CS74/174, Winter 2015, Problem Set # 1

January 21, 2015

Due: February 5th 2015, 11:59pm

This problem set requires you to provide written answers to a few technical questions. In addition, you will need to implement some learning algorithms in MATLAB and apply them to the enclosed data sets. The questions requiring a written answer are denoted with **W**, while the MATLAB programming questions are marked with **M**.

You must provide your written answers in paper form and leave them in the course mailbox located near the lobby of the Sudikoff building (the mailbox is tagged with the label CS74/174). Since each problem will be graded by a different person, we ask that you provide a separate submission for each individual problem. For example, for problem 4 of this homework assignment you should turn in a single set of pages stapled together containing your answers to questions 4(a), 4(b), 4(e) and 4(f). **Please use the cover pages included at the end of this document. For each problem, fill out the corresponding cover page by writing your name, date and time of submission and use it as front page for your problem submission. A penalty of 5 points will be applied if you fail to follow these instructions.**

Instead, your MATLAB code must be submitted via Canvas. We have provided MATLAB function files with an empty body for all the functions that you need to implement. Each file specifies in detail the input and the output arguments of the function. A few of the functions include some commands to help you get started with the implementation. We have also provided scripts that check the sizes of the inputs and outputs of your functions. Do not modify these scripts. In order to allow us to test your code, it is imperative that you do not rename the files or modify the syntax of the functions. Make sure to include all the files necessary to run your code. **If we are unable to run one of your functions, you will receive 0 points for that question.** Please place all your files (data, .fig files automatically generated by the provided scripts, and function files) in a single folder (**without subfolders**). Zip up the entire folder, and upload the compressed file to Canvas.

Please do not use built-in functions from Matlab toolboxes (e.g., the Statistics Toolbox) for your homework code. If in doubt, please ask.

1. *Problem for graduate credit only: if you are enrolled in COSC74 (instead of COSC174) you do not need to solve this problem.*
   [**W, 6 points**] Consider one-dimensional random variables x and y. Show that if x and y are independent, their covariance is zero, i.e., $\mathbf{cov}(x, y) = 0$.

2. [**W, 6 points**] Suppose we have three boxes: r (red), g (green), and b (blue). Box r contains 3 apples, 4 oranges, and 3 limes; box g contains 3 apples, 3 oranges, and 4 limes; box b

contains 1 apple, 2 oranges, and 0 limes. If as box is selected at random with probabilities $p(r) = 0.2, p(g) = 0.5, p(b) = 0.3$, and a piece of fruit is picked from the box (with equal probability of choosing any of the items in the box), then what is the probability of selecting an apple? If we observe that the selected fruit is an orange, what is the probability that it came from the green box?

3. **[W, 12 points]** You are on a game show. The game host asks you to choose one of three doors, #1, #2, or #3. Behind one door there is a car. Behind the other two doors there is a zonk (i.e., a worthless consolation prize). You obviously want to win the car. You choose a door, say #1 (without loss of generality). Then the host (who knows what's behind each door), opens one of the other two doors **not** containing the car, say #3. At this point the host asks you whether you want to stick with your initial choice (i.e., #1), or choose the other unopened door (i.e., #2). What do you decide to do and why?

   You should provide a formal justification of your answer using random variable $C \in \{\#1, \#2, \#3\}$ to indicate the actual position of the car, so that $P(C = \#1)$ denotes the probability that the car is behind door #1. Similarly, use random variable $H \in \{\#1, \#2, \#3\}$ to denote the door opened by the host.

4. **[17 points]** Let's flip a coin. Let $c \in \{0, 1\}$ be a random variable indicating the result of the flip (1 for heads, 0 for tails). The probability that the coin lands heads on any trial is given by a parameter μ. Note that we can write the distribution of c as: $P(c\,; \mu) = \mu^c(1-\mu)^{1-c}$. We flip the coin m times, and denote the result of the i-th flip by variable $c^{(i)}$. We assume that the coin flips are *independent*. We observe heads H times.

   (a) [**W**, 2 points] Write the likelihood function, i.e., the probability of the data $\mathcal{D} = \{c^{(1)}, ..., c^{(m)}\}$ given the model described above. Keep in mind that the likelihood is the probability of the observed training set $\mathcal{D}$ (rather than all possible training sets containing H heads in m examples). Thus your likelihood function should *not* include a combinatorial term counting the number of different ways H heads could occur in m trials.

   (b) [**W**, 5 points] Derive the parameter μ using Maximum Likelihood estimation (hint: maximize the log likelihood).

   (c) [**M**, 1 point] Code the MATLAB function `q4_likelihood.m` to compute the likelihood given scalar input arguments H, m and an input vector μ (the function should return a vector of likelihood values as big as μ). You can now inspect how the likelihood function differs for the three cases $\{m = 1, H = 1\}$, $\{m = 100, H = 100\}$, and $\{m = 100, H = 80\}$ by executing the script `q4c.m`, which calls your function `q4_likelihood.m`.

   Let us now assume that we have good reasons to believe that the coin is not counterfeit. However, we are not certain. We model this prior knowledge in the form of a prior distribution over μ:

$$p(\mu; a) = \frac{1}{Z}\mu^{a-1}(1-\mu)^{a-1} \tag{1}$$

   where a is a parameter governing the prior distribution and Z is a normalization constant (so that $\int_0^1 p(\mu; a)d\mu = 1$).

(d) [**M**, 1 point] Implement the prior function by coding the file `q4_prior.m`. Then, execute the script `q4d.m`, which plots the prior for $a = 2$ ($Z = 1/6$), and $a = 10$ ($Z = 1/923780$).

(e) [**W**, 5 points] Assuming the prior $p(\mu; a)$ in Eq. 1, derive the analytical expression of parameter μ using Maximum A Posteriori (MAP) estimation (hint: maximize the log posterior and drop the term related to the evidence, i.e., the denominator in Bayes' rule).

(f) [**W**, 2 points] By looking at the MAP estimate derived in the previous point, can you provide an interpretation of parameter a in terms of training examples?

(g) [**M**, 1 points] Implement the posterior by coding the file `q4_posterior.m`, and plot the posterior for the same coin flipping results considered above (i.e., $\{m = 1, H = 1\}$, $\{m = 100, H = 100\}$, and $\{m = 100, H = 80\}$), by running the script `q4g.m`. Do not include the evidence term (i.e., the denominator) in the posterior calculation. There is no way to estimate the evidence and in any case it is just a constant.

5. [**30 points**] Write MATLAB code implementing a regression algorithm for multi-dimensional inputs x and 1D outputs y. Your software must learn the regression hypothesis by minimizing the regularized least-square objective

$$E(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left[y^{(i)} - \theta^\top b(x^{(i)})\right]^2 + \frac{\lambda}{2}\sum_{j=1}^{d}\theta_j^2 \tag{2}$$

where $\theta = [\theta_0, \theta_1, ..., \theta_d]^\top$ and $b(x)$ is a vector that encodes either of the following two distinct choices of features:
$b^l(x) = [1, x_1, ..., x_d]^\top$ (where, d is the number of input entries in vector x and x_j denotes the j-th element of vector x), or
$b^q(x) = [1, x_1, ..., x_d, x_1^2, x_1x_2, ..., x_1x_d, x_2^2, x_2x_3, ..., x_2x_d, ..., x_d^2]^\top$ (i.e., a quadratic function of the elements of x).

Please note that the regularization term in Eq.2 does not include the bias term θ_0 (i.e., the parameter θ_0 is *not* encouraged to stay close to zero). The justification for this is that θ_0 is not associated to a real feature and thus is not affected by data noise as much as the others, so it is beneficial to let it vary without penalty. The closed-form solution optimizing this variant of the least-squares regularized objective is obtained by solving the following system of linear equations:

$$(B^\top B + \lambda U)\theta = B^\top y \tag{3}$$

where U is a diagonal matrix having value 0 in position $(1, 1)$ and value 1 in the other diagonal entries, $B = \left[b(x^{(1)}), ..., b(x^{(m)})\right]^\top$, and $y = \left[y^{(1)}, ..., y^{(m)}\right]^\top$.

You will use these models to predict the MPG (miles per gallon) of a car from several attributes of the vehicle. The data set is contained in the enclosed MATLAB file *autompg.mat* [1]. You can load the data in MATLAB by typing "load('autompg.mat');": this command will load four variables in the workspace: *trainsetX*, *trainsetY*, *testsetX*, and *testsetY*. Each row in these matrices corresponds to one example: *trainsetX(i,:)* contains the input vector of

[1]These examples were derived from the Auto-Mpg data set available at the UC Irvine Machine Learning Data Repository.

training example i, and *trainsetY(i)* the corresponding output value. Look at *autompg.names* for a description of the features. You should train your algorithm on the training set (*trainsetX*, *trainsetY*), and evaluate its performance on the test set (*testsetX*, *testsetY*). For all experiments, you should measure the performance as mean squared error on the test set, i.e., evaluate it by computing $\frac{1}{M}\sum_{i=1}^{M}\left[\hat{y}^{(i)} - f_\theta(\hat{x}^{(i)})\right]^2$ where $(\hat{x}^{(i)}, \hat{y}^{(i)})_{i=1,..,M}$ are the M test examples, and f_θ is the function learned on the training set.

(a) [**M**, 12 points] Implement the Matlab functions described as follows:

- [4 points] `q5_features.m` computes the linear and quadratic features $b^l(x)$ and $b^q(x)$.
- [2 point] `q5_mse.m` calculates the mean squared error.
- [5 points] `q5_train.m` learns model parameters θ given a training set of examples by solving the system in Eq. 3 (use the MATLAB command `'\'` to solve the system).
- [1 points] `q5_predict.m` performs prediction by evaluating a given learned model for the input examples.

(b) [**M**, 6 points] The performance of the algorithm will depend on the choice of the parameter λ. Implement the function `q5_cross_validation_error.m` to perform N-fold cross-validation. Given a training set of examples and a finite set of values for hyperparameter λ, this function should return the cross-validation score, i.e., the average of the mean squared errors over the validation sets. Once you have coded this function, you can run the script `q5b.m` to plot the 10-fold cross-validation scores for $\lambda \in \{10^{-5}, 10^{-3}, 10^{-1}, 10, 10^3, 10^5, 10^7\}$ with both the linear model and the quadratic model.

(c) [**W**, 3 points] Are there any values of λ producing underfitting? If yes, which values?

(d) [**W**, 3 points] Are there any values of λ producing overfitting? If yes, which values?

(e) [**W**, 3 points] Which of the two versions of feature vector $b(x)$ produces more overfitting, $b^l(x)$ or $b^q(x)$? Can you explain why?

(f) [**M**, 2 points] Code the function `q5_test_error.m` which trains a model on the training set and returns the test error. Then, execute the script `q5f.m`, which plots the *test set* mean squared error for the same set of values of λ as before.

(g) [**W**, 1 point] Is the cross-validation score a good predictor of performance on the test set? Please comment on why or why not.

6. **[29 points]** For this problem you need to implement the locally weighted linear regression (LWLR) and to apply it to the MPG regression problem above. LWLR predicts the output of test example x as $f_\theta(x) = \theta(x)^\top b^l(x)$, where $\theta(x)$ is given by:

$$\theta(x) = \arg\min_\theta E^{LWLR}(\theta) \tag{4}$$

with $E^{LWLR}(\theta) = \frac{1}{2}\sum_{i=1}^{m} w^i(x)\left[y^{(i)} - \theta^\top b^l(x^{(i)})\right]^2$ and $w^i(x) = \exp\left(-\frac{||x^{(i)}-x||^2}{2\tau^2}\right)$.

(a) [**W**, 8 points] Show that the locally weighted linear regression error $E^{LWLR}(\theta)$ can be written in matrix notation as

$$E^{LWLR}(\theta) = (B\theta - \mathbf{y})^\top W(B\theta - \mathbf{y}) \tag{5}$$

where B and $\mathbf{y}$ are defined as in the slides used in class, and W is a matrix whose entries are written as functions of the weights $w^i(x)$. Specify W in full detail and remember to include the scaling factor 1/2.

(b) [**M**, 12 points] It is easy to show (you may want to do it as an exercise) that the minimizer $\theta(x)$ of Eq. 5 can be computed via the following closed-form solution:

$$\theta(x) = (B^\top W B)^{-1} B^\top W \mathbf{y} \tag{6}$$

Use this closed-form solution to implement LWLR. Your task is to fill the body of the following functions that we have provided to you:

- [4 points] `q6_W.m` constructs the matrix W as specified in your answer to question 6(a). In order to avoid numerical problems, before computing W, scale the weights $w^i(x)$ so that they sum up to 1, i.e., normalize them so that $\sum_{i=1}^m w^i(x) = 1$.
- [4 points] `q6_train.m` trains the model for a given test example x via the closed-form solution of Eq. 6 (use the MATLAB command `'\'` to compute the solution). This function should call `q6_W.m` from the previous point to construct W. Note that the matrix B can be computed using the function `q5_features.m` that you have implemented in problem 5, using argument `mode` set to `'linear'`.
- [2 points] `q6_predict.m` predicts the output value for an input test example x. This function should invoke `q6_train.m` to learn the parameter $\theta(x)$ specific to the input test example.
- [2 points] `q6_test_error.m` performs full evaluation of LWLR on the entire test set and returns the test Mean Squared Error. Note that you can reuse your function `q5_mse.m` from problem 5 to compute the error.

After you are done with all these implementations, you can run the enclosed script `q6b.m`, which uses your functions to plot the test set error of LWLR for $\tau \in \{10^2, 10^3, 10^5, 10^6\}$.

(c) [**M**, 4 points] Now implement `q6_cross_validation_error.m` to perform N-fold cross-validation given a training set of examples and a set of values for hyperparameter τ. For each value of τ, this function should return the cross-validation error, i.e., the average of the mean squared errors over the validation sets. Again, reuse the function `q5_mse.m` to calculate the mean squared error. Note that in each validation run, you will need to evaluate m distinct functions $f_{\theta(\hat{x}^{(i)})}$, as the function itself varies with the test input $\hat{x}^{(i)}$. Run the script `q6c.m` to plot the cross validation error for $\tau \in \{10^2, 10^3, 10^5, 10^6\}$.

(d) [**W**, 2 points] Using the plot of the cross validation error, please identify the values of τ (if any) causing underfitting.

(e) [**W**, 2 points] Which values of τ yield overfitting?

(f) [**W**, 1 points] Now look at the plot of the test error. Is the performance similar to the one measured via cross validation (cross validation error)? Why or why not?

HW1 | CS74/174, Winter 2015

Problem 1

First name ______________________ **Last name** ______________________

Date and time of submission ____

HW1 | CS74/174, Winter 2015

Problem 2

First name ______________________ **Last name** ______________________

Date and time of submission ______________________________________

HW1 | CS74/174, Winter 2015

Problem 3

First name		**Last name**	

Date and time of submission ____________________

HW1 | CS74/174, Winter 2015

Problem 4

First name ______________________ **Last name** ______________________

Date and time of submission ______________________________________

HW1 | CS74/174, Winter 2015

Problem 5

First name ______________________ **Last name** ______________________

Date and time of submission ______________________________________

HW1 | CS74/174, Winter 2015

Problem 6

First name ____________________ **Last name** ____________________

Date and time of submission ____