**CS 441**

# HW 1: Intro to Classification and Regression

## Due Date: Feb 6, 2023 11:59pm

In this assignment we will explore a few of the most fundamental machine learning algorithms on two problems: handwritten digit recognition and temperature prediction. The aims of this homework are:

1. Become familiar with KNN, Naive Bayes, Linear Logistic Regression, and Linear Regression methods.
2. Gain practice using correct methodologies for parameter selection and testing
3. Gain experience in classification and regression applications

Read this document and the report template and tips and tricks before beginning to code.

- Report template
- Tips and Tricks
- Starter code and data

## 1. MNIST Digit Classification

MNIST is a dataset of handwritten digits created in 1998. The task is to learn to classify an image into a digit from "0" to "9". The samples are collected from employees of the US Census Bureau and high school students. Each sample is a 28x28 image with values from 0 to 255. We can reshape and rescale it to a 784 dimensional vector with values from 0 to 1. There are 60,000 training images, and 10,000 test images. In this homework, we will treat 10,000 images of the training images as validation for initial testing and parameter selection. **For parts a and b, use the validation set to measure error, not the test set.**

### a. Implement and test KNN, Naive Bayes (NB), and Linear Logistic Regression (LLR)

For KNN, use K=1 and L2 distance. For NB, threshold the values with a threshold of 0.5 so that each feature value is binary (i.e. $x > 0.5$ are the features). Use a prior of $\alpha = 1$ count (i.e., initialize counts to 1 for each value of each feature for each class). You can assume $P(y)$ is uniform. *Implement KNN and Naive Bayes from scratch.* For LLR, use `sklearn.linear_model.LogisticRegression` with default parameter values.

Using the full train and validation sets, report the average validation error, the training time in seconds required to prepare your model, and the average inference time in milliseconds to predict the label for one example. You are not graded on the speed of training or inference, but they must be reported. You can estimate inference time based on a smaller sample size. Also report the confusion matrix for KNN only.

Try visualizing the errors by displaying the digits that are misclassified along with their predicted labels. This does not need to be reported, but can help provide an understanding of how to improve performance.

### b. Plot Error vs. Training Size

Different ML algorithms have different sensitivities to the amount of training data. Report the training and validation error while varying the number of training examples using the “xs” (50), “s” (500), “m” (5000), and “all” (50000) indices provided in the starter code. Use the same parameters as in part 1.

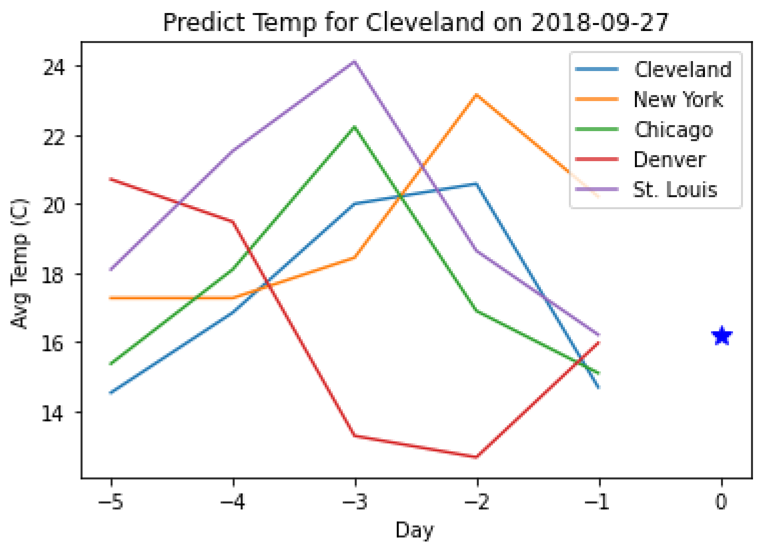
*Reflect:* Which method is most/least sensitive to training size? Why do you think that is? If you had 10x more training examples, what performance would you predict for each method? What if you had 100x more examples? (“Reflect” suggestions are to improve your understanding, but you don’t need to report the answers.)

### c. Parameter Selection

For each algorithm, use the **validation** set to select the best regularization parameter/prior for the “s” training set with 500 samples. Select among KNN (varying K), NB (varying $\alpha$), and LLR (varying *C*). After determining the best parameter for each method, evaluate on the **test** set and report the best parameter value, validation error, and test error for each method.

## 2. Temperature Regression

The Temperature Regression problem is derived from a Kaggle dataset. The task is to predict the average temperature in Cleveland for the next day given the average temperature of major US cities in the preceding five days. **Use the validation set to measure error, not the test set.**



### a. Implement and test KNN, Naive Bayes (NB), and Linear Regression (LR)

For KNN, use L2 distance and K=3, averaging the predictions from the 3 nearest neighbors. For Naive Bayes, model $P(X|y) = \Pi_f Gaussian(y - X_f)$, that is the difference of each feature and the target is a one-dimensional Gaussian function. You can assume $P(y)$ is uniform. For linear regression, use `sklearn.linear_model.Ridge` (which uses L2 regularization) with default parameters.

*Reflect:* Which method performs best here? Why do you think that is?

### b. Most important features

Identify the most important features by: (1) fitting a L1 Linear Regression model (`sklearn.linear_model.Lasso`) with default parameters; and (2) selecting the features with the highest magnitude coefficients (`model.coef_` is the linear coefficient vector). How many feature coefficients have magnitude greater than 0.001? For the top 10 features, report their indices and corresponding city and day. Also report the error rates of each algorithm on when trained using only those features.

*Reflect*: Did the feature selection reduce error? For which algorithms did it make the biggest difference? Why do you think it made the biggest difference for those algorithms? Think about the cities and days of the best features – does it tell you anything about how the temperature tends to move across the country?

## 3. Stretch Goals

For the first two stretch goals, try to improve MNIST classification or Temperature regression performance through parameter selection, feature normalization, data pre-processing, or any other means. Your algorithm must still be KNN, NB, or LLR/LR. You must design your method using only the train/val sets. Then, **test only once on the test set** with the method selected based on validation performance. If your results are invalid due to fitting the test data or other bugs, no points will be awarded.

**a. Improve MNIST Classification Performance**

Report your validation and test classification errors and the method. Points: 5 for effort; +5 for test error < 2.7%, +5 for test error < 2.3%, +5 for test error < 1.9%.

**b. Improve Temperature Regression Performance**

Report your validation and test RMS errors and the method. Points: 5 for effort; +5 for test RMSE < 2.05, +5 for RMSE < 1.95, +5 for RMSE < 1.85.

**c. Generate a train/test classification dataset in which Naive Bayes outperforms 1-NN and Logistic Regression**

Generate a x_train, y_train, x_test, y_test, in which a trained Naive Bayes outperforms 1-NN and Logistic Regression on the test samples. Explain how you generated the data and report test performance for each method. The data should be synthetic, not resampled from existing datasets.

## Submission Instructions

Submit three files: (1) completed report template; (2) an html or pdf version of your Jupyter notebook; and (3) a zip of your notebook and any other python files needed to run your code. Be sure to include your name, number of points claimed, and acknowledgments. **The report is your primary deliverable**. The notebook and code are included for verification/clarification by the grader.