ITCS 3153 Introduction to Artificial Intelligence

Homework 1 **Due: September 26th, 2016 11:59 pm**

In this homework, you will start to work on implementing code to perform basic tasks in Python (Problem 1) and then move on to more complex tasks related to text processing and linear classifiers (Problem 2). In Problem 3, you will engage with the SVM classifier implementation and in Problem 4, you will work on unsupervised learning with k-means clustering.

For each problem, we have created a separate python file. You should submit solution files by modifying the problemX_firstname_lastname.py file and renaming using the file with your first name and last name. All solutions should be uploaded to Canvas by the specified deadline above. For this homework, we expect 4 solution python files and 1 output text file (more details below).

Problem 1 (10 points): *Python warmup*

Use the code blocks in problem1_firstname_lastname.py to implement your solutions. Descriptions of the problems and their expected input and output are given in the file.

Problem 2 (30 points): *Website Classification*

In this problem, you will implement a binary classifier to detect whether a website is a homepage (class +1) or not (class −1). The input consists of two strings, the URL and the title of the website.

You will only need to edit problem2_firstname_lastname.py (**Hint 1:** looking at util.py will be helpful), and that is the code file you will submit. (**Hint 2**: all feature vectors and weight vectors are represented as util.Counter objects).

In Problem 2a (10 points), You should implement the functions:
logisticLoss
logisticLossGradient
hingeLoss
hingeLossGradient
squaredLoss
squaredLossGradient

Problem 2b (10 points): you will implement a general stochastic gradient descent algorithm. Your implementation should work with any of the already implemented loss functions in Problem 2a.

You will start by implementing the *predict* function, which uses the current weights to predict the output of an input *x*. Run your learning algorithm by typing
python <yourfilename.py>

This will run your code on the default dataset and help you debug your code (**Hint 3:** you should get 0 error on both train and validation). You can then switch to *python <yourfilename.py> --dataset websites* and try out different loss functions that you implemented:
python <yourfilename.py> --dataset websites --loss hinge
python <yourfilename.py> --dataset websites --loss squared
python <yourfilename.py> --dataset websites --loss logistic

(**Hint 4:** You should be able to observe the objective function generally go down. If it does not, something is wrong).

The training error and the validation error should decrease too, but there will be some randomness. Eventually, your errors should be about 14% and 17%, respectively. You can also try tuning the various hyperparameters (see util.py for the options). We will evaluate your predictor on a different test set which you do not see, to make sure that you do not manually overfit your model. This also means that you should debug your algorithm on the training set, rather than the validation set, to avoid unintentionally overfitting.

Problem 2c (10 points): In Problem 2b you ran the *basicFeatureExtractor*, which looks at the URL alone. You should now implement your own *customFeatureExtractor* which should look at both the URL and the title of the website. You should run the program using:
python <yourfilename.py> --dataset websites --loss logistic --featureExtractor custom

Experiment with different features. You can pass the *--verbose 1* flag, which will print out more details about what kind of errors the algorithm is making. Error analysis is an important part of doing machine learning. By looking at the errors, you can add any features that capture phenomena, which are not modeled in the current implementation of your algorithm. You should also look at the weights file to look at the feature weights to see if you are learning a sensible model. When you are done setting your features and fine-tuning the algorithm, implement *setTunedOptions* to set the hyperparameters (e.g., stepsizes, loss functions) to what you think are good settings. We will call your *setTunedOptions*, call your learning algorithm, and then evaluate on a hidden test set. You can test your options with the command:
python <yourfilename.py> --dataset websites --featureExtractor custom --setTunedOptions True

(**Hint 5:** You should be able to get your validation error down to about 8% without much work)

Submit problem2_firstname_lastname.py as your solution for this problem set.

Problem 3 (40 points): *SVM Classifier*

In this problem, you will train and test an SVM classifier. We will not code an SVM classifier by hand; instead, we will use the scikit-learn module that provides an easy-to-use SVM.

The first step is to install scikit-learn on your machine and make sure you can call *import sklearn* without error in python. Third-party software/packages (when explicitly permitted by the instructor!) can save you time over implementing everything from scratch. Follow the installation instructions on scikit-learn website.

Fill in the *train* and *classify* method in problem3_firstname_lastname.py. Run your code with:

python dataClassifier.py -c <yourfilename>

(**Hint 6:** there are some configurable parameters in *svm.SVC(...),* especially the choice of the kernel function and its parameters. To improve performance of the classifier, try optimizing these parameters. A good choice of parameters would result in 100% accuracy on the training set and >70% accuracy on the validation set).

Submit problem3_firstname_lastname.py as your solution to this problem set.

Problem 4 (10 points): *K-means*

Implement the *k*-means function. You should initialize your *k* cluster centers to random elements of examples. After a few iterations of *k*-means, your centers should be very dense vectors. (**Hint 7:** your code is taking too long to terminate, make sure you perform any necessary pre-computations).

Submit problem4_firstname_lastname.py as your solution to this problem.