CS273a Homework #3

Machine Learning: Winter 2015

Due: Tuesday January 27th, 2015

Write neatly (or type) and show all your work!

Please remember to turn in at most two documents, one with any handwritten solutions, and one PDF file with any electronic solutions.

Download the provided Homework 3 code, and replace last week's code (several new functions have been added).

Problem 1: Perceptrons and Logistic Regression

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, I've named the class `logisticClassify2`.

We'll start by building two binary classification problems, one separable and the other not:

```
iris=load('data/iris.txt');     % load the text file
X = iris(:,1:2); Y=iris(:,end); % get first two features
[X Y] = shuffleData(X,Y);       % reorder randomly
X  = rescale(X);                % works much better for rescaled data
XA = X(Y<2,:); YA=Y(Y<2);       % get class 0 vs 1
XB = X(Y>0,:); YB=Y(Y>0);       % get class 1 vs 2
```

For this problem, we are focused on the learning algorithm, rather than performance – so, we will not bother creating training and validation splits; just use all your data for training.

Note: Be sure to shuffle your data before doing SGD in part (f) – otherwise, if the data are in a pathological ordering (e.g., ordered by class), you may experience strange behavior and slow convergence during the optimization.

(a) Show the two classes in a scatter plot and verify that one is linearly separable while the other is not.

(b) Write (fill in) the function `@logisticClassify2/plot2DLinear.m` so that it plots the two classes of data in different colors, along with the decision boundary (a line). Include the listing of your code in your report. To demo your function plot the decision boundary corresponding to the classifier

$$\text{sign}(\ .5 + 1x_1 - .25x_2\ )$$

along with the A data, and again with the B data. You can create a "blank" learner and set the weights by:

```
learner=logisticClassify2();              % create "blank" learner
learner=setClasses(learner, unique(YA));  % define class labels using YA or YB
wts = [theta0 theta1 theta2];             % TODO: fill in values
learner=setWeights(learner, wts);         % set the learner's parameters
```

(c) Complete the `predict.m` function to make predictions for your linear classifier. Note that, in my code, the two classes are stored in the variable `obj.classes`, with the first entry being the

"negative" class (or class 0), and the second entry being the "positive" class. Again, verify that your function works by computing & reporting the error rate of the classifier in the previous part on both data sets A and B. (The error rate on data set A should be ≈ 0.0505.)

You can also test this and your previous function by comparing your `plot2DLinear` output with the generic `plotClassify2D` function, which shows the decision boundary "manually" by calling `predict` on a dense grid of locations, rather than analytically as your `plot2DLinear` function should do.

(d) In my provided code, I first transform the classes in the data Y into "class 0" (negative) and "class 1" (positive). In our notation, let $z = \theta x^{(i)}$ is the linear response of the perceptron, and σ is the standard logistic function

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

The (regularized) logistic negative log likelihood loss for a single data point j is then

$$J_j(\theta) = -y^{(j)} \log \sigma(\theta x^{(j)T}) \;-\; (1 - y^{(j)}) \log(1 - \sigma(\theta x^{(j)T})) \;+\; \alpha \sum_i \theta_i^2$$

where $y^{(j)}$ is either 0 or 1. Derive the gradient of the regularized negative log likelihood J_j for logistic regression, and give it in your report. (You will need this in your gradient descent code for the next part.)

(e) Complete your `train.m` function to perform stochastic gradient descent on the logistic loss function. This will require that you fill in:
(1) computing the surrogate loss function at each iteration ($J = \frac{1}{m} \sum J_j$, from the previous part);
(2) computing the prediction and gradient associated with each data point $x^{(i)}, y^{(i)}$;
(3) a gradient step on the parameters θ;
(4) a stopping criterion (usually either `stopIter` iterations or that J has not changed by more than `stopTol` since the last iteration through all the data).

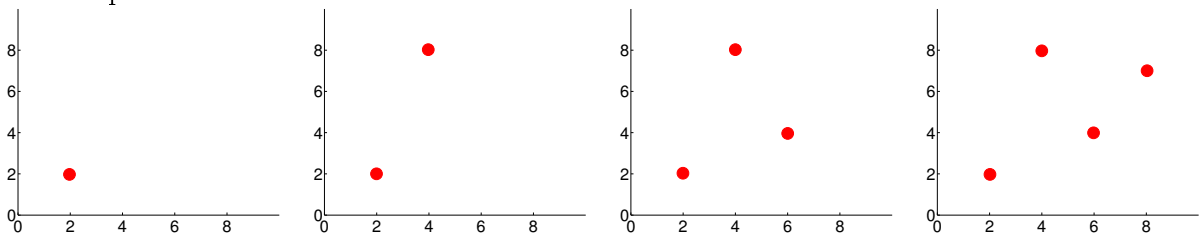
(f) Run your logistic regression classifier on both data sets (A and B); for this problem, use no regularization ($\alpha = 0$). Describe your parameter choices (stepsize, etc.) and show a plot of both the convergence of the surrogate loss and error rate, and a plot of the final converged classifier with the data (using e.g. `plotClassify2D`). In your report, please also include the functions that you wrote (at minimum, `train.m`, but possibly a few small helper functions as well).

Note: Debugging machine learning algorithms can be quite challenging, since the results of the algorithm are highly data-dependent, and often somewhat randomized (initialization, etc.). I suggest starting with an extremely small step size and verifying both that the learner's prediction evolves slowly in the correct direction, and that the objective function J decreases monotonically. If that works, go to larger step sizes to observe the behavior. I often use the `pause` command to slow down execution so that I can examine my code's behavior; you can also step through the code using Matlab's debugger.

Problem 2: Shattering and VC Dimension

Consider the following learners and data points, which have two real-valued features x_1, x_2. Which of the following four examples can be shattered by each learner? Give a brief explanation / justification and use your results to guess the VC dimension of the classifier. (You do not have to give a formal proof, just your reasoning.)

Data points:



(a) (b) (c) (d)

For the two learners, $T[z]$ is the sign threshold function, $T[z] = +1$ for $z \geq 0$ and $T[z] = -1$ for $z < 0$. The learner parameters a, b, c are real-valued scalars, and each data point has two real-valued input features x_1, x_2.

(a) $T(\, a + bx_1 \,)$

(b) $T(\, (x_1 - a)^2 + (x_2 - b)^2 + c \,)$

(c) $T(\, (a * b)x_1 + (c/a)x_2 \,)$