Machine Learning 10-701 October 7, 2013
http://alex.smola.org/teaching/cmu2013-10-701x/ due October 21, 2013
Carnegie Mellon University

Homework 2

START HERE: Instructions

- The homework is due at 10:30am on October 21, 2013. Anything that is received after that time will not be considered.
- Answers to everything but question 3 will be submitted electronically through the submission website: http://alex.smola.org/teaching/cmu2013-10-701x/submission.html. Let us know if you have any problems.
- **Read this before handwriting or LaTeXing your solutions:** In HW1 some students reported difficulty with submitting large image files to the discussion/handin server. So, we recommend that, to the extent possible, you should not handwrite or LaTeXyour solutions; instead use "plain text" or "markup text" mode and type or paste your solutions into the compose box. We will make our best effort to provide support for image-based handins, but until further notice they should be considered an experimental feature.
- Please follow the instructions for code submission in problem 3 correctly. The code handout is at http://alex.smola.org/teaching/cmu2013-10-701x/assignments/assignment_2_handout.tar.
- 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.

1 Convexity [Dougal; 25 pts]

In this question we'll get some practice working with convex functions. You may want to refer to these notes on convexity: http://alex.smola.org/teaching/cmu2013-10-701x/convexity-notes.pdf.

1.1 Calculus of convex functions [6 pts]

Show that the following functions are convex. (Each proof should be only a few lines.)

(a) $h_1(x) = f(Ax + b)$ for convex f, matrix A, and vector b.

(b) $h_2(x) = \max(f(x), g(x))$ for convex f and g.

(c) $h_3(x) = g(f(x))$ for convex f with range $A \subseteq \mathbb{R}$ and $g : A \to \mathbb{R}$ both convex and non-decreasing.

1.2 First-order condition [7 pts]

Prove that $f : \mathbb{R} \to \mathbb{R}$, where $\operatorname{dom} f$ is an open set and f is continuously differentiable, is convex iff

$$\operatorname{dom} f \text{ is convex and } f(y) - f(x) \geq f'(x)\,(y - x) \text{ for all } x, y \in \operatorname{dom} f. \tag{1}$$

That is, show that (1) is equivalent to the "zeroth-order" condition

$$f(x + \lambda(y - x)) \leq f(x) + \lambda(f(y) - f(x)) \quad \text{for all } x, y \in \operatorname{dom} f \text{ and } \lambda \in [0, 1]. \tag{2}$$

Hint: Only one variable in (2) can vary freely; use it and the definition of a derivative to introduce f' to show that (2) implies (1). The other direction requires a different approach to get rid of the f' term.

Homework 2

1.3 Strict and strong convexity [6 pts]

(a) Show that all m-strongly convex functions are also n-strongly convex for $m > n$.

(b) Prove that all strongly convex functions are strictly convex.

(c) Give a function $f(x)$ that is strictly convex that is not strongly convex for any $m > 0$. (Don't just write down a function; prove that it is strictly convex and not strongly convex.)

1.4 Examples [6 pts]

For each of the following functions, show whether it is strongly convex, strictly convex, convex, or not convex.

(a) $x^2 + x^4$ on $\mathbb{R}$ (b) $x^2 + x^4$ on $[1, 5]$ (c) Any norm $\|x\|$ on $\mathbb{R}^n$

Recall that a norm is any function $\mathbb{R}^n \to \mathbb{R}$ satisfying the following for any scalar $a \in \mathbb{R}$ and vectors $x, y \in \mathbb{R}^n$: $\|ax\| = |a| \, \|x\|$, $\|x + y\| \leq \|x\| + \|y\|$, and $\|x\| = 0$ iff $x = 0$.

2 Linear Regression, Again? [Ahmed; 25 pts]

For this question, X denotes an $n \times d$ matrix whose rows are training points, y denotes an $n \times 1$ vector of corresponding output values, β denotes a $d \times 1$ parameter vector and β^* denotes the optimal parameter vector.

2.1 Why Lasso Works [12 pts]

Lasso is a form of regularized linear regression, where the L1 norm of the parameter vector is penalized. It is used in an attempt to get a sparse parameter vector where features of little "importance" are assigned zero weight. But why does lasso encourage sparse parameters? In this part you will work out the math to explain this behavior. To make analysis easier we will consider the special case where the training data is *whitened* (i.e. $X^T X = I$). For lasso regression, the optimal parameter vector is given by

$$\beta^* = \operatorname*{argmin}_{\beta} J_\lambda(\beta) = \operatorname*{argmin}_{\beta} \frac{1}{2}\|y - X\beta\|^2 + \lambda|\beta|_1,$$

where $\lambda > 0$.

(a) Show that whitening the training data nicely decouples the features, making β_j^* determined by the j^{th} feature and the output regardless of other features. To show this, write $J_\lambda(\beta)$ in the form

$$J_\lambda(\beta) = g(y) + \sum_{i=1}^{d} f(X_{\cdot i}, y, \beta_i, \lambda),$$

where $X_{\cdot i}$ is the i^{th} column of X.

(b) Assume that $\beta_j^* > 0$, what is the value of β_j^* in this case?

(c) Assume that $\beta_j^* < 0$, what is the value of β_j^* in this case?

(d) From (b) and (c) , what is the condition for β_j^* to be 0 ? How can you interpret that condition?

(e) Now consider ridge regression where the regularization term is replaced by $\frac{1}{2}\lambda\|\beta\|_2^2$. What is the condition for $\beta_j^* = 0$? How does it differ from the condition you obtained in (d) (why is β_j^* more likely to be 0 in the lasso case)?

2.2 Kernel Ridge Regression [13 pts]

[Note: In this part, training data is not necessarily whitened.]

In this part you will derive a kernelized version of ridge regression. This means we need to formulate training and prediction such that data points appear only in inner products. By replacing the inner product with the appropriate kernel, we can model non-linear functions or model regression over objects that are not natively in a vector space (e.g. strings). Recall that for ridge regression the optimal weight vector is given by

$$\beta^* = (X^T X + \lambda I)^{-1} X^T y$$

(a) Show that β^* is in the row space of X. That means β is in the space spanned by training points.

(b) Based on (a) , we can write β as a weighted combination of training data points

$$\beta = \sum_{i=1}^{n} \alpha_i x_i,$$

where x_i denotes the i^{th} column of X^T. The training problem now becomes finding the optimal α. Show that

$$\alpha^* = (X X^T + \lambda I)^{-1} y$$

and hence show that computing α^* requires only inner products between training data points.

(c) Show that predicting $\hat{f}(x)$ for a new data point x requires only inner products between x and training data points.

(d) If we are to store each real value as a floating point number, how many floating point numbers are needed to store the trained model in the non-kernelized version and the kernelized version ?

3 Kernels and Features [Carlton; 50 pts + 5 Bonus]

This is a coding question. In this question you will implement *Perceptron Classification*, and *Kernel Perceptron Classification* in Octave.

For each sub question you will be given a single function signature, and asked to write a single octave function which satisfies the signature.

This problem is Autograded 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.

- **Data** All questions will use the following data structures: XTrain $\in \mathbb{R}^{n \times f}$ is a matrix of training data, where each row is a training point, and each column is a feature. XTest $\in \mathbb{R}^{m \times f}$ is a matrix of (hidden) test data, where each row is a test point, and each column is a feature. yTrain $\in \{1, ..., c\}^{n \times 1}$ is a vector of training labels. yTest $\in \{1, ..., c\}^{m \times 1}$ is a (hidden) vector of test labels.

Homework 2

- **Submission Instructions** 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.
- **CHECKLIST: You MUST do all of the following in order for your submission to be graded.**
  - MUST execute on our machines in less than 20 minutes.
  - MUST be smaller than 100K
  - MUST be a .tar file
  - MUST return matrices of the exact dimension specified.

3.1 Perceptrons [18 pts]

In this question you will implement the perceptron classification algorithm using two different weight-learning algorithms. The idea is that we will use the training data to learn a set of weights w which can be used to predict the class of unknown examples.

(a) **Perceptron-Algorithm Weights [6 pts]**
Complete the function weightsP(XTrain, yTrain, nIter) which takes as input the training data XTrain, the training labels yTrain, and the number of iterations nIter and returns a $f \times 1$ weight vector w. weightsP() should implement the iterative perceptron weight learning algorithm described in class.

(b) **Least Squares Weights [6 pts]**
Complete the function weightsL(XTrain, yTrain) which takes as input the training data XTrain, and the training labels yTrain, and returns a $f \times 1$ weight vector w. weightsP() should implement the least squares weight learning algorithm: $w = argmin_w ||\mathsf{XTrain} \times w - \mathsf{yTrain}||^2$.

(c) **Perceptron Classifier [6 pts]**
Complete the function perceptronClassify(XTest, w) which takes as input the test data, and the weight vector, and returns a $n \times 1$ vector of class predictions $p \in \{1, ..., c\}^m$. Note that p_i is the predicted class for the ith row of XTest. (Note: For this question, make sure you go through the test examples in order, NOT in randomized order as we discussed in class.)

3.2 Kernel Perceptrons [17 pts]

In this question you will implement the kernel perceptron classification algorithm.

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. for all 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 $\phi(x_1), ..., \phi(x_n)$ (i.e. $w = \sum_i \alpha_i \phi(x_i)$ for some $\alpha_1, ..., \alpha_n$. Therefore $\langle \phi(x), w \rangle = \sum_i \alpha_i \langle \phi(x), \phi(x_i) \rangle$. Therefore all we really need to do is calculate the inner products between points, and keep track of the α values.

Note that in this question we use a nth degree polynomial kernel with inner product $\langle \phi(x_i), \phi(x_j) \rangle = \langle x_i, x_j \rangle^n$.

(a) **Kernel Weights [9 pts]**
Complete the function kernelWeights(XTrain, yTrain, nIter, d) which takes as input the training data XTrain, the training labels yTrain, the number of iterations to run for nIter, and the degree of the polynomial kernel d. kernelWeights() should output an $n \times 1$ weight vector w. Note that w is equivalent to the α values in the above discussion; in other words, we can reconstruct our weight vector in feature space as $\sum_i w_i \times \text{XTrain}_i$.

(b) **Kernel Perceptron Classifier [8 pts]**
Complete the function kernelPerceptronClassify(XTrain, XTest, w, d) which takes as input the training data XTrain, the test data XTest, the weight vector w, and the degree of the polynomial kernel d. kernelPerceptronClassify() should output a $n \times 1$ vector of class predictions p. Note that p_i is the predicted class for the ith row of XTest.

3.3 CHALLENGE: SPAM Feature Engineering [15 pts + 5 bonus]

Welcome to the second 10-701 Challenge Question! Once again this problem is much, much more difficult than the previous questions. Once again this is a competition (view the class leaderboard on the autolab website) and bonus points will be awarded to students who have top 10 classification accuracy on the class leaderboard. I was really impressed with how well you guys did on the first challenge question, and I look forward to seeing what you can do with this one.

- In this question you will engineer a list of features for SPAM email classification. The idea is to convert an email into a list of real numbers, which can then be used train a machine learning classification algorithm. The performance of the resulting classifier will depend ***heavily*** on how good your features are, so select them carefully!
- Specifically, your task is to complete the function mail2Feat(Mail) which converts a set of emails into a matrix of real numbers. Suppose we have e emails, where the ith email has w_i words. Mail is a $e \times 1$ cell array of emails, each of which is $w_i \times 1$ cell array of words. You can produce as many output features as you like; the only caveat is that all emails must have the same set of features. If you decide to output f features, then mail2Feat() should output a $e \times f$ matrix of real numbers.
- In addition to the mail2Feat() function you may also use the $dictionary.csv$ file. This file allows you to include arbitrary data (such as a dictionary of words) with your submission. Use this file to store and retrieve preprocessed results in your submission. An example of how to use this file is included in the mail2Feat.m template file.
- Your will be graded based on how well your features support classification. Specifically, your mail2Feat() function will be used to convert two hidden sets of emails into features. We will then train a perceptron using one set of emails, and test the performance of the perceptron on another. Your grade will be directly proportional to the classification accuracy obtained.

HINTS: I have included code in the mail2Feat.m template file which demonstrates an incredibly basic feature map. This code simply takes a few words and counts how many times each word appears in each email. These counts are the features. For more advanced (and successful) feature maps, TF-IDF is a good place to start. (https://en.wikipedia.org/wiki/Tf-idf)