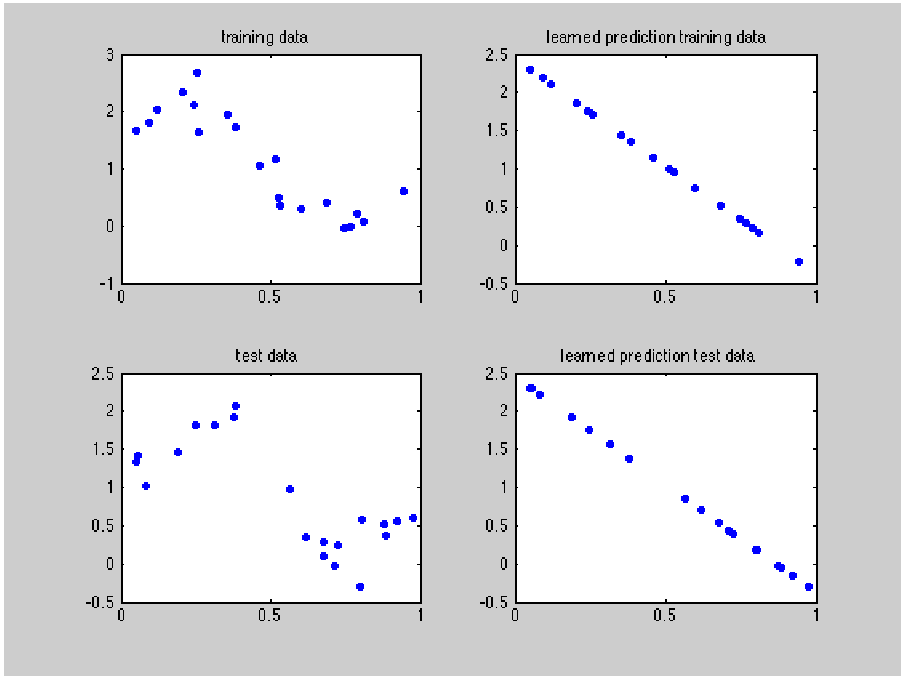
EECS445: Introduction to Machine Learning
Homework #1 Solutions

1 [32 points] Linear regression on a polynomial

(a) **[12 points]**

i. **Answer[9 points]:** These solutions are approx. and exact solutions depend on convergence criterion used (e.g., norm of the gradient is small, reduction in objective value is small, etc.). The following image is not required, just instructive.



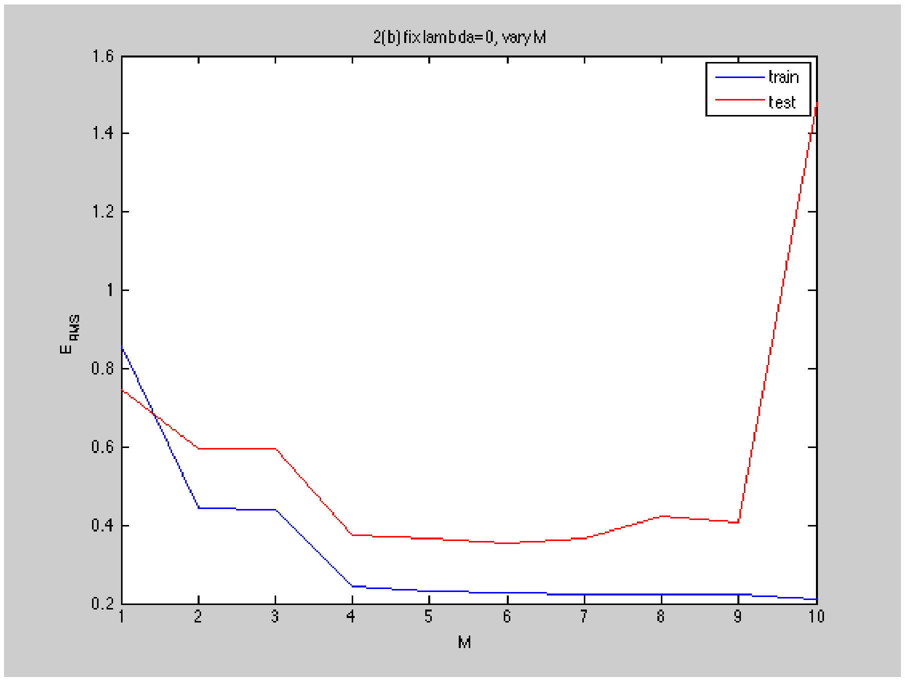
See attached `q1_sol.m`.

- Newton's method gives the **exact** (Maximum Likelihood estimate) weights, as: $y = -2.8164x + 2.4464$. Deduct [**1 marks**] if it is not clear if the each parameter is the coefficient of x or the constant.
- Both Stochastic Gradient Descent and Gradient Descent methods should obtain similar weights to the Newton's method (actual numbers depend on number of iterations and stopping condition). Both answers should be at most 0.01 away from the actual parameters. Deduct [**1 mark**] if if student's solution is out of this range but still close.
- Grade on code correctness and final answers.

ii. **Answer[3 points]:** Newton's method should converge immediately; the Stochastic will converge more quickly than the Batch (in terms of number of "outerloop" iterations). Deduct [**1 mark**] if there is no justification or explanation.

(b) **[10 points]**

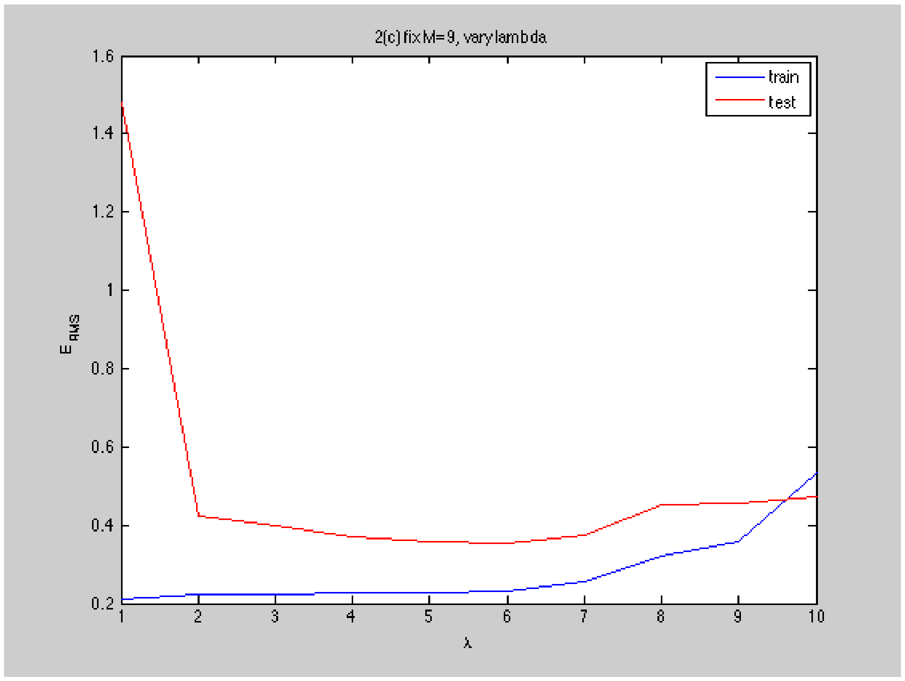
i. **Answer[7 points]:**



ii. **Answer[3 points]:** For the figure above, $M = 6$ minimizes the test error (while also performing reasonably well on training error). **Note: Accept answers if the student's curve looks different. i.e.: if student made mistake on earlier parts and had a low-point on the test curve different than 6, accept it**

(c) **[10 points]**

i. **Answer[7 points]:**



Where $\lambda = \{0, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$ (labeled $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ respectively).

ii. **Answer[3 points]:** $\lambda = 10^{-5}..10^{-4}$ seems to be the "sweet spot" for this particular problem.

2 [24 points] Locally weighted linear regression

(a) **[2 points]** Let $\mathbf{z} = X\mathbf{w} - \mathbf{y}$, i.e. $z_i = \mathbf{w^T x}_i - y_i$. Then we have:

$$\begin{aligned} E_D(\mathbf{w}) &= \frac{1}{2}\sum_{i=1}^{N} r_i(\mathbf{w^T x}_i - y_i)^2 \\ &= \sum_{i=1}^{N} \frac{1}{2} r_i z_i^2 \\ &= \mathbf{z}^T R\mathbf{z} \\ &= (X\mathbf{w} - \mathbf{y})^T R(X\mathbf{w} - \mathbf{y}) \end{aligned}$$

where $R_{ii} = \frac{1}{2}r_i$, $R_{ij} = 0$ for $i \neq j$.

(b) **[6 points]**

$$\nabla_\mathbf{w} E_D(\mathbf{w}) = \nabla_\mathbf{w}(\mathbf{w}^T X^T R X\mathbf{w} + \mathbf{y}^T R\mathbf{y} - 2\mathbf{y}^T R X\mathbf{w}) = 2X^T R X\mathbf{w} - 2X^T R\mathbf{y}$$

So, $\nabla_\mathbf{w} E_D(\mathbf{w}) = 0$ when

$$X^T R X\mathbf{w} = X^T R\mathbf{y}$$

These are the normal equations, from which we can get a closed form formula for $\mathbf{w}$:

$$\mathbf{w} = (X^T R X)^{-1} X^T R\mathbf{y}$$

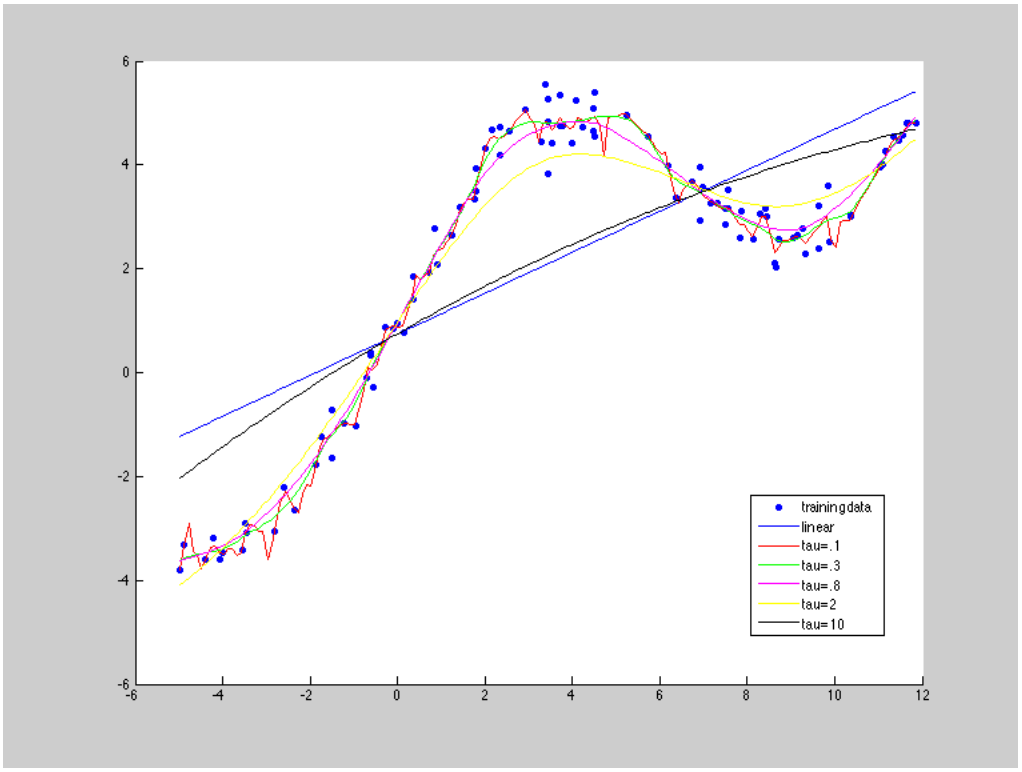
(c) **Answer[5 points]:**

$$\begin{aligned}
\arg\max_{\mathbf{w}} \prod_{i=1}^{N} p(t_i|\mathbf{x}_i;\mathbf{w}) &= \arg\max_{\mathbf{w}} \sum_{i=1}^{N} \log p(t_i|\mathbf{x}_i;\mathbf{w}) \\
&= \arg\max_{\mathbf{w}} \sum_{i=1}^{N} \log \frac{1}{\sqrt{2\pi}\sigma_i} - \frac{(t_i - \mathbf{w}^T\mathbf{x}_i)^2}{2(\sigma_i)^2} \\
&= \arg\min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^{N} \frac{(t_i - \mathbf{w}^T\mathbf{x}_i)^2}{(\sigma_i)^2} \\
&= \arg\min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^{N} \frac{1}{(\sigma_i)^2}(t_i - \mathbf{w}^T\mathbf{x}_i)^2 \\
&= \arg\min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^{N} r_i(t_i - \mathbf{w}^T\mathbf{x}_i)^2
\end{aligned}$$

where in the last step, we substituted $r_i = \frac{1}{(\sigma_i)^2}$ to get the linear regression form.

(d) **Answer[11 points]:**

See attached `q2_sol.m`.



For small bandwidth parameter τ, the fitting is dominated by the closest by training samples. The smaller the bandwidth, the less training samples that are actually taken into account when doing the regression, and the regression results thus become very susceptible to noise in those few training samples. For larger τ, we have enough training samples to reliably fit straight lines, unfortunately a straight line is not the right model for these data, so we also get a bad fit for large bandwidths. Deduct **[2 marks]** for missing plots.

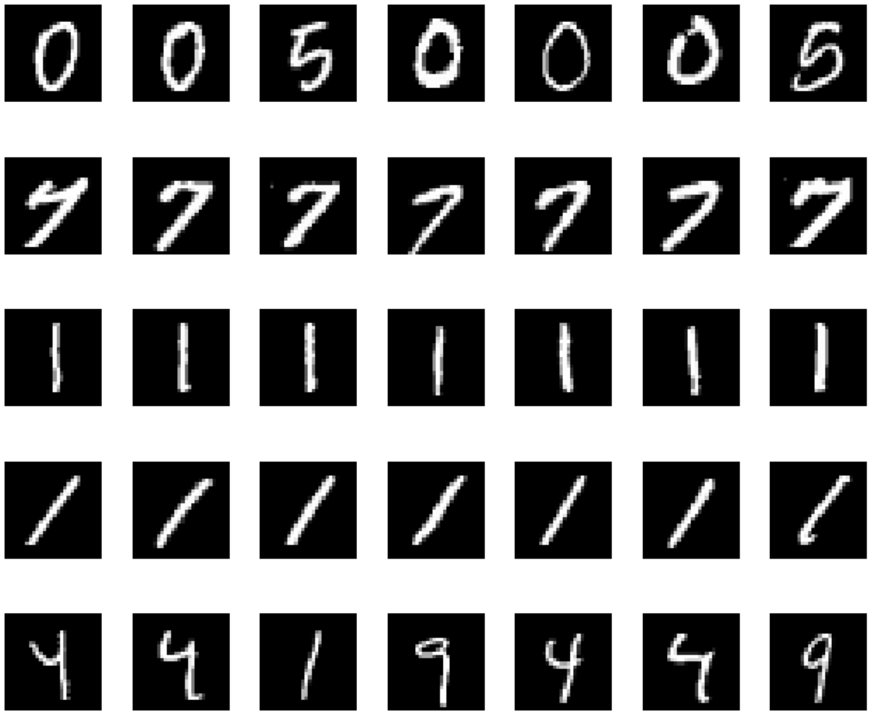
3 [19 points] k-Nearest Neighbors (kNN)

The code for parts (a) and (b) is in `q3_sol.m`.

(a) **Answer[9 points]:** These are the indices for the training examples:

item	first NN	second NN	third NN	fourth NN	fifth NN	sixth NN
1	953	691	321	211	193	837
2	724	754	72	16	912	982
3	485	399	455	639	9	679
4	588	952	594	962	476	638
5	577	605	305	27	719	901

The image below has 5 rows (corresponding to the table above) and 7th columns. The first column is the query, and the subsequent 6 columns are the 6-nearest neighbors:



(b) **Answer[8 points]:** Accuracy is 85.70

(c) **Answer[2 points]:** Note: Any 2 of the below should get full credit.

Advantages:

- It is simple to implement and debug.
- Theoretical guarantee: When K=1 and N goes to infinity, the asymptotic error is at most twice of Bayes optimal error (the minimum achievable error rate of an optimal classifier).

- kNN can be useful in explaining the output of classifier through the analysis of the neighbors.

Disadvantages

- For every test example, the naive implementation computes the distance between the test example and all entries in the training set. So, the complexity is O(N) where N is the size of the training set. This can be very slow when N is large.
- Need a big memory to hold an access all examples.
- kNN is sensitive to distance metric. Depending on how you define the distance metric, you will get quite a different result. One related research topic is distance metric learning, which tries to learn a better distance metric for classification.
- kNN is not robust to irrelevant features. For example, if a certain input feature (coordinate) is distributed from the white noise, then it can still adversely affect the performance of kNN classifier. In such cases, some techniques like feature selection can be helpful.
- On more difficult classification problems, kNN is often outperformed by other discriminative classifiers, like SVM.

(d) **Answer[(extra credit) 3 points]:** It is possible to improve kNN accuracy in multiple ways. Partial credit will be given to solutions that improve the accuracy by 1% to 3% and full credit will be given to solutions that give over 4% accuracy improvements, provided code is submitted. The following are possible things to do:

- Applying gaussian smoothing to the data. (i.e.: convolve the training images and a test image with a gaussian mask). The uploaded solution has Gaussian smoothing coded which can be enabled by commenting line 11 in `q3_knn.m`.
- Do weighted majority vote (rather than majority vote). Where weight is proportional to the position. (i.e.: the 1st nearest neighbor gives more weight to the final answer than the 2nd nearest neighbor and so on).

4 [25 points] Logistic regression

(a) **Answer[10 points]:** (Note we do things in a slightly shorter way here; this solution does not use the hint.) Recall that we have $g'(z) = g(z)(1-g(z))$ where $g(z) = sigmoid(z)$, and thus for $h(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x})$, we have $\frac{\partial h(\mathbf{x})}{\partial \mathbf{w}_k}\partial h(\mathbf{x}) = h(\mathbf{x})(1-h(\mathbf{x}))x_k$.

Remember we have shown in class:

$$\frac{\partial l(\mathbf{w})}{\partial \mathbf{w}_k} = \sum_{i=1}^{N}(t^{(i)} - h(\mathbf{x}^{(i)}))x_k^{(i)}$$

By taking second derivative, we get

$$\begin{aligned} H_{kl} &= \frac{\partial^2 l(\mathbf{w})}{\partial \mathbf{w}_k \partial \mathbf{w}_l} \\ &= \sum_{i=1}^{N} -\frac{\partial h(\mathbf{x}^{(i)})}{\partial \mathbf{w}_l} x_k^{(i)} \\ &= \sum_{i=1}^{N} -h(\mathbf{x}^{(i)})(1-h(\mathbf{x}^{(i)}))x_l^{(i)}x_k^{(i)} \end{aligned}$$

In a matrix form,

$$H = -\sum_{i=1}^{N} h(\mathbf{x}^{(i)})(1 - h(\mathbf{x}^{(i)}))\mathbf{x}^{(i)}\mathbf{x}^{(i)T}$$

To prove H is negative semidefinite, we show $\mathbf{z}^T H\mathbf{z} \leq 0$ for all $\mathbf{z}$.

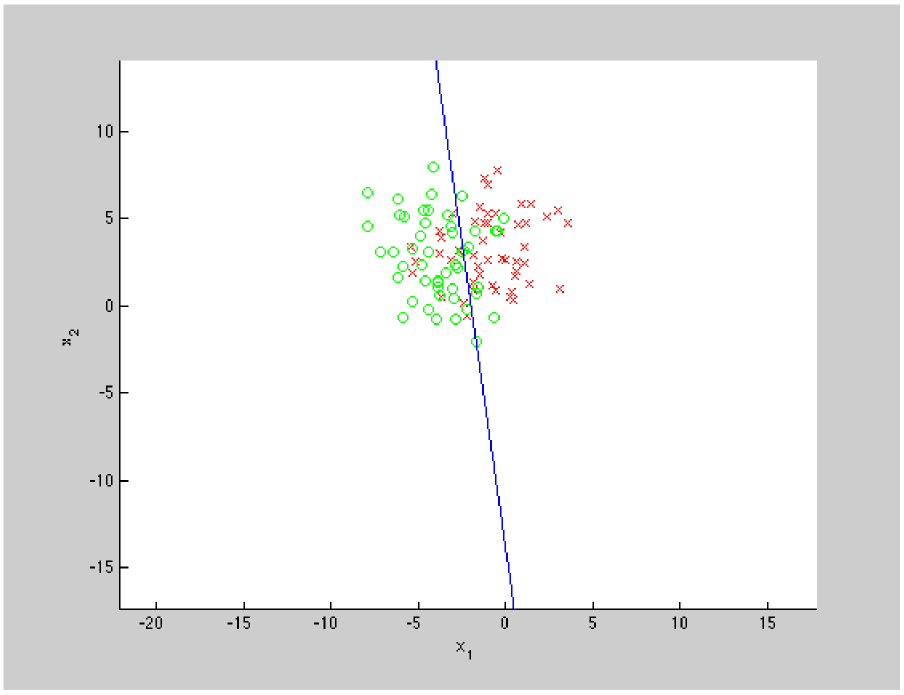
$$\begin{aligned}
\mathbf{z}^T H\mathbf{z} &= \text{-}\mathbf{z}^T \left(\sum_{i=1}^{N} h(\mathbf{x}^{(i)})(1 - h(\mathbf{x}^{(i)}))\mathbf{x}^{(i)}\mathbf{x}^{(i)T} \right) \mathbf{z} \\
&= -\sum_{i=1}^{N} h(\mathbf{x}^{(i)})(1 - h(\mathbf{x}^{(i)}))\mathbf{z}^T\mathbf{x}^{(i)}\mathbf{x}^{(i)T}\mathbf{z} \\
&= -\sum_{i=1}^{N} h(\mathbf{x}^{(i)})(1 - h(\mathbf{x}^{(i)})) \left(\mathbf{z}^T\mathbf{x}^{(i)}\right)^2 \\
&\leq 0
\end{aligned}$$

with the last inequality holding, since $0 \leq h(\mathbf{x}^{(i)}) \leq 1$, which implies $h(\mathbf{x}^{(i)})(1 - h(\mathbf{x}^{(i)})) \geq 0$, and $(\mathbf{z}^T\mathbf{x}^{(i)})^2) \geq 0$.
Deduct [**5 marks**] for missing proof.

(b) **Answer[10 points]:** If an intercept term was added, the coefficients of (intercept, x_1, x_2) are: $\mathbf{w} = (-1.4196, -0.7849, -0.1993)^T$
Deduct [**2 marks**] if no intercept is added.

See attached `q4_sol.m`.

(c) **Answer[5 points]:**



10
5
0
-5
-10
-15
-20
-15
-10
-5
0
5
10
15
x_2
x_1