EECS445: Introduction to Machine Learning, Fall 2014
Homework #1

Due date: 5pm on 9/23 (Tuesday)

Reminder: While you are encouraged to think about problems in small groups, all written solutions must be independently generated. Please type or hand-write solutions legibly. While these questions require thought, please be as concise and clear in your answer as possible. Please address all questions to `http://piazza.com/class#fall2014/eecs445` with a reference to the specific question in the subject line (E.g. RE: Homework 1, Q2(c)). For any solutions that require programming, please submit commented code along with any figures that you are asked to plot.

Remarks about notation: As in the lectures, we will interchangeably use $\mathbf{x}_i$ and $\mathbf{x}^{(i)}$ to denote the i-th training example (and similar notation for other variables). Often, it's less confusing to use $\mathbf{x}^{(i)}$ than $\mathbf{x}_i$ (especially in handwriting) when your expression involves another subscripts to denote specific coordinates in the vector. The Bishop's book uses the former notation, but some other machine learning textbooks use the latter notation. In addition, to be consistent with the Bishop's book, we use t to denote the labels. You are free to use whichever notation as long as you are clear about it. However, please do not mix them in each problem.

Submission Instructions: You must hand-submit your assignment at the Learning Center (BBB 1637). As you enter the Learning Center, immediately to your right, there is a cabinet. The top drawer of the cabinet is labeled "EECS445 - Fall 2014". Drop your hand-written assignment there. Make sure your name is written on your assignment, and that all your pages are stapled together properly. Please do **not** submit code in your hand-written assignment. For programming assignments, include figures and output values (unless there are too much of them) in your hand-written assignment. All code should be zipped (or gzipped) and named *yourFirstName_yourLastName*.zip (or *yourFirstName_yourLastName*.tgz). Organize your matlab files into folders, one folder per question. There is no need to submit the data files - only submit code you wrote.

1 [32 points] Linear regression on a polynomial

The files `q1xTrain.dat`, `q1xTest.dat`, `q1yTrain.dat` and `q1yTest.dat` specify a linear regression problem for a polynomial. `q1xTrain.dat` represent the inputs ($\mathbf{x}^{(i)} \in \mathbb{R}$) and `q1yTrain.dat` represents the outputs ($t^{(i)} \in \mathbb{R}$) of the training set, with one training example per row.

(a) **[12 points]** For each of the following optimization methods, find the coefficients (slope and intercept) that minimize the error for a first degree polynomial.

- Batch gradient descent
- Stochastic gradient descent
- Newton's method (note that this is the same as closed-form solution we covered in class)

i. **[9 points]** Give the coefficients generated by each of the three optimization methods.

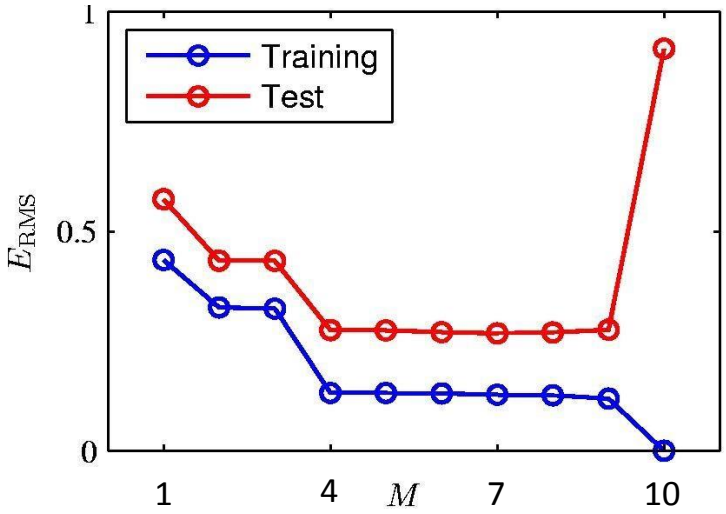
ii. **[3 points]** How did the methods compare in terms of rate of convergence?

(b) **[10 points]** Next, you will investigate the problem of over-fitting. Recall the figure from lecture that explored over-fitting as a function of the degree of the polynomial M, where the Root-Mean-Square (RMS) Error is define as

$$E_{RMS} = \sqrt{2E(\mathbf{w}^*)/N},$$

where

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(\sum_{j=0}^{M-1} w_j\phi_j(\mathbf{x}^{(i)}) - t^{(i)})^2 = \frac{1}{2}\sum_{i=1}^{N}(\mathbf{w}^T\phi(\mathbf{x}^{(i)}) - t^{(i)})^2 :$$



i. **[7 points]** Using Newton's method, find the coefficients that minimize the error for a polynomial with M features (for $M = 1 \ldots 10$) for the training data specified in `q1xTrain.dat` and `q1yTrain.dat`. Now use these parameters to regenerate the above chart, using `q1xTest.dat` and `q1yTest.dat` as the test data.

ii. **[3 points]** Which degree polynomial would you say best fits the data? Was there evidence of under/over-fitting the data? Use your generated charts to defend your answer.

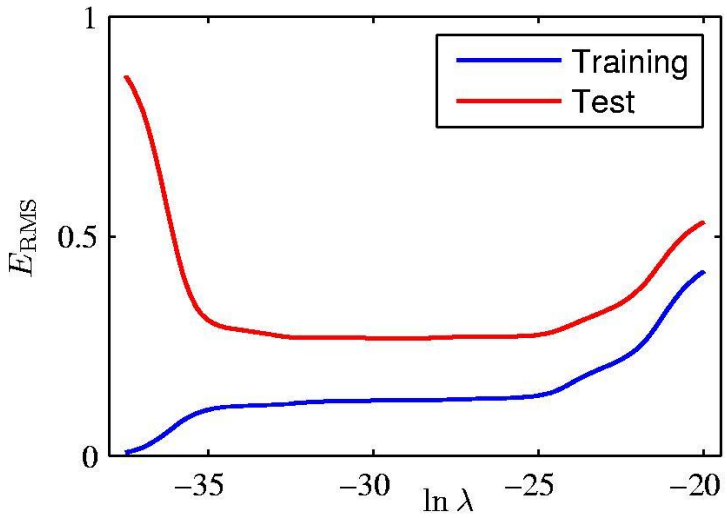
(c) **[10 points]** Finally, you will explore the role of regularization. Recall the image from lecture that explored the affect of the regularization factor λ:

i. **[7 points]** Using Newton's method, find the coefficients that minimize the error for a ninth degree polynomial ($M = 10$) given regularization factor λ (for $\lambda = \{0, 10^{-6}, 10^{-5}, \ldots, 10^{-1}, 10^{0}(1)\}$) for the training data specified in `q1xTrain.dat` and `q1yTrain.dat`. Now use these parameters to plot the E_{RMS} of both the training data and test data as a function of λ (regenerate the above chart, using `q1xTest.dat` and `q1yTest.dat` as the test data). Specifically, use the following regularized objective function:

$$\frac{1}{2}\sum_{i=1}^{N}(\mathbf{w}^T\phi(\mathbf{x}^{(i)}) - t^{(i)})^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2.$$

for optimizing the parameters $\mathbf{w}$, but please use the original (unregularized) E_{RMS} for plotting.

ii. **[3 points]** Which λ value seemed to work the best?



2 [24 points] Locally weighted linear regression

Note: In this problem, we assume that the feature $\phi(\mathbf{x})$ is simply the raw input vector $\mathbf{x}$ (i.e., $\phi(\mathbf{x}) = \mathbf{x}$).

Consider a linear regression problem in which we want to weight different training examples differently. Specifically, suppose we want to minimize

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N} r^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} - t^{(i)})^2.$$

In class, we worked out what happens for the case where all the weights (the $r^{(i)}$'s) are the same. In this problem, we will generalize some of those ideas to the weighted setting, and also implement the locally weighted linear regression algorithm. (Note that the weight $r^{(i)}$ can be different for each of the training example.)

(a) **[2 points]** Show that $E_D(w)$ can also be written

$$E_D(\mathbf{w}) = (\Phi\mathbf{w} - \mathbf{t})^T R(\Phi\mathbf{w} - \mathbf{t})$$

for an appropriate diagonal matrix R, and where Φ and $\mathbf{t}$ are as defined in class. State clearly what R is.

(b) **[6 points]** If all the $r^{(i)}$'s equal 1, then we saw in class that the normal equation is

$$\Phi^T\Phi\mathbf{w} = \Phi^T\mathbf{t},$$

and that the value of $\mathbf{w}^*$ that minimizes $E_D(\mathbf{w})$ is given by $(\Phi^T\Phi)^{-1}\Phi^T\mathbf{t}$. By finding the derivative $\nabla_{\mathbf{w}} E_D(\mathbf{w})$ and setting that to zero, generalize the normal equation to this weighted setting, and give the new value of $\mathbf{w}^*$ that minimizes $E_D(w)$ in closed form as a function of Φ, R and $\mathbf{t}$.

(c) **[5 points]** Suppose we have a training set $\{(\mathbf{x}^{(i)}, t^{(i)}); i = 1\ldots, N\}$ of N independent examples, but in which the $t^{(i)}$'s were observed with differing variances. Specifically, suppose that

$$p(t^{(i)}|\mathbf{x}^{(i)}; \mathbf{w}) = \frac{1}{\sqrt{2\pi}\sigma^{(i)}} \exp(-\frac{(t^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2}{2(\sigma^{(i)})^2})$$

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

(d) **[11 points]** The following items will use the files `q2x.dat` which contains the inputs ($\mathbf{x}^{(i)}$) and `q2y.dat` which contains the outputs ($t^{(i)}$) for a linear regression problem, with one training example per row.

i. **[2 points]** Implement (unweighted) linear regression ($t = \mathbf{w}^T\mathbf{x}$) 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.)

ii. **[6 points]** Implement locally weighted linear regression on this dataset (using the weighted normal equations you derived in part (b)), and plot on the same figure the data and the curve resulting from your fit. When evaluating $h(\cdot)$ at a query point $\mathbf{x}$ (which is real-valued in this problem), use weights

$$r^{(i)} = \exp\left(-\frac{(\mathbf{x} - \mathbf{x}^{(i)})^2}{2\tau^2}\right)$$

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

iii. **[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.

3 [19 points] k-Nearest Neighbors (kNN)

In kNN classification, a system has access to a labeled dataset: $\mathcal{D} = \{(\mathbf{x}^{(1)}, t^{(1)}), (\mathbf{x}^{(2)}, t^{(2)}), \ldots (\mathbf{x}^{(N)}, t^{(N)})\}$. Given an unseen test example $\mathbf{x}_{test}$, a kNN algorithm will predict the example's class (y_{test}) by:

1. Finding the k *closest* examples to $\mathbf{x}_{test}$ from the dataset, i.e. $kNN(\mathbf{x}_{test}) = \{(\mathbf{x'}^{(1)}, t'^{(1)}), (\mathbf{x'}^{(2)}, t'^{(2)}), \ldots, (\mathbf{x'}^{(k)}, t'^{(k)})\}$, such that $\mathbf{x'}^{(1)} \ldots \mathbf{x'}^{(k)}$ are the best k points among all training data at minimizing $d(\mathbf{x}', \mathbf{x}_{test})$, where $d(\mathbf{x}', \mathbf{x})$ is a *distance measure* between $\mathbf{x}'$ and $\mathbf{x}$.[1]

2. Predicting the class label y_{test} as the *mode* class of the k Nearest Neighbors. In particular:

$$y_{test} = \arg\max_t \sum_{(\mathbf{x}', t') \in kNN(\mathbf{x}_{test})} \mathbf{1}[t' = t]$$

Break ties either arbitrarily or randomly.

For this question, you will play with a (*small*) portion of the MNIST dataset (hand-written digits dataset). Given some test example (a digit that has not been seen), your end goal is to classify it into its correct class (0 to 9).

Sample data files are available in ctools. Load the file `q3_digits.mat` which will load matlab variables `digits_train`, `labels_train`, `digits_test` and `labels_test`. The below is a code snippet that can visualize an image from the dataset in matlab; for example, you can view the 55^{th} image by:

```
im = digits_train(55, :, :);
im = reshape(im, 28, 28);
imshow(im);
```

[1] A more precise definition of $kNN(\mathbf{x}_{test})$ would be: if $kNN(\mathbf{x}_{test}) = \{(\mathbf{x'}^{(1)}, t'^{(1)}), (\mathbf{x'}^{(2)}, t'^{(2)}), \ldots, (\mathbf{x'}^{(k)}, t'^{(k)})\}$, where $(\mathbf{x'}^{(i)}, t'^{(i)}) \in D$, then for all i, j such that $(\mathbf{x'}^{(i)}, t'^{(i)}) \in kNN(\mathbf{x}_{test})$ and $(\mathbf{x'}^{(j)}, t'^{(j)}) \in D \backslash kNN(\mathbf{x}_{test})$, $d(\mathbf{x'}^{(i)}, \mathbf{x}_{test}) \leq d(\mathbf{x'}^{(j)}, \mathbf{x}_{test})$

(a) **[9 points]** For the first 5 test `digits_test(1:5, :, :)`, find the 8-Nearest neighbors. Assume that $d(\mathbf{x}, \mathbf{x}')$ is the Euclidean distance between the two examples $\mathbf{x}$ and $\mathbf{x}'$ (calculated on the $28 \times 28 = 784$ pixels). For each of the 5 test digits, write down the *indices* of the 8 nearest neighbors and their *class labels*. In addition, submit the images of each of the 5 images and its 8 nearest neighbors. The code in `q3_starter.m` provides a commented example of how to plot multiple images on the same Matlab figure.

(b) **[8 points]** Classify all the test images (1000 examples) into their digit class using $k = 10$. Submit your code and report the classification accuracy.

(c) **[2 points]** Based on your implementation of kNN, what are advantages and disadvantages of kNN? *[hint: what happens if we have over 100,000 training examples?]* How does the value of k effect the classification accuracy? *[hint: run your code using many different values of k and observe how the accuracy changes.]*

(d) **[(extra credit) 3 points]** What are ways that can improve the accuracy of this kNN classifier? For full credits, implement your ideas in code, report your improved accuracy and submit your code. *[hint: One thing you could do is to come up with a different/better distance $d(.,.)$ function. It could help to view the examples that are incorrectly classified]*

4 [25 points] Logistic regression

(a) **[10 points]** Consider the log-likelihood function for logistic regression:

$$\ell(\mathbf{w}) = \sum_{n=1}^{N} t^{(n)} \log h(\mathbf{x}^{(n)}) + (1 - t^{(n)}) \log(1 - h(\mathbf{x}^{(n)})),$$

where $h(\mathbf{x}) = sigmoid(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x})}$.

First, find the Hessian H. Specifically, show that the i, jth entry ($1 \leq i, j \leq M$) of the Hessian matrix can be written as:

$$H_{ij} = -\sum_{n} \Phi_{ni} h(\mathbf{x}^{(n)})(1 - h(\mathbf{x}^{(n)}))\Phi_{nj}. \tag{1}$$

[Hint: calculate $H_{ij} = \frac{\partial^2 l(\mathbf{w})}{\partial w_i \partial w_j}$ by taking the derivative of $l(\mathbf{w})$ twice with respect to w_i and w_j.]

In addition, show that the Hessian H is negative semi-definite and thus ℓ is concave and has no local maxima other than the global one. That is, show that

$$\mathbf{z}^T H \mathbf{z} \leq 0$$

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

(b) **[10 points]** Using the H you calculated in part (a), write down the update rule implied by Newton's method for optimizing $\ell(\mathbf{w})$. Now use this rule (and not a library function) to implement Newton's method and apply it to binary classification problem specified in files `q4x.dat` and `q4y.dat`. The two columns of `q4x.dat` represent the inputs ($x^{(i)}$) and `q4y.dat` represents the outputs ($t^{(i)} \in \{0, 1\}$), with one training example per row. Initialize Newtons method with $\mathbf{w} = \mathbf{0}$ (the vector of all zeros). What are the coefficients $\mathbf{w}$, including the intercept term, resulting from your fit?

(c) **[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 $h(\mathbf{x}) > 0.5$ from where $h(\mathbf{x}) \leq 0.5$.)

Helpful commands: `plot, hold on, hold off, figure, close.` You can always find detailed information about commands by typing `help` ⟨`command`⟩ in the command window in Matlab.

```
Example Code:
%Plot data points in different classes:
q4xClass0 = q4x(q4y==0,:);
q4xClass1 = q4x(q4y==1,:);
hold on
plot(q4xClass0(:,1),q4xClass0(:,2),?rx?);
plot(q4xClass1(:,1),q4xClass1(:,2),?go?);
hold off
```