Machine Learning 10-701 Feb 2, 2015
http://alex.smola.org/teaching/10-701-15/ due on Feb 16, 2015
Carnegie Mellon University

Homework 4

START HERE: Instructions

- The homework is due at 11:59pm on Feb 16, 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 2, 2015
http://alex.smola.org/teaching/10-701-15/ due on Feb 16, 2015
Carnegie Mellon University

Homework 4

---

1 Kernels [40 pts]

In this problem you are asked to implement kernels. Radial Basis Function (Gaussian) Kernel and Boxcar Kernel are mandatory. You can also implement other kernels for the leaderboard problem. The RBF kernel and Boxcar kernel are computed as follows:

- RBF kernel:
  $h(x) = (2\pi)^{-\frac{f}{2}} e^{\frac{-x^T x}{2}}$
- Boxcar kernel:
  $h(x) = c * \mathbf{1}\{\|x\|_2 \leq 1\}$. $\mathbf{1}\{cond\}$ is an indicator function which equals to 1 if the condition is satisfied, 0 if not satisfied. c is a normalizing constant that makes sure the kernel sums up to 1. $c = 0.5$ in one dimensional case, $c = \frac{1}{\pi}$ in two dimensional case. You are only required to implement these two cases. For n dimensional cases, please refer to n-sphere.

$x \in \mathcal{R}^f$. The kernel function is defined as $K(x_1, x_2) = \frac{1}{r^f} h(\frac{x_1 - x_2}{r})$, where r denotes the bandwidth. Please be aware that smoothing kernels are usually positive, symmetric, and sum to 1.

- Compute kernel [20 pts for each kernel]
  Complete the function $compute\_kernel(XTrain, XTest, bw, kernel\_type)$ which returns a $nTrain \times nTest$ matrix K. bw is the bandwidth. $kernel\_type$ is a string variable (e.g. 'rbf', 'boxcar') denoting the type of the kernel. K is the matrix where $K_{i,j}$ denotes the kernel for i-th training data and j-th test data.

2 Kernel Density Estimation [20 pts]

In this problem you are asked to implement kernel density estimation. Recall that the probability density function is calculate as: $p(x) = \frac{\sum_{j=1}^{n} K(X_j, x)}{n}$, where $x \in \mathcal{R}^f$ denotes a test sample, X_j denotes jth training sample, n denotes the total number of training data, and h denotes the bandwidth.

- Kernel Density Estimation
  Complete the function $KDE(XTrain, XTest, bw, kernel\_type)$ which returns a $\mathcal{R}^{nTest}$ vector p. p_i denotes the estimated density at ith test data point. You should call the function you implemented in the first problem.

3 Kernel Regression [30+10 pts]

In this problem you are asked to implement Nadaraya-Watson kernel regression. You should call the function you implemented in the first problem.

- Kernel Regression [20 pts]
  Complete the function $kernel\_regression(XTrain, yTrain, XTest, bw, kernel\_type)$ which returns a $\mathcal{R}^{nTest}$ vector y. y_i denotes the predicted value for ith test data point.
- Leaderboard [10+10 pts]
  Complete the function $regression(XTrain, yTrain, XTest)$, which is a wrapper for kernel_regression. Train your model on the dataset we provide (see 'data.mat' in Data folder), choose the best bandwidth and kernel to get high score on a hidden test set. You can find some useful kernel functions here.

Please be aware of the bias-variance trade off in homework 2. Your regressor will be evaluated with R-squared value. Students with a regressor above baseline (0.25) can get 10 points. Bonus points will be awarded to students ranking top 10 on the class leaderboard.

4 Hints for the Leaderboard Problem

- Normalize features before building the regressor.
- Think about some ways to handle outliers in test set.
- Implement data pre-processing in regression.m and your own kernels in compute_kernel.m.