CSL 341: Assignment 1

Due Date: 11:50 pm, Sunday Sep 15, 2013. Total Points: 80

Notes:

- This assignment has a mix of theoretical as well as implementation questions.
- Only the implementation questions will be graded.
- You are strongly encouraged to try out theoretical questions though they are not graded.
- You should submit all your code as well as any graphs that you might plot. Do not submit answers to theoretical questions.
- For each problem that you implement, you should include a file <question-no.writeup.txt> which should have a brief explanation of what you did.
- Your code should have appropriate documentation for readability.
- You will be graded based on what you have submitted as well as your ability to explain your code.
- Refer to the course website for assignment submission instructions.
- This assignment is supposed to be done individually. You should carry out all the implementation by yourself.
- We plan to run Moss on the submissions. Any cheating will result in a zero on assignment and possibly even heavier penalties (including a fail grade).
- Many of the problems below have been adapted (or borrowed) from the Machine Learning course offered by Andrew Ng at Stanford University. One question has been adapted from the Machine Learning course offered by Oliver Schulte at Simon Fraser University.

1. **(15 points) Logistic Regression**
   Consider the log-likelihood function for logistic regression:

$$l(\theta) = \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))$$

   For the following, you will need to calculate the value of the Hessian H of the above function. You do not need to submit the derivation of the Hessian, but you should include the form in your write-up.

   (a) **(10 points)** The files "q1x.dat" and "q1y.dat" contain the inputs ($x^{(i)} \in R^2$) and outputs ($y^{(i)} \in \{0, 1\}$) respectively for a binary classification problem, with one training example per row. Implement[1] Newton's method for optimizing $l(\theta)$, and apply it to fit a logistic regression model to the data. Initialize Newton's method with $\theta = \vec{0}$ (the vector of all zeros). What are the coefficients θ resulting from your fit? (Remember to include the intercept term.)

[1] Write your own version, and do not call a built-in library function.

(b) **(5 points)** Plot the training data (your axes should be x_1 and x_2 , corresponding to the two coordinates of the inputs, and you should use a different symbol for each point plotted to indicate whether that example had label 1 or 0). Also plot on the same figure the decision boundary fit by logistic regression. (i.e., this should be a straight line showing the boundary separating the region where $h(x) > 0.5$ from where $h(x) \leq 0.5$.)

(c) **(Not Graded)** Show that for any vector z, it holds true that $z^T H z \leq 0$.

[Hint: You might want to start by showing the fact that $\sum_i \sum_j z_i x_i x_j z_j = (x^T z)^2 \geq 0$.]

Remark: This is one of the standard ways of showing that the matrix H is negative semi-definite, written "$H \leq 0$". This implies that l is concave, and has no local maxima other than the global one. If you have some other way of showing $H \leq 0$, you are also welcome to use your method instead of the one above.

2. **(14 points) Locally Weighted Linear Regression**
Consider a linear regression problem in which we want to "weigh" different training examples differently. Specifically, suppose we want to minimize

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} w^{(i)} (\theta^T x^{(i)} - y^{(i)})^2.$$

In class, we worked out what happens for the case where all the weights (the $w^{(i)}$'s) are the same. In this problem, we will generalize some of those ideas to the weighted setting and implement the locally weighted linear regression algorithm.

In the above setting, $J(\theta)$ can also be written

$$J(\theta) = (X\theta - \vec{y})^T W (X\theta - \vec{y})$$

for an appropriate diagonal matrix W, and where X and $\vec{y}$ are as defined in class.

If all the $w^{(i)}$'s equal 1, then we saw in class that the normal equation is

$$X^T X\theta = X^T \vec{y},$$

and that the value of θ that minimizes $J(\theta)$ is given by $(X^T X)^1 X^T \vec{y}$. By finding the derivative $\nabla_\theta J(\theta)$ and setting that to zero, generalize the normal equation to this weighted setting, and find the new value of θ that minimizes $J(\theta)$ in closed form as a function of X, W and $\vec{y}$.

As earlier, you do not need to submit the derivations for the above, but you should include the the form of W (in terms of $w(i)$'s) as well as the closed form solution for the parameters in your write-up.

The files "q2x.dat"and "q2y.dat"contain the inputs ($x^{(i)}$) and outputs and "q2y.dat"($y^{(i)}$), respectively, for a regression problem, with one training example per row.

(a) **(3 points)** Implement (unweighted) linear regression ($y = \theta^T x$) on this dataset (using the normal equations), and plot on the same figure the data and the straight line resulting from your fit. (Remember to include the intercept term.)

(b) **(8 points)** Implement locally weighted linear regression on this dataset (using the weighted normal equations you derived above), and plot on the same figure the data and the curve resulting from your fit. When evaluating $h_\theta(.)$ at a query point x, use weights

$$w^{(i)} = \exp\big(-\frac{(x - x^{(i)})^2}{2\tau^2}\big)$$

with a bandwidth parameter $\tau = 0.8$. (Again, remember to include the intercept term.)

(c) **(3 points)** Repeat (ii) four times, with $\tau = 0.1, 0.3, 2$ and 10. Comment **briefly** on what happens to the fit when τ is too small or too large.

(d) **(Not Graded)** Suppose we have a training set $\{(x^{(i)}, y^{(i)}); i = 1 \ldots, m\}$ of m independent examples, but in which the $y^{(i)}$'s were observed with differing variances. Specifically, suppose that

$$p(y^{(i)} \mid x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi}\sigma^{(i)}} \exp\big(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2(\sigma^{(i)})^2}\big)$$

i.e., $y^{(i)}$ has mean $\theta^T x^{(i)}$ and variance $(\sigma^{(i)})^2$ (where the $\sigma^{(i)}$'s are fixed, known, constants). Show that finding the maximum likelihood estimate of θ reduces to solving a weighted linear regression problem. State clearly what the $w^{(i)}$'s are in terms of the $\sigma^{(i)}$'s.

3. **(21 points) Linear Regression with Polynomial Basis Functions**
In the class, we looked at linear regression where the hypothesis function is given as $h_\theta = \theta^T x$. Here, the hypothesis is a linear function of the features as well as the parameters θ. In general, the relationship between the hypothesis and the input features may not be linear. In such cases, we can define a new basis function $\phi(x)$ which transforms the original feature space into a new feature space over which linear regression can be performed. The hypothesis can then defined as a linear function of θ defined over the new basis function i.e. $h_{\theta(x)} = \theta^T x$. For instance, if x is one-dimensional vector, and $\phi(x) = (1, x, x^2)$, then, $h_\theta(x) = \theta^T \phi(x)$, which can represent the hypothesis as a quadratic function over the original feature space.

Dataset for this problem is available at `http://archive.ics.uci.edu/ml/datasets/Auto+MPG`. We will ignore the last feature (car name) for this problem. The goal is to predict miles per gallon (mpg), given the values of remaining seven features. We will use the first 100 points as the training data and the remainder as the test data. You can ignore any examples with missing values for any of the features.

In this problem, we will implement linear regression with polynomial basis functions. Given the input feature vector $x = (x_1, x_2 \cdots x_7)$, define a polynomial basis function of degree d where we consider only the terms for each variable independently. In other words, we define our basis function as $\phi(x) = (x_0, x_1, {x_1}^2, {x_1}^3 \cdots {x_1}^d, x_2, {x_2}^2, {x_2}^3 \cdots {x_2}^d \cdots x_7, {x_7}^2, {x_7}^3 \cdots {x_7}^d$. Here, x_0 denotes the intercept term. So, in total there will be $1 + 7 * d$ parameters to be learnt.

(a) **(6 points)** Implement linear regression over the polynomial basis function defined above to learn a model for this problem. Vary the value of d from 1 to 10. Use stochastic gradient descent to learn the parameters. Plot the training as well as test errors with varying d. What do you observe?

(b) **(4 points)** In order to visualize the data well, use only one of the features (horsepower) and again perform linear regression over the polynomial basis function ($d \in \{1 \cdots 10\}$) defined over this single input feature. Make sure to include the intercept term. Plot the training data points, learned polynomial, and test data points. Include the plots for $d = 1, 2, 8$ in your output. What do you observe?

(c) **(6 points)** One of the ways to keep the weights from growing too large is to include a quadratic penalty term in the cost function. This is called L_2-regularized regression. In the L_2-regularization framework, the cost function can be define as:

$$J(\theta) = \frac{1}{2}\left(\sum_{i=1}^{m}(\theta^T \phi(x^{(i)}) - y^{(i)})^2 + \gamma(\theta^T \theta)\right)$$

Here, γ is the weight of the regularizer. Find the expression of the gradient for the above L_2-regularized formalism. Next, implement L_2-regularized regression. Using only one of the features (horsepower) as earlier, fit a degree 8 polynomial and find the parameters using stochastic gradient descent. Use $\gamma =$ { 0.01, 0.1, 1, 10, 100, 1000 }. Plot the training as well as test set accuracies for varying γ (plot γ on a log scale). What do you observe?

(d) (**5 points**) A principled way of finding the best value of the γ parameter is to use cross-validation. Read about cross-validation from the lecture notes. Find the best value of γ using 5-fold cross validation over the training data. For the best value of the γ obtained obtained using cross-validation, report the average validation set and test set accuracies. Plot training data points, learned degree 8 polynomial (for the best γ obtained using cross-validation) and test data points. Also, plot the learned degree 8 polynomial when $\gamma = 0$ on the same graph. Comment on your observations.

4. **(30 points) Spam classification**
In this problem, we will use perceptron and SVM training algorithms to build a spam classifier. The dataset we will be using is a subset of 2005 TREC Public Spam Corpus. It contains a training set and a test set. Both files use the same format: each line represents the space-delimited properties of an email, with the first one being the email ID, the second one being whether it is a spam or ham (non-spam), and the rest are words and their occurrence numbers in this email. The dataset presented to you is processed version of the original dataset where non-word characters have been removed and some basic feature selection has been done. [2] You can download the dataset from here.

[2] Some of the pre-processing was done for the Machine Learning course taught by Pedro Domingos at the University of Washington.

(a) **(6 points)** Implement the perceptron training algorithm to classify spam. Use your algorithms to learn from the training set and report the accuracy on the test set.

Note: You should implement the learning algorithm from the first principles and not use any existing Matlab modules.

(b) **(6 points)** Train an SVM on this dataset using the LibSVM library, available for download from LibSVM. Convert the data into the format required by LibSVM and train a linear SVM. You can use the default set of parameters that come with the package. Report the accuracies on the test set.

Note: You should submit any code that you wrote as a wrapper to get LibSVM to run on this dataset.

(c) **(4 points)** Vary the number of examples used in training from 1000 to 9000 at the interval of 1000. Plot on a graph the test set accuracies (y-axis) as you vary the number of training examples (x-axis) for the two algorithms. This is called a learning curve. What do you observe? Explain.

(d) **(2 points)** Intuitively, some tokens may be particularly indicative of an email being in a particular class. For both SVMs and perceptron, we can find out the most indicative tokens by looking at the relative weights assigned to the corresponding features. Find out the 5 most indicative words for classifying spam as well as not spam i.e. you should pick the 5 words with the most positive weight and 5 with the most negative weight. Repeat the same for SVM. Do you see any difference?

(e) **(4 points)** For SVMs, what is the number of support vectors output by your algorithm? How many training points lie on the wrong side of the margin (how would you identify these points)? Also, look at the content of some of the documents in each of the above cases. Comment on what you observe.

(f) **(2 points)** Now train an SVM using the RBF kernel provided in the package. Use the default settings of parameters. Report the test set accuracies.

(g) **(4 points)** Try different values of the cost parameter ($c \in \{.1, 1, 5, 10, 20, 50, 100\}$) to see if you can improve the results. Use five fold cross-validation over the training set to find the best value of the cost parameter. Report the test set accuracies using the best cost parameter value found in the previous step.

(h) **(2 points)** Which of the three algorithms gives best test set accuracies? Which runs faster? Comment.

Following problems are for your practice and will not be graded.

5. **(Not Graded) Poisson Regression**

(a) Consider the Poisson distribution parameterized by λ:

$$p(y; \lambda) = \frac{e^{-\lambda}\lambda^y}{y!}$$

Show that the Poisson distribution is in the exponential family, and clearly state what are $b(y), \eta, T(y)$, and $a(\eta)$.

(b) Consider performing regression using a GLM model with a Poisson response variable. What is the canonical response function for the family? (You may use the fact that a Poisson random variable with parameter λ has mean λ.)

(c) For a training set $\{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$, let the log-likelihood of an example be $\log p(y^{(i)} \mid x^{(i)}; \theta)$. By taking the derivative of the log-likelihood with respect to θ_j , derive the stochastic gradient ascent rule for learning using a GLM model with Poisson responses y and the canonical response function.

6. **(Not Graded) Gaussian Discriminant Analysis:**
Suppose we are given a dataset $\{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$ consisting of m independent examples, where $x^{(i)} \in R^n$ are n-dimensional vectors, and $y^{(i)} \in \{0, 1\}$. We will model the joint distribution of (x, y) according to:

$$p(y) = \phi^y(1-\phi)^{1-y}$$

$$p(x \mid y = 0) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\big(-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\big)$$

$$p(x \mid y = 1) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\big(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\big)$$

Here, the parameters of our model are ϕ, Σ, μ_0 and μ_1 . (Note that while there're two different mean vectors μ_0 and μ_1 , there's only one covariance matrix Σ.)

(a) Suppose we have already fit ϕ, Σ, μ_0 and μ_1, and now want to make a prediction at some new query point x. Show that the posterior distribution of the label at x takes the form of a logistic function, and can be written

$$p(y = 1|x; \phi, \Sigma, \mu_0, \mu_1) = \frac{1}{1 + \exp(-\theta^T x)},$$

where θ is some appropriate function of ϕ, Σ, μ_0, μ_1. (Note: To get your answer into the form above, for this part of the problem only, you may have to redefine the $x^{(i)}$'s to be $n + 1$-dimensional vectors by adding the extra coordinate $x_0^{(i)} = 1$, like we did in class.)

(b) For this part of the problem only, you may assume n (the dimension of x) is 1, so that $\Sigma = [\sigma^2]$ is just a real number, and likewise the determinant of Σ is given by $|\Sigma| = \sigma^2$. Given the dataset, we claim that the maximum likelihood estimates of the parameters are given by

$$\phi = \frac{1}{m} \sum_{i=1}^{m} 1\{y^{(i)} = 1\}$$

$$\mu_0 = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}}$$

$$\mu_0 = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}$$

$$\Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T$$

The log-likelihood of the data is

$$\begin{aligned} l(\phi, \mu_0, \mu_1, \Sigma) &= \log \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma) \\ &= \log \prod_{i=1}^{m} p(x^{(i)}|y^{(i)}; \mu_0, \mu_1, \Sigma) p(y^{(i)}; \phi) \end{aligned}$$

By maximizing l with respect to the four parameters, prove that the maximum likelihood estimates of ϕ, μ_0, μ_1 and Σ are indeed as given in the formulas above. (You may assume that there is at least one positive and one negative example, so that the denominators in the definitions of μ_0 and μ_1 above are non-zero.)