,10)', result333');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

result444=knnclassifier(crossValidationTrainData(r*3+1:r*4,1:9),crossValidati
onTrainData(r*3+1:r*4,10),crossValidationTestData(t*3+1:t*4,1:9), k(3));
%prec_rec(result4, crossValidationTestData(t*3+1:t*4,10));
%%conf4=confusionmatStats(crossValidationTestData(t*3+1:t*4,10),result4);
%Eval4=Evaluate(crossValidationTestData(t*3+1:t*4,10),result4);
%conf44=CopconfusionmatStats(crossValidationTestData(t*3+1:t*4,10),result4);

% Compute empirical curves
[k3TPR_emp4, k3FPR_emp4, k3PPV_emp4] =
prc_stats_empirical(crossValidationTestData(t*3+1:t*4,10)', result444');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

result555=knnclassifier(crossValidationTrainData(r*4+1:r*5,1:9),crossValidati
onTrainData(r*4+1:r*5,10),crossValidationTestData(t*4+1:t*5,1:9), k(3));
%prec_rec(result5, crossValidationTestData(t*4+1:t*5,10));
%%conf5=confusionmatStats(crossValidationTestData(t*4+1:t*5,10),result5);
%Eval5=Evaluate(crossValidationTestData(t*4+1:t*5,10),result5);
%conf55=CopconfusionmatStats(crossValidationTestData(t*4+1:t*5,10),result5);
```

```
% Compute empirical curves
[k3TPR_emp5, k3FPR_emp5, k3PPV_emp5] =
prc_stats_empirical(crossValidationTestData(t*4+1:t*5,10)', result555');


%recallvector=[conf11.recall conf22.recall conf33.recall conf44.recall
conf55.recall];
%precisionvector=[conf11.precision conf22.precision conf33.precision
conf44.precision conf55.precision];

k3recallvector=[k3TPR_emp k3TPR_emp2 k3TPR_emp3 k3TPR_emp4 k3TPR_emp5];
k3precisionvector=[k3PPV_emp k3PPV_emp2 k3PPV_emp3 k3PPV_emp4 k3PPV_emp5];
k3FPRvector=[k3FPR_emp k3FPR_emp2 k3FPR_emp3 k3FPR_emp4 k3FPR_emp5];

[k3FPRvector_sorted k3indFPR]=sort(k3FPRvector);
k3recallvectoradapt=k3recallvector(k3indFPR);

[k3recallvector_sorted k3indrecall]=sort(k3recallvector);
k3precisionvectoradapt=k3precisionvector(k3indrecall);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%
% Plot results
cols = [200 45 43; 37 64 180; 0 176 80; 0 0 0]/255;

% Plot ROC curves
figure; hold on;
axis([0 1 0 1]); %// Adjust axes for better viewing
grid;
%plot(FPRvector_sorted, recallvectoradapt, '-o',k2FPRvector_sorted,
k2recallvectoradapt,'g',k3FPRvector_sorted, k3recallvectoradapt,'c*',
'linewidth', 2);
plot(FPRvector_sorted, recallvectoradapt, '-', 'color', cols(4,:),
'linewidth', 2);
plot(k2FPRvector_sorted, k2recallvectoradapt, '-o', 'color', cols(1,:),
'linewidth', 2);
plot(k3FPRvector_sorted, k3recallvectoradapt, '-', 'color', cols(2,:),
'linewidth', 2);


xlabel('FPR'); ylabel('TPR'); title('ROC curves');
set(gca, 'box', 'on');

% Plot PR(Precision Recall) curves
figure; hold on;
axis([0 1 0 1]); %// Adjust axes for better viewing
grid;
%plot(recallvector_sorted, precisionvectoradapt, '-o',k2recallvector_sorted,
k2precisionvectoradapt,'g',k3recallvector_sorted,
k3precisionvectoradapt,'c*','linewidth', 2);
plot(recallvector_sorted, precisionvectoradapt, '-', 'color', cols(4,:),
'linewidth', 2);
plot(k2recallvector_sorted, k2precisionvectoradapt, '-o', 'color', cols(1,:),
'linewidth', 2);
```

```
plot(k3recallvector_sorted, k3precisionvectoradapt, '-', 'color', cols(2,:),
'linewidth', 2);
xlabel('TPR (recall)'); ylabel('PPV (precision)'); title('PR curves');
legend({'k=3','k=25','k=40'},'FontSize',8,'FontWeight','bold')
set(gca, 'box', 'on');
%%%%%%%%%%%%%%%%%%%%%%%%
```

Knnclassifier function

==================

```
function result = knnclassifier(trainx, trainy,test, k)
class= unique(trainy);
N= size(test,1);
n1=length(trainy);
if ( n1 < k)
   error('You specified more neighbors than existed points.')
end

%// Use distance
for i=1:N
newpoint=test(i,:);
%%%dists = sqrt(sum(bsxfun(@minus, trainx, test(i,:)).^2, 2));Euclidean
distance
dists = sum(abs(trainx - ones(n1,1)*test(i,:)),2);
%dist= sqrt(sum((trainx - ones(n1,1)*test(i,:)).^2,2));
%dists = sqrt(sum(trainx-test(i,:)*ones(m1,m2)).^2, 2);

[d,ind] = sort(dists);
ind_closest = ind(1:k);
x_closest = trainx(ind_closest,:);
x_closest_class=trainy(ind_closest,:);
x_closesthist=hist(x_closest_class,class);
[c, best]= max(x_closesthist);

result(i,1)= class(best);
end
```

prc_stats function

==============

```
% Computes empirical statistics based on classification output.
%
% Usage:
%     [TPR, FPR, PPV, AUC, AP] = prc_stats_empirical(targs, dvs)
```

```
%
% Arguments:
%     targs: true class labels (targets)
%     dvs: decision values output by the classifier
%
% Return values:
%     TPR: true positive rate (recall)
%     FPR: false positive rate
%     PPV: positive predictive value (precision)
%     AUC: area under the ROC curve
%     AP: area under the PR curve (average precision)
%
% ----------------------------------------------------------------------------
function [TPR, FPR, PPV, AUC, AP] = prc_stats_empirical(targs, dvs)

    % Check input
    assert(all(size(targs)==size(dvs)));

    % Sort decision values and true labels according to decision values
    n = length(dvs);
    [dvs_sorted,idx] = sort(dvs,'ascend');
    targs_sorted = targs(idx);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    field1 = 'confusionMat';
if nargin < 2
    value1 = targs;
else
    value1 = confusionmat(targs,dvs);
end

numOfClasses = size(value1,1);
totalSamples = sum(sum(value1));

%[TP,TN,FP,FN,sensitivity,specificity,precision,f_score] =
deal(zeros(numOfClasses,1));
for class = 1:numOfClasses
   TP(class) = value1(class,class);
   tempMat = value1;
   tempMat(:,class) = []; % remove column
   tempMat(class,:) = []; % remove row
   TN(class) = sum(sum(tempMat));
   FP(class) = sum(value1(:,class))-TP(class);
   FN(class) = sum(value1(class,:))-TP(class);
end

field2 = 'accuracy';  value2 = (TP+TN)/(TP+TN+FP+FN);

for class = 1:numOfClasses

    TPR((class)) = TP(class)/(TP(class)+FN(class));
        FPR(class) = FP(class)/(FP(class)+TN(class));
        PPV(class) = TP(class)/(TP(class)+FP(class));

    f_score(class) = 2*TP(class)/(2*TP(class) + FP(class) + FN(class));
```

```
end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Inititalize accumulators
   % TPR = repmat(NaN,1,n+1);
    %FPR = repmat(NaN,1,n+1);
    %PPV = repmat(NaN,1,n+1);

    % Now slide the threshold along the decision values (the threshold
    % always lies in between two values; here, the threshold represents the
    % decision value immediately to the right of it)
    %for thr = 1:length(dvs_sorted)+1

      %  TP = sum(targs_sorted(thr:end)>0);
     %   FN = sum(targs_sorted(1:thr-1)>0);
       % TN = sum(targs_sorted(1:thr-1)<0);
        %FP = sum(targs_sorted(thr:end)<0);


        %TPR(thr) = TP/(TP+FN);
        %FPR(thr) = FP/(FP+TN);
        %PPV(thr) = TP/(TP+FP);
   % end

    % Compute empirical AUC
    %[tmp,tmp,tmp,AUC] = perfcurve(targs,dvs,1)%'ProcessNaN','addtofalse');

    % Compute empirical AP
    AP = abs(trapz(TPR(~isnan(PPV)),PPV(~isnan(PPV))));

end
```