Problem 1 – Parzen windows

1. In Parzen windows we estimated only the likelihood of each class and given prior, we can perform full classification. Such classification relies on MAP, which is the Bayesian decision rule under zero-one loss function.

   Show that the full classification can be reduced to the following rule: "To a given sample, assign category which gives majority vote of neighbors". Recall that for the cubic window function, majority vote of neighbors can be written as:

   (1)
   $$\forall i \in \{1, \dots c\}: \sum_{k=1}^{n_j} \Phi\left(\frac{x_k^j - x}{h}\right) \geq \sum_{k=1}^{n_i} \Phi\left(\frac{x_k^i - x}{h}\right)$$
   Where x is a test sample, c is the number of classes and n_i is the number of samples in class $i = 1,...,c$. The notation x_k^j means the k[th] sample from the j class.

   Hint: Start from $P(w_j \mid x) \geq P(w_i \mid x) \ \ \forall i \neq j$ and use Bayes rule to derive the rule in eq. 1.

2. This question has nothing to do with the first one. You are given some samples from unknown distribution, D={1, -3, 2, 4, 5, -8, 0, -1, -2, -4}.

   Given h=4, use the 0-1 window function and build a full PDF for this distribution (no code, only drawing).

   - What do you think the real distribution of the data is?
   - Use Dr. Google to find the MLE formulas and estimate the real distribution. Show the similarity between your PDF drawing and the real distribution PDF (can be drawn by desmos.com).

Problem 2 – PCA and KNN

In this problem, you'll implement a KNN classifier using the Fashion-MNIST dataset and PCA (also to implement on your own).

The dataset contains train (50k) and test (10k) images, where each image is 28x28 pixels. Each image also has a class from 10 possible classes (and the dataset is balanced).

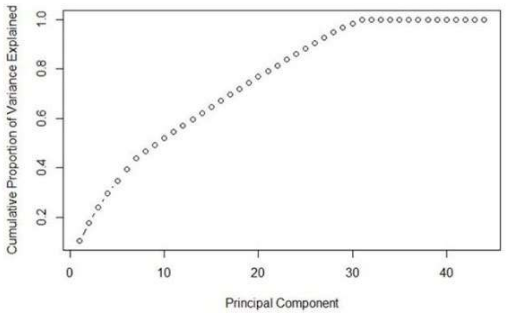
1. Load the dataset (you only need the CSV files – one for training and one for test) and display a random image from it (no need to include this in pdf).

2. Use PCA to reduce the dimension of the images from 784=28x28 to 81=9x9. Make sure you fit the PCA model only to the training set (but apply it to both training and test sets).

3. Pick another random image and show the result of applying PCA to it, and then try to recover the whole size again (use inverse_transform to recover the original vector from the compressed one).

   Make sure to include three plots in the report: the original image, the image after PCA and the image after the recovery.

4. Before training the model, use a function that given eigenvalues, draws a CDF of them **like** here:



   As seen in the tutorials, we use them to see how much "energy" we preserve from the data (when 1 means we didn't lose data at all). Use this to choose optimal dimension to reduce into (maybe differ than 81) and explain why you chose it.

   Attach the same image from step 3, before and after new dimensions reduction.

5. Now, you are ready to train the model. Find the optimal number of K (in terms of test accuracy). Warning: using KNN on all data at once is 50000x10000 matrix, which will not be allowed by your computer.

6. Print the accuracy of your model on the test set and attach it to the report. The printing should have the following structure:

```
D:\desktop_backup\ML\hws\2023\hw2\q2>python q2.py
Test accuracy is: 86.04%
```

Don't use a library implementation!

Problem 3 – Logistic Regression

In this problem, you will implement logistic regression via gradient descent.

The data set is taken from Andrew course on Coursera. The data consists of marks from two exams for 100 applicants and binary labels: 1 when applicant was admitted and 0 otherwise.

The objective is to build a classifier that can predict whether an application will be admitted to the university or not.

1. Read the data set into numpy array and split it into 90% training samples and 10% test samples. **Don't use** random_state, in order to get a different train each time.

2. Implement the function Logistic_Regression_via_GD(P,y,lr):

- Input: an np array 'P' of 'n' rows and 'd' columns, a label vector 'y' of 'n' entries and learning rate parameter 'lr'.
- Output: The function computes the output vector 'w' (and 'b') which minimzes the logistic regression cost function on 'P' and 'y'.
    - The implementation should be fully yours. **Don't use library implementation!**
    - It should be done by implementing Gradient descent (with 'lr' as the learning rate) to solve logistic regression

3. Call 'Logistic_Regression_via_GD(P,y,lr)', where 'P' and 'y' are the training data and the corresponding labels. Try to find the best lr for this task.

4. Implement the function Predict(w,b,p):

- Input: an input vector (numpy) which represents a sample, a vector (numpy) 'w' and a number 'b'.
- Output: the class prediction for 'p' of the logistic regression model defined by 'w' and 'b'.

5. In order to evaluate the accuracy of the model and since each running, we have different data, we will evaluate the model several times and get the average accuracy (this evaluation method is widely used in DL, but we do much more there). Report (print) the average test accuracy of your model (to the pdf, too).

6. Use the given plot function with the entire data and the **best** separating line from the epochs. Attach this to the report (**block** the calling in code before submission).

   Is it linearly separable? What classifier would separate best this data?

```
D:\desktop_backup\ML\hws\2023\hw2\q3>python q3.py
Avg test accuracy: 71.09756097560975%
```

Problem 4 - SVM

In this exercise you are given 4 toy datasets for binary classification in 2D and you are asked to build the best SVM classifier for each dataset.

- You need to download q4.py and svm_2d_data.mat
- Most of the implementation is provided in file q4.py.
    - It loads the datasets, plots the training data, and then plots the resulting decision regions (in color), boundaries (in a solid line), margins (in dotted line) and the support vectors in double circles.
    - It also plots the test set after the classification is done (in faded hue).
- These tools are provided to analyze your choice of the kernel (among 'linear', 'poly' (quadratic), and 'rbf'), slack parameter C, and gamma parameter for the width of the RBF kernel when used.
- Running the provided code will present the training data.
- **You need to** uncomment the bottom part of the code and insert the corresponding parameters for getting the plots associated with the classifier. The code also prints out the test accuracy. Try to reason about the type of the kernel using the shape of the classes and ***not*** by enumerating possibilities to maximize the test accuracy.
- **Save the final drawing and attach it to your answer sheet.**

Before submission, ***block*** (comment) the line plt.show()

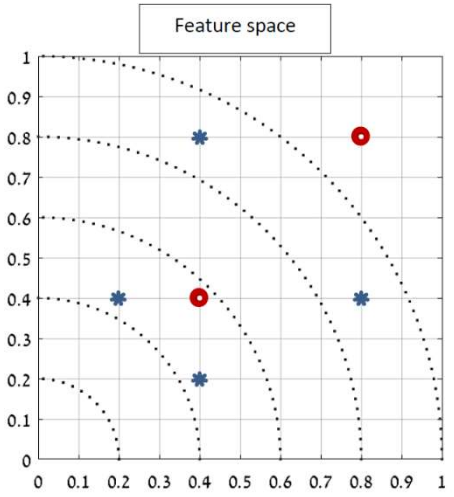
Problem 5 – Nonlinear SVM

Before starting, note that sections 4,5 are above the expected level in this course.

We now introduce the normalized linear kernel:

$$K(x,y) = \frac{x^t y}{\|x\| \cdot \|y\|}$$

1. Find the feature vector mapping, i.e., the φ that maps each sample to its feature space.



2. Given the following data, map the points to their new feature representations using the figure as the feature space. Draw the 6 new points (with colors) on the figure.

3. Draw the resulting margin decision boundary in the feature space. Also draw the separating line. You can use the following link.

4. For this section, work **only** with the new feature space.

   Given that the separating hyperplane is defined by $\ell: -x - y + 1.378 = 0$, so w=(-1,-1), b=1.378, find the alphas for each sample.

   Guide: start from the rule that $w = \sum_{i=1}^{3} y_i \alpha_i x_i$ to get 2 equations. Obtain the third equation by the rule $\sum_{i=1}^{3} y_i \alpha_i = 0$ and solve system of 3 equations (no need to show, but write the final alphas).

   Note: most of the time, it is not that simple, but since we have only 3 points in 2d, it works.

5. Draw the decision boundary in the original input space resulting from the normalized linear kernel. Recall that the nonlinear hyperplane given by:

$$\sum_{i=1}^{3} y_i \alpha_i K(x, x_i) + b = 0$$

   Where b is given to you, you found the alphas earlier. Use the following diagram: