CS 184A/284A: Artificial Intelligence in Biology and Medicine

Homework 1

Due date: **Oct. 17, 2025**

Instructor: Xiaohui Xie

This homework (and many subsequent ones) will involve data analysis and reporting on methods and results using Python code. We have provided a **Jupyter/iPython notebook** which you need to write both your code and reasoning for each question on the cell specific to that question.

You have to submit **`a single PDF file`** that contains everything to Gradescope, and associated each page of the PDF to each problem. This includes any text you wish to include to describe your results, the complete code snippets of how you attempted each problem, and any figures that were generated. It is important that you include enough detail that we know how you solved the problem, since otherwise we will be unable to grade it.

Jupyter/iPython notebooks provide an easy way to **export your results to a PDF** file [1]. I recommend liberal use of Markdown cells to create headers for each problem and sub-problem, explaining your implementation/answers, and including any mathematical equations. For parts of the homework you do on paper, scan it in such that it is legible (there are a number of free Android/iOS scanning apps, if you do not have access to a scanner), and include it as an image in the iPython notebook[2]. If you have any questions/concerns about using iPython, ask us on Ed Discussion.

Summary so far: (1) submit a single, standalone PDF report, with all code; (2) Use provided Jupyter notebooks.

Points: This homework adds up to a total of **55 points**, as follows:

Problem 0: Get Connected	5 points
Problem 1: Python & Data Exploration	20 points
Problem 2: kNN Predictions	25 points
Statement of Collaboration	5 points

Problem 0: Get Connected (5 points)

Please visit our class forum on Ed Discussion. Ed Discussion will be the place to post your questions and discussions, rather than by email to me or the TAs, since chances are that other students have the same or similar questions, and will be helped by seeing the discussion. Remember, your Ed Discussion participation will be taken into account for the participation grade as well. You do not need to mention anything regarding this in the report, we will be able to check whether you have visited Ed Discussion or not.

Problem 1: Python & Data Exploration (20 points)

In this problem, we will explore some basic statistics and visualizations of an example data set. First, install the required libraries (`NumPy`, `matplotlib`, and `Scikit-Learn`), then download the zip file for Homework 1, which contains the "Fisher iris" data set, and load the latter into Python:

```
import numpy as np
import matplotlib.pyplot as plt

iris = np.genfromtxt("data/iris.txt",delimiter=None)  # load the text file
Y = iris[:,-1]              # target value is the last column
X = iris[:,0:-1]            # features are the other columns
```

[1] For example, by doing a `Print Preview` in Chrome and `printing` it to a PDF.

[2] Tips from Gradescope: `http://gradescope-static-assets.s3-us-west-2.amazonaws.com/help/submitting_hw_guide.pdf`

The Iris data consist of four real-valued features used to predict which of three types of iris flower was measured (a three-class classification problem).

1. Use `X.shape` to get the number of features and the data points. Report both numbers, mentioning which number is which. `(5 points)`
2. For each feature, plot a histogram (`plt.hist`) of the data values `(5 points)`
3. Compute the mean & standard deviation of the data points for each feature (`np.mean`, `np.std`) `(5 points)`
4. For each pair of features (1,2), (1,3), and (1,4), plot a scatterplot (see `plt.plot` or `plt.scatter`) of the feature values, colored according to their target value (class). (For example, plot all data points with $y = 0$ as blue, $y = 1$ as green, etc.) `(5 points)`

Problem 2: kNN predictions (25 points)

In this problem, you will continue to use the Iris data and explore a KNN classifier using `KNeighborsClassifier` class from scikit-learn. While doing the problem, please explore the implementation to become familiar with how it works.

First, we will split the data into training and validation subsets:

```
iris = np.genfromtxt("data/iris.txt",delimiter=None)  # load the data
Y = iris[:,-1]
X = iris[:,0:-1]

# Note: indexing with ":" indicates all values (in this case, all rows);
#  indexing with a value ("0", "1", "-1", etc.) extracts only that value (here, columns);
#  indexing rows/columns with a range ("1:-1") extracts any row/column in that range.

from sklearn.model_selection import train_test_split

Xtr, Xva, Ytr, Yva = train_test_split(
    X, Y, test_size=0.25, random_state=0, stratify=Y, shuffle=True
) # split data into 75/25 train/validation
```

You may also find it useful to set the random number seed at the beginning (in general, for every assignment), e.g., `numpy.random.seed(0)`, to ensure consistent behavior each time.

You can build now and `train` a kNN classifier on `Xtr,Ytr` and make predictions on some data `Xva` with it:

```
from sklearn.neighbors import KNeighborsClassifier

K = 1   # K is an integer, e.g. 1 for nearest neighbor prediction
knn = KNeighborsClassifier(n_neighbors=K)  # create the knn classifier model
knn.fit(Xtr, Ytr)  # train the classifer
Yva_hat = knn.predict(Xva)  # get estimates of y for each data point in Xva
```

If your data are 2D, you can visualize a data set and a classifier's decision regions using the following function:

```

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

def plotClassify2D(model, X2d, Y, grid_step=0.02, padding=0.5):
    """
    Plot 2D decision regions for a trained classifier with consistent colors.

    Parameters
```

```
    ----------
    model : fitted classifier with .predict()
    X2d : (n_samples, 2)
    Y : (n_samples,)
    grid_step : mesh resolution
    padding : extra margin around data
    """
    # Define consistent color maps
    unique_classes = np.unique(Y)
    n_classes = len(unique_classes)

    # Use a subset of matplotlib's default tab10 colormap
    cmap = plt.cm.get_cmap('tab10', n_classes)
    colors = [cmap(i) for i in range(n_classes)]
    cmap_background = ListedColormap(colors)

    # Create grid
    x_min, x_max = X2d[:, 0].min() - padding, X2d[:, 0].max() + padding
    y_min, y_max = X2d[:, 1].min() - padding, X2d[:, 1].max() + padding
    xx, yy = np.meshgrid(
        np.arange(x_min, x_max, grid_step),
        np.arange(y_min, y_max, grid_step)
    )

    # Predict over grid
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    # Plot decision regions
    plt.contourf(xx, yy, Z, alpha=0.3, cmap=cmap_background)

    # Plot points with matching colors
    for i, c in zip(unique_classes, colors):
        plt.scatter(X2d[Y == i, 0], X2d[Y == i, 1],
                    c=[c], edgecolor='k', s=20, label=f"class {i}")

    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.legend(frameon=False)

plotClassify2D(knn, Xtr, Ytr)  # make 2D classification plot with data (Xtr,Ytr)
```

This function plots the training data and colored points as per their labels, then calls `knn`'s `predict` function on a densely spaced grid of points in the 2D space, and uses this to produce the background color.

1. Modify the code listed above to use only the first two features of X (e.g., let X be only the first two columns of `iris`, instead of the first four), and visualize (plot) the classification boundary for varying values of $K = [1, 5, 10, 50]$ using `plotClassify2D`. `(10 points)`

2. Again using only the first two features, compute the error rate (number of misclassifications) on both the training and validation data as a function of $K = [1, 2, 5, 10, 50, 100, 200]$. You can do this most easily with a for-loop:

```
K=[1,2,5,10,50,100];
for i,k in enumerate(K):
  learner = KNeighborsClassifier(...  # TODO: complete code to train model
  Yhat = learner.predict(...    # TODO: predict results on training data
  errTrain[i] = ...             # TODO: count what fraction of predictions are wrong
  #TODO: repeat prediction / error evaluation for validation data

```

```
plt.semilogx(...                #TODO: average and plot results on semi-log scale

```

Plot the resulting error rate functions using a semi-log plot (`semilogx`), with training error in red and validation error in green. Based on these plots, what value of K would you recommend? `(10 points)`

3. Provide the same plots as the previous, but with all the features in the dataset. Are the plots very different? Is your recommendation different? `(5 points)`

Statement of Collaboration (5 points)

It is **mandatory** to include a `Statement of Collaboration` in each submission, with respect to the guidelines below. Include the names of everyone involved in the discussions (especially in-person ones), and what was discussed.

All students are required to follow the academic honesty guidelines posted on the course website. For programming assignments, in particular, I encourage the students to organize (perhaps using Ed Discussion) to discuss the task descriptions, requirements, bugs in my code, and the relevant technical content `before` they start working on it. However, you should not discuss the specific solutions, and, as a guiding principle, you are not allowed to take anything written or drawn away from these discussions (i.e. no photographs of the blackboard, written notes, referring to Ed Discussion, etc.). Especially `after` you have started working on the assignment, try to restrict the discussion to Ed Discussion as much as possible, so that there is no doubt as to the extent of your collaboration.