# Assignment 1

**Due on October 26, 2016 (23:59:59)**

**Instructions.** There are two parts in this assignment. The first part involves a series of theory questions and the second part involves coding. The goal of this problem set is to make you understand and familiarize with k-Nearest Neighbor (k-NN) classification algorithm and Least Mean Squares estimation.

## Part I: Theory Questions

### k-Nearest Neighbor Classification

1. Let k-NN(S) be the k Nearest Neighbor classification algorithm on sample set S, which takes the majority of the closest k points.
   - Show that if in both 1-NN($S_1$) and 1-NN($S_2$) the label of point x is positive, then in 1-NN($S_1 \cup S_2$) the label of $x$ is positive.
   - Show an example such that in both 3-NN($S_1$) and 3-NN($S_2$) the label of x is positive, and in 3-NN($S_1 \cup S_2$) the label of x is negative.
2. Suppose you use 2 features $(F_1, F_2)$ for classification. While $F_1$ can take values between [500, 1000], $F_2$ can take 1 or 2 which also gives the correct class. For example you have [1 900] and [2 700 ] features as training set and you want to classify [1 600] point(which is originally belongs to class 1). But according to 1-NN (with Euclidean distance), input data belongs to class 2. What can cause this misclassification? Comment about the reason(s) and offer solution(s).

### Linear Regression

1. Suppose you have 50 training examples and you'll use 5 features (without bias). According to the normal equation $\theta = (X^T X)^{-1} X^T y$, what are the dimensions of $X, y, \theta$?
2. When you consider ridge regression $\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \Big\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \Big\}$ or the lasso $\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \Big\{ \frac{1}{2} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \Big\}$, what can you say about the effect of $\lambda$? What are the differences between these methods and linear regression with least squares?

### PART II: Image Classification

In this part, you will implement a method that predicts where the given photo is taken from. To achieve this you basically implement a method that consists of two steps: Least squares

estimation and k-NN methods. You'll use the least squares estimation method to learn a model that estimates the given image's GPS coordinates and then you'll use k-NN to predict the class of given image.

You have a dataset consists photographs which are taken in Rome and New York [1]. Each image has GPS coordinates information. For a given image you'll try to predict to where the photo is taken.

A dataset is provided for your training phase. Test images will be provided later and announced from Piazza group. Since test images will be provided later, you should use a subset of the training set as the test set. In other words, you should split your training dataset into two set: training set which will be used to learn model and validation set which will be used to measure the success of your model. You can use k-fold cross-validation method which is explained in the class.

**Dataset**

- You can download it from `https://www.dropbox.com/sh/o995qh0juku85w5/AABSCb_KYPqX4jE8hPv-cGSda?dl=0`.
- Dataset contains images from two classes: Rome and New York. Latitude and longitude information are provided in the dataset.
- Training set contains 27,680 Rome and 25,094 New York images.
- Latitude and longitude information are provided in the dataset.
- Dataset folder contains:
  - NY_Train folder : Contains images that are taken from New York.
  - Rome_Train folder : Contains images that are taken from Rome.
  - NY_Train.mat : Contains image name, latitude and longitude data for each image in the training set(NY_Train).
  - Rome_Train.mat : Contains image name, latitude and longitude data for each image in the training set(Rome_Train).
  - NY_Features.mat : Contains precomputed features for New York training set. Gist and tiny image (images are resized to 16,16,3]) features.
  - Rome_Features.mat : Contains precomputed features for Rome training set. Gist and tiny image (images are resized to 16,16,3]) features.

**Features**

- There is no limitation about features. You can use any feature that you think it's proper for your classification assignment. Some features are listed below:
  - Tiny images: You can resize images to a very small size (for example [16,16,3] as provided) and use raw RGB values as feature.
  - Color histograms: You can generate a RGB or CIE L*a*b* color histogram for each image. You can use openCV library to extract histogram of image via

*cv2.calcHist()* method (`http://docs.opencv.org/3.1.0/d1/db7/tutorial_py_histogram_begins.html`) or any library you want.

  - Gist descriptors: The gist descriptors has been shown to work well for scene classification. You can use gist descriptors which are provided.

- You can use more than one feature by concatenating.

**Steps to follow**

1. Extract feature for each image in training set (Gist, color histogram etc.).
2. Learn the weights of your estimator via least mean squares estimation method by using GPS coordinates as the target (or label).
3. For a given test image, you will try to estimate it's coordinates.
4. Use predicted coordinates to predict the class of image (taken in Rome or New York). For this part, you'll use the k-NN method. You'll determine images which have the closest GPS coordinates from the training set and decide the class of test image.
5. You can only consider the most similar image according to GPS coordinates or the first $k$ number of similar image and decide according to the majority. You can use FLANN (`http://www.cs.ubc.ca/research/flann/`) or scikit-learn library (`http://scikit-learn.org/stable/modules/classes.html#module-sklearn.neighbors`) for k-NN.
6. Finally you will compute accuracy of your model to measure the success of your classification method:

   **Accuracy = 100 * ((number of correctly classified examples) / (number of examples))**

   You will report mean accuracy by averaging your accuracy results for k folds.

## Submit

- report.pdf (PDF file containing your report)
- code/ (directory containing all your codes as Python file .py)

The ZIP file will be submitted via the department's submission system.

**NOTE:** To enter the competition, you have to register kaggle in Class with your department email account. The webpage of the competition will be announced later.

## Grading

- Code (50): Least-Squares: 40 points, k-NN: 10

- Report(50): Theory part: 12 points, Analysis of the results for prediction: 38 points.

  **Notes for the report**: You should explain your choices and their effects to the results. You can create a table to report your results like:

| Feature | Accuracy |
|---|---|
| Gist | 0.7 |
| Histogram | 0.3 |

### Late Policy

You may use up to five extension days (in total) over the course of the semester for the three problem sets you will take. Any additional unapproved late submission will be weighted by 0.5. You have to submit your solution in (rest of your late submission days + 5 days), otherwise it will not be evaluated.

### Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.

## References

[1] Chao-Yeh Chen and Kristen Grauman, Clues from the Beaten Path: Location Estimation with Bursty Sequences of Tourist Photos, *Proceedings of the IEEE Conf.on Computer Vision and Pattern Recognition (CVPR)*, 2011,