Machine Learning 10-701 Feb 23, 2015
http://alex.smola.org/teaching/10-701-15/ due on Mar 9, 2015
Carnegie Mellon University

Homework 6

START HERE: Instructions

- The homework is due at 11:59pm on Mar 9th, 2015. Anything that is received after that time will be considered as late submission.
- Answers to everything but the coding question will be submitted electronically (e.g. as a PDF or handwritten and scanned).
- Please follow the instruction for code submission in problem correctly.
- The handout for question can be found on Autolab or here.
- Collaboration on solving the homework is allowed (after you have thought about the problems on your own). However, when you do collaborate, you should list your collaborators! You might also have gotten some inspiration from resources (books or online etc...). This might be OK only after you have tried to solve the problem, and couldn't. In such a case, you should cite your resources.
- If you do collaborate with someone or use a book or website, you are expected to write up your solution independently. That is, close the book and all of your notes before starting to write up your solution.

This is a programming assignment. You are asked to implement several algorithms in Octave. Octave is a free scientific programming language, with syntax (almost) identical to that of Matlab. Installation instructions can be found on the Octave website as linked above. If you've never used Octave or Matlab before, you may wish to check out this tutorial or this one.

For each question you will be given a single function signature, and asked to write a single Octave function which satisfies the signature. Please do *not* change the names of the functions and their variables. Do *not* modify the structure of the directories.

The problems are automatically graded using the CMU Autolab system. The code which you write will be executed remotely against a suite of tests, and the results used to automatically assign you a grade. In order for your code to execute correctly on our servers you should avoid using libraries beyond the *basic* octave libraries.

The Autolab interface for this course can be found at https://autolab.cs.cmu.edu/. You can sign in using your andrew credentials. You should make sure to edit your account information and choose a nickname/handle. This handle will be used to display your results for the challenge question on the class leaderboard.

We have provided you with a single folder containing each of the functions you need to complete. *Do not modify the structure of this directory or rename these files*. Complete each of these functions, then compress this directory as a tar file and submit to autolab online. You may submit as many times as you like (up to the due date).

Please *only* submit a tarball called "code.tar", which is compressed from "code" folder. Do not submit any other files such as data files.

If you have a question, please post it on Piazza https://piazza.com/class/i4ivtbjbrt219e. If you discover a bug or want to talk other technical issues, please send an email to Jin Sun jins@andrew.cmu.edu.

Notations

- **XTrain**: $\mathcal{R}^{nTrain \times f}$ is a matrix of training data, where each row is a training instance with f features.
- **XTest**: $\mathcal{R}^{nTest \times f}$ is a matrix of test data, where each row is a test instance with f features.
- **yTrain**: $\mathcal{R}^{nTrain \times 1}$ is a vector of labels (for classification) or real numbers (for regression).

Machine Learning 10-701 Feb 23, 2015
http://alex.smola.org/teaching/10-701-15/ due on Mar 9, 2015
Carnegie Mellon University

Homework 6

General Instructions

- Always pad a column of ones to data points as the first feature. With the bias term, the decision boundary does not have to go across the origin. The weight vector has an extra dimension because of padding.
- Always initialize the weight vector to be all zeros.
- The labels are always -1 or +1. If your prediction is exactly 0 in training phase, treat it as a mistake and update the weight vector. If it happens in testing phase, please convert it to +1.

1 Perceptron [20 pts]

In this problem you are asked to implement the perceptron algorithm.

- Learn Perceptron Weights [10 pts]
  Complete function $pTrain(XTrain, yTrain, nIter)$ which returns w, a $\mathcal{R}^{f+1}$ vector. $nIter$ is the number of iterations that the perceptron algorithm runs. In each iteration, the perceptron is trained from the first data point to the last data point.
- Classify with the Perceptron algorithm [10 pts]
  Complete function $pClassify(XTest, w)$ which returns y, a vector of predicted labels (-1 or +1). w is the trained weight vector with $f + 1$ dimensions.

2 Margin Perceptron [30 pts]

In this problem you are asked to implement margin perceptron. As you may noticed, the original perceptron algorithm will never converge for linearly inseparable cases. However, the algorithm can be generalized by "The Delta Trick". We can make the data to be linearly separable by adding an "id" feature to each data point.

$$
\begin{aligned}
&\mathbf{x}^1 = (x_1^1, x_2^1, \ldots, x_f^1) \rightarrow (x_1^1, x_2^1, \ldots, x_f^1, \overbrace{\Delta, 0, \ldots, 0}^{n \text{ new features}}) \\
&\mathbf{x}^2 = (x_1^2, x_2^2, \ldots, x_f^2) \rightarrow (x_1^2, x_2^2, \ldots, x_f^2, 0, \Delta, \ldots, 0) \\
&\ldots \\
&\mathbf{x}^n = (x_1^n, x_2^n, \ldots, x_f^n) \rightarrow (x_1^n, x_2^n, \ldots, x_f^n, 0, 0, \ldots, \Delta)
\end{aligned}
$$

Figure 1: The Delta Trick

In the training process, we add n dimensions to the weight vector, and update the weight vector if $\frac{y^i w^T x^i}{\|w\|} \leq \gamma$, where w is the extended weight vector. A short version of the convergence proof is as follows:

Homework 6

Let $d_i = \max(0, \gamma - y^i x^i w)$, where $\|w\| = 1$ and $\gamma > 0$. Let $w' = (w_1, w_2, \ldots, w_f, \frac{y^1 d_1}{\Delta}, \ldots, \frac{y^n d_n}{\Delta})/Z$, where $Z = \sqrt{1 + \frac{D^2}{\Delta^2}}$ for $D = \sqrt{d_1^2 + d_2^2 + \ldots + d_n^2}$. Now the perceptron is separable by w' with normalized margin $\frac{\gamma}{Z}$. The mistake bound is $\frac{(R^2+\Delta^2)Z^2}{\gamma^2}$. The bound is minimized if we set $\Delta = \sqrt{RD}$.

The implementation is very similar as the first problem. You don't need to keep $\|w\| = 1$ and compute d_i. Those setups are just for the convenience of the proof.

- Learn Weights [20 pts]
  Complete function $mpTrain(XTrain, yTrain, gamma, nIter)$ which returns w, a $\mathcal{R}^{1+f+nTrain}$ vector. Pad a column of ones to $XTrain$ as its first feature. $gamma$ is the margin. $nIter$ is the number of iterations that the perceptron algorithm runs. Let $\Delta = 1$. In each iteration, the perceptron is trained from the first data point to the last data point. To avoid divide by zero warning, directly update the weight vector if $\|w\| = 0$.

- Classify with Margin Perceptron [10 pts]
  Complete function $mpClassify(XTest, w)$ which returns y, a vector of predicted labels (-1 or +1). w is the trained vector of $\mathcal{R}^{1+f+nTrain}$. Ignore the extended terms during classification.

3 Kernel Perceptron [30 pts]

At an abstract level kernel perceptrons work by projecting the features into a high dimensional feature space, then doing normal perceptron learning in the high dimensional feature space. i.e. $\forall x$ we replace x with $\phi(x)$, then run perceptron learning using $\phi(x)$ as our features.

Unfortunately it is computationally infeasible to do this explicitly. Instead we work in the high dimensional space implicitly using inner products. We note that given a data point $\phi(x)$ and a weight vector w, we classify $\phi(x)$ by calculating $\langle \phi(x), w \rangle$. Furthermore we know w is a linear combination of the points $\phi(x_1), \ldots, \phi(x_n)$ (i.e. $w = \sum_i \alpha_i \phi(x_i)$ for some $\alpha_1, \ldots, \alpha_n$). Therefore $\langle \phi(x), w_i \rangle = \sum_i \alpha_i \langle \phi(x), \phi(x_i) \rangle$. Therefore all we really need to do is to calculate the inner products between points, and keep track of the α values. (credit to Adona Iosif)

Note in this question we use a dth degree polynomial kernel with inner product $\langle \phi(x_i), \phi(x_j) \rangle = \langle x_i, x_j \rangle^d$. You don't need to consider margins in this problem.

- Learn Weights [20 pts]
  Complete function $kpTrain(XTrain, yTrain, d, nIter)$ which returns $alpha$, a $\mathcal{R}^{nTrain}$ vector, whose elements are the multipliers for each example learned by the kernel perceptron algorithm. Note that the perceptron weight vector w is implicitly represented through these α values. d is the order of the poly kernel. $nIter$ is the number of iterations the perceptron algorithm runs.

- Classify with Kernel Perceptron [10 pts]
  Complete function $kpClassify(XTrain, XTest, alpha, d)$ which returns y, a vector of predicted labels (-1 or +1).

4 Leaderboard Problem [10+10pts]

Complete the function $run\_perceptron(XTrain, yTrain, XTest)$ which returns y, a vector of predicted labels (-1 or +1). Implement your own perceptron algorithm to win the leaderboard! You are not able to call any functions you previously implemented, so please write everything in this file. Training data can

be found in the data folder in the handout. Students who achieve the baseline accuracy (65%) will get 10 points. Students who rank top 10 on the leaderboard will get 10 bonus points.

- **Implementing other classifiers will give you zero points for this entire assignment.** Do not cheat!
- The dataset is used for classifying if patients have diabetes.
- Normalize features.
- Always be aware of the model selection issue.
- You may want to implement a margin perceptron with kernels. Here are some useful kernel functions.
- You can add a learning rate to the update rule.
- For linearly inseparable case, perceptrons will never converge. Trying weighted perceptron or voted perceptron may be a good idea.
- For linearly inseparable case, perceptrons may have cyclic behaviors. Use random sequenced training data for each iteration may be a good idea. Set a seed for random number generator for debugging.
- Do not have too many iterations. Otherwise you will get timeout on Autolab.
- Check out an excellent paper by Aaditya Ramdas.