Machine Learning 10-701 Mar 19, 2015
http://alex.smola.org/teaching/10-701-15/ due on Mar 30, 2015
Carnegie Mellon University

Homework 8

START HERE: Instructions

- The homework is due at 11:59pm on Mar 30th, 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 Mar 19, 2015
http://alex.smola.org/teaching/10-701-15/ due on Mar 30, 2015
Carnegie Mellon University

Homework 8

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.
- All labels are -1 or +1.

1 Loss Functions and Regularization Functions [50 pts]

Empirical Risk Minimization is a principle in statistical machine learning. Basically we can train our models by minimizing the empirical risk (training error). Linear regression and SVM are two examples we've learned in class. The objective function can be written as $\operatorname{argmin}_w \sum_i l(y_i, h_w(x_i)) + \lambda R(w)$, where $l(\cdot)$ is the loss function, $R(\cdot)$ is the regularization function, and $h_w(x_i) = x_i^T w$ is the hypothesis function. In this assignment you are asked to implement stochastic gradient descent (SGD) with different loss functions and regularization functions.

The autograder will not test the gradient where the objective function is not differentiable. However, you may refer to subgradient method or proximal gradient method (or simply set them to zero in this assignment) for implementation.

Some loss functions are listed as follows:

- L1 loss: $l(y, h_w(x)) = |y - h_w(x)|$
- L2 loss: $l(y, h_w(x)) = \frac{1}{2}\|y - h_w(x)\|_2^2$
- Logistic loss: $l(y, h_w(x)) = \ln(1 + e^{-y \cdot h_w(x)})$
- Hinge loss: $l(y, h_w(x)) = \max\{1 - y \cdot h_w(x), 0\}$

L1 and L2 loss are usually used for regression tasks. You may find out by yourself why it works poorly in classification tasks.

Logistic loss is actually the loss function for binary logistic regression, in which we use the logistic function to compute the probability of a data point, $p = \frac{1}{1+e^{-y_i w^T x_i}}$ for $y_i \in \{-1, 1\}$. You may find different forms for $y_i \in \{0, 1\}$. The loss function is the negative log-likelihood of the logistic function. Softmax regression (a.k.a. multinomial logistic regression, multi-class logistic regression, etc.) is the generalized logistic regression model for multiple classes. It uses softmax function to compute the probability for all classes. Softmax regression is pretty useful in practice although you are not required to implement it in this assignment.

Hinge loss is the loss function for soft margin support vector machine. Recall the objective function is $\min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$ with respect to $y_i(w^T x_i) \geq 1 - \xi_i, \xi_i \geq 0$. So that we can re-write ξ_i as $\max\{1 - y_i(w^T x_i), 0\}$.

- Compute the gradients for loss functions
  Complete the function $loss\_grad(X, y, w, loss\_type)$, which returns $grad \in \mathcal{R}^{f+1}$, the gradient of w. $X \in \mathcal{R}^{1\times(f+1)}$ denotes one padded training instance, $y \in \{-1, +1\}$ denotes one label, $w \in \mathcal{R}^{f+1}$ denotes the weight vector, and $loss\_type$ is a string variable which denotes the type of the loss function. You are asked to implement all the loss functions list above. You can also implement your own loss function for the leaderboard problem.

- Compute the gradients for regularization functions
  Complete the function $reg\_grad(w, reg\_type)$, which returns $grad \in \mathcal{R}^{f+1}$, the gradient of w. $w \in \mathcal{R}^{f+1}$ denotes the weight vector, and reg_type is a string variable which denotes the type of the regularization function. You are only required to implement L2 regularization($R(w) = \frac{1}{2}\|w\|^2$). You can implement your own regularization function for the leaderboard problem.

2 Empirical Feature Map [20 pts]

A kernel function can be written as $k(x_i, x_j) = \Phi(x_i)\Phi(x_j)$. Sometimes it is very hard to directly work with $\Phi(x)$ especially when it has infinite dimensions. Empirical Feature Map is a method to approximate $\Phi(x)$. Let $X \in \mathcal{R}^{n\times f}$ be our training data set. We denote the kernel matrix as $K \in \mathcal{R}^{n\times n}$, where $K_{ij} = \Phi(x_i)\Phi(x_j) = k(x_i, x_j)$. We map an arbitrary data point x with the following rule:

$$x \to \Phi(x) = (k(x, x_1), k(x, x_2), \ldots, k(x, x_n))^T$$

Where x_i is a training data point. (This is what you should implement.)

A brief proof (do not implement this) is shown as follows:

To reconstruct the original kernel matrix, we can explicitly write out the feature map as $\Phi^e(X) = K^{-\frac{1}{2}}K$, so that $\Phi^e(X)^T\Phi^e(X) = KK^{-1}K = K$.

- Empirical Feature Map [20 pts]
  Complete the function $feature\_map(X, XTrain, kernel\_type)$ which returns $phi \in \mathcal{R}^{1\times nTrain}$, the instance in empirical feature space. $X \in \mathcal{R}^{1\times(f+1)}$ denotes one instance, $XTrain \in \mathcal{R}^{nTrain\times(f+1)}$ denotes the training data matrix. $kernel\_type$ denotes the type of kernel. You are only required to implement linear kernel $K(x_i, x_j) = x_i^T x_j$ and Gaussian RBF kernel where $K(x_i, x_j) = \exp(-0.5\|x_i - x_j\|^2)$. You can implement other kernels for the leaderboard problem.

3 Stochastic Gradient Descent [10 pts]

Very similar to gradient descent, the weight vector is updated every iteration of SGD. In each iteration, one training instance is used for computing the gradient.

$$w \leftarrow w - lr * (\frac{\partial l(w)}{\partial w} + \lambda\frac{\partial R(w)}{\partial w})$$

There are two ways for picking the training instance, one is cyclically from the first training instance to the last training instance, the other one is using random order. The stopping criterion for SGD may be pretty tricky since it might "bounce" around when it approaches the optimal solution. You can design your own stopping criterion but you have to terminate SGD after it reaches maximum number of iterations. In this problem, you are asked to implement cyclic order with w initialized to a zero vector.

- Cyclic SGD [10 pts]
  Complete the function $SGD(\ldots)$ which returns $w \in \mathcal{R}^{f+1}$, the learned weights from stochastic gradient descent. Here are the variables of this function: $XTrain \in \mathcal{R}^{nTrain\times(f+1)}$ is the padded training set. $yTrain \in \mathcal{R}^{nTrain}$ is the training labels. $loss\_type$, reg_type are the types of loss function and regularization function correspondingly. sgd_type denotes how SGD is performed, lr is the learning rate, $lambda$ is the coefficient multiplied to the regularization function, $maxIter$ denotes the maximum number of iterations of SGD. You can call two functions you implemented in part 1.

4 Leaderboard Problem [10+10 pts]

Complete the function $leaderboard(XTrain, yTrain, XTest)$ which returns $y \in \mathcal{R}^{nTest \times 1}$, a vector of labels in $-1, +1$. Cats are denoted as -1 and dogs are $+1$. The features are the HoG features of dogs and cats images. As usual, students achieve baseline accuracy (70%) can get 10 points. Students ranking top 10 can get extra 10 points. If you use empirical feature map, $f + 1$ dimension goes to $nTrain$ dimension. You can call all functions you previously implemented.

Some hints:

- Mini-batch or batch gradient descent converge faster than SGD. Vectorize your code in problem 1 and 2. Assume $X \in \mathcal{R}^{n \times (f+1)}$ in those problems.
- Try to mitigate overfitting. Other than using regularization, try some simple tricks such as early stopping.