CS 178: Machine Learning: Spring 2020

Homework 3

Due Date: **Thursday, May 7, 2020**

The submission for this homework should be ***a single PDF file*** containing all of the relevant code, figures, and any text explaining your results. For many questions you will be completing missing sections in a Python file that we provide. Be sure to include copies of any code you write as answers to the appropriate question, so that it may be graded. Also, be sure to download and use the latest version of the mltools package provided with this homework.

Problem 1: Logistic Regression (75 points)

In this problem, we'll build a logistic regression classifier and train it on separable and non-separable data. Since it will be specialized to binary classification, we've named the class `logisticClassify2`. We start by creating two binary classification datasets, one separable and the other not:

```
iris = np.genfromtxt("data/iris.txt",delimiter=None)
X, Y = iris[:,0:2], iris[:,-1]  # get first two features & target
X,Y  = ml.shuffleData(X,Y)      # order randomly rather than by class label
X,_  = ml.transforms.rescale(X) # rescale to improve numerical stability, speed convergence

XA, YA = X[Y<2,:], Y[Y<2]       # Dataset A: class 0 vs class 1
XB, YB = X[Y>0,:], Y[Y>0]       # Dataset B: class 1 vs class 2
```

For this problem, we focus on the properties of the logistic regression learning algorithm, rather than classification performance. Thus we will not create a separate validation dataset, and simply use all available data for training.

1. For each of the two datasets, create a separate scatter plot in which the training data from the two classes is plotted in different colors. Which of the two datasets is linearly separable? *(5 points)*

2. Write (fill in) the function `plotBoundary` in `logisticClassify2.py` to compute the points on the decision boundary. In particular, you only need to make sure `x2b` is set correctly using `self.theta`. This will plot the data & boundary quickly, which is useful for visualizing the model during training. To demonstrate your function, plot the decision boundary corresponding to the classifier

$$\text{sign}(\ 0.5 - 0.25x_1 + 1.0x_2\ )$$

   along with dataset A, and again with dataset B. These fixed parameters should lead to an OK classifier on one data set, but a poor classifier on the other. You can create a "blank" learner and set the weights as follows:

```
import mltools as ml
from logisticClassify2 import *

learner = logisticClassify2();          # create "blank" learner
learner.classes = np.unique(YA)         # define class labels using YA or YB
wts = np.array([theta0,theta1,theta2]); # TODO: fill in values
learner.theta = wts;                    # set the learner's parameters
```

   Include the lines of code you added to the `plotBoundary` function, and the two generated plots. *(10 points)*

3. Complete the `logisticClassify2.predict` function to make predictions for your classifier. Verify that your function works by computing & reporting the error rate for the classifier defined in the previous part on both datasets A and B. (The error rate should be about 0.05 for one dataset, and higher for the other.) Note that in the code, the two classes are stored in the variable `self.classes`, where the first entry is the "negative" class (class 0) and the second entry is the "positive" class (class 1). You should create different learner objects for each dataset, and use the `learner.err` function. Your solution pdf should include the `predict` function implementation and the computed error rates. *(10 points)*

4. Verify that your `predict` and `plotBoundary` implementations are consistent by using `plotClassify2D` with your manually constructed learner on each dataset. This will call `predict` on a dense grid of points, and you should find that the resulting decision boundary matches the one you plotted previously. *(5 points)*

5. In the provided training code, we first transform the classes in the data Y into YY, with canonical labels for the two classes: "class 0" (negative) and "class 1" (positive). Let $r^{(j)} = x^{(j)} \cdot \theta = \sum_i x_i^{(j)} \theta_i$ denote the linear response of the classifier, and $\sigma(r)$ equal the standard logistic function:

   $$\sigma(r) = \big(1 + \exp(-r)\big)^{-1}.$$

   The logistic negative log-likelihood loss for a single data point j is then

   $$J_j(\theta) = -y^{(j)} \log \sigma(x^{(j)} \cdot \theta) \; - \; (1 - y^{(j)}) \log(1 - \sigma(x^{(j)} \cdot \theta)),$$

   where $y^{(j)}$ is 0 or 1. Derive the gradient of the negative log likelihood $J_j(\theta)$ for logistic regression (this will be needed to implement stochastic gradient descent in the next part). Include the gradient equation in your solutions. *Hint:* The logistic function has a simple derivative, $\sigma'(r) = \sigma(r)(1 - \sigma(r))$. *(15 points)*

6. Complete the `train` function to perform stochastic gradient descent on the logistic regression loss function. This will require that you fill in:
   (a) computing the response $r^{(j)}$ and gradient $\nabla J_j(\theta)$ associated with each data point $x^{(j)}, y^{(j)}$;
   (b) computing the overall loss function, $J = \frac{1}{m} \sum_j J_j$, after each pass through the full dataset (or epoch);
   (c) a stopping criterion that checks two conditions: stop when either you have reached `stopEpochs` epochs, or J has changed by less than `stopTol` since the last epoch.

   Include the complete implementation of `train` in your solutions. *(20 points)*

7. Run the logistic regression `train` algorithm on both datasets. Describe the parameter choices (step sizes and stopping criteria) you use for each dataset. Include plots showing the convergence of the surrogate loss and error rate as a function of the number of training epochs, and the classification boundary after the final training iteration. (The included `train` function creates plots automatically.) *(10 points)*

Plotting hints: The code generates plots as the algorithm runs, so you can see its behavior over time; this is done with `pyplot.draw()`. Run your code either interactively or as a script to see these display over time; refer to discussion notebooks on how to plot the loss over time in Jupyter.

Debugging hints: Debugging machine learning algorithms can be quite challenging, since the results of the algorithm are highly data-dependent, and often somewhat randomized (from the initialization, as well as the order points are visited by stochastic gradient descent). We suggest starting with a small step size and verifying both that the learner's prediction evolves slowly in the correct direction, and that the objective function J decreases. If that works, explore the convergence of the algorithm with larger step sizes. It is often useful to manually step through the code, for example by pausing after each parameter update using `input()`. Of course, you may also use a more sophisticated debugger.

Problem 2: Maximum Margin Classifiers (20 points)

In this question, we examine the properties of max-margin separating hyperplanes on toy, two-dimensional data.

1. First consider a dataset with $m = 4$ points. There are 2 examples of class $+1$, located at $x = (-1, -1)$ and $x = (+1, +1)$. There are 2 examples of class -1, located at $x = (-1, +1)$ and $x = (+1, -1)$.

   Define the non-linear feature mapping $\phi(x) = [x_1, x_1 x_2]$. Plot the four input points in this space, and the maximum margin separating hyperplane. What max-margin weight vector w would produce predictions $w \cdot \phi(x)$ that, when thresholded, perfectly classify the training data? What is the corresponding margin? *(5 points)*

2. The max-margin separator is the set of points x for which $w \cdot \phi(x) = 0$. Plot the max-margin separator from part 1 in the original input space. *(5 points)*

3. Now consider a different dataset with $m = 6$ points. There are 3 examples of class $+1$, located at $x = (1, 1)$, $x = (2, 2)$, and $x = (2, 0)$. There are 3 examples of class -1, located at $x = (0, 0)$, $x = (1, 0)$, and $x = (0, 1)$.

   Define the feature mapping $\phi(x) = [1, x_1, x_2]$, which adds a bias feature to the raw inputs. Plot the data and a maximum margin separating hyperplane in the original input space. What max-margin weight vector w would produce predictions $w \cdot \phi(x)$ that, when thresholded, perfectly classify the training data? What is the corresponding margin? *(5 points)*

4. For part 3, which data points are support vectors? If you remove one of these support vectors from the training data does the size of the optimal margin decrease, stay the same, or increase? Justify your answer for each of the support vectors. *(5 points)*

Problem 3: Statement of Collaboration (5 points)

It is **mandatory** to include a *Statement of Collaboration* in each submission, that follows 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 students to organize (perhaps using Piazza) to discuss the task descriptions, requirements, possible bugs in the support 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 (no photographs of the blackboard, written notes, referring to Piazza, etc.). Especially *after* you have started working on the assignment, try to restrict the discussion to Piazza as much as possible, so that there is no doubt as to the extent of your collaboration.