CS 229, Autumn 2016
Problem Set #1 Solutions: Supervised Learning

Due Wednesday, October 19 at 11:00 am on Gradescope.

Notes: (1) These questions require thought, but do not require long answers. Please be as concise as possible. (2) If you have a question about this homework, we encourage you to post your question on our Piazza forum, at `http://piazza.com/stanford/autumn2016/cs229`. (3) If you missed the first lecture or are unfamiliar with the collaboration or honor code policy, please read the policy on Handout #1 (available from the course website) before starting work. (4) For problems that require programming, please include in your submission a copy of your code (with comments) and any figures that you are asked to plot. If typing your solutions, include your code as text in your PDF. Do not submit extra files. (5) To account for late days, the due date listed on Gradescope is October 22 at 11 am. If you submit after October 19, you will begin consuming your late days. If you wish to submit on time, submit before October 19 at 11 am.

All students must submit an electronic PDF version. We highly recommend typesetting your solutions via latex. If you are scanning your document by cell phone, please check the Piazza forum for recommended scanning apps and best practices.

1. **[25 points] Logistic regression**

(a) [10 points] Consider the average empirical loss (the risk) for logistic regression:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\log(1+e^{-y^{(i)}\theta^T x^{(i)}}) = -\frac{1}{m}\sum_{i=1}^{m}\log(h_\theta(y^{(i)}x^{(i)}))$$

where $h_\theta(x) = g(\theta^T x)$ and $g(z) = 1/(1+e^{-z})$. Find the Hessian H of this function, and show that for any vector z, it holds true that

$$z^T H z \geq 0.$$

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 positive semi-definite, written "$H \succeq 0$." This implies that J is convex, and has no local minima other than the global one.[1] If you have some other way of showing $H \succeq 0$, you're also welcome to use your method instead of the one above.

Answer: (Note we do things in a slightly shorter way here; this solution does not use the hint.) Note that if $g(z) = 1/(1+e^{-z})$, then $g'(z) = g(z)(1-g(z))$. and thus for $h(x) = g(\theta^T x)$, we have $\frac{\partial h(x)}{\partial \theta_k} = h(x)(1-h(x))x_k$. This latter fact is very useful to make the following derivations. Remember we have shown in class:

$$\frac{\partial}{\partial \theta_k}\log(1+e^{-yx^T\theta}) = -\frac{1}{1+e^{yx^T\theta}}yx = -h_\theta(-yx)yx.$$

Thus we have

$$\frac{\partial}{\partial \theta_k}J(\theta) = \frac{1}{m}\sum_{i=1}^{m} -\frac{1}{1+e^{y^{(i)}\theta^T x^{(i)}}}y^{(i)}x_k^{(i)} = -\frac{1}{m}\sum_{i=1}^{m} h_\theta(-y^{(i)}x^{(i)})y^{(i)}x_k^{(i)}.$$

[1] If you haven't seen this result before, please feel encouraged to ask us about it during office hours.

Consequently, we have the Hessian

$$\begin{aligned} H_{kl} &= \frac{\partial^2}{\partial\theta_k \partial\theta_l} J(\theta) \\ &= -\frac{1}{m}\sum_{i=1}^{m} \frac{\partial}{\partial\theta_l} h_\theta(-y^{(i)}x^{(i)}) y^{(i)} x_k^{(i)} \\ &= \frac{1}{m}\sum_{i=1}^{m} h_\theta(x^{(i)})(1-h_\theta(x^{(i)})) x_l^{(i)} x_k^{(i)}. \end{aligned}$$

So we have for the Hessian matrix H (using that for $X = xx^T$ if and only if $X_{ij} = x_i x_j$):

$$H = \frac{1}{m}\sum_{i=1}^{m} h(x^{(i)})(1-h(x^{(i)})) x^{(i)} x^{(i)T}.$$

To prove H is positive semidefinite, we show $z^T H z \geq 0$ for all z:

$$\begin{aligned} z^T H z &= \frac{1}{m} z^T \left(\sum_{i=1}^{m} h(x^{(i)})(1-h(x^{(i)})) x^{(i)} x^{(i)T} \right) z \\ &= \frac{1}{m}\sum_{i=1}^{m} h(x^{(i)})(1-h(x^{(i)})) z^T x^{(i)} x^{(i)T} z \\ &= \frac{1}{m}\sum_{i=1}^{m} h(x^{(i)})(1-h(x^{(i)}))(z^T x^{(i)})^2 \geq 0. \end{aligned}$$

The last inequality holds, because $0 \leq h(x^{(i)}) \leq 1$, which implies $h(x^{(i)})(1-h(x^{(i)})) \geq 0$, and $(z^T x^{(i)})^2) \geq 0$.

(b) [10 points] We have provided two data files:

- `http://cs229.stanford.edu/ps/ps1/logistic_x.txt`
- `http://cs229.stanford.edu/ps/ps1/logistic_y.txt`

These files contain the inputs ($x^{(i)} \in \mathbb{R}^2$) and outputs ($y^{(i)} \in \{-1, 1\}$), respectively for a binary classification problem, with one training example per row. Implement[2] Newton's method for optimizing $J(\theta)$, 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.)

Answer: $\theta = (-2.6205, 0.7604, 1.1719)$ with the first entry corresponding to the intercept term.

```
%%%%%%%   plot_log_regression.m   %%%%%%%

X = load('logistic_x.txt');
Y = load('logistic_y.txt');

X = [ones(size(X, 1), 1) X];
[theta, ll] = log_regression(X ,Y, 20);
```

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

```
m=size(X,1);
figure; hold on;

plot(X(Y < 0, 2), X(Y < 0, 3), 'rx', 'linewidth', 2);
plot(X(Y > 0, 2), X(Y > 0, 3), 'go', 'linewidth', 2);

x1 = min(X(:,2)):.01:max(X(:,2));
x2 = -(theta(1) / theta(3)) - (theta(2) / theta(3)) * x1;

plot(x1,x2, 'linewidth', 2);
xlabel('x1');
ylabel('x2');


%%%%%%% log_regression.m %%%%%%%

function [theta,ll] = log_regression(X,Y, max_iters)

% rows of X are training samples
% rows of Y are corresponding -1/1 values

% newton raphson: theta = theta - inv(H)* grad;
% with H = hessian, grad = gradient

mm = size(X,1);
nn = size(X,2);

theta = zeros(nn,1);

ll = zeros(max_iters, 1);
for ii = 1:max_iters

  margins = Y .* (X * theta);
  ll(ii) = (1/mm) * sum(log(1 + exp(-margins)));
  probs = 1 ./ (1 + exp(margins));
  grad = -(1/mm) * (X' * (probs .* Y));
  H = (1/mm) * (X' * diag(probs .* (1 - probs)) * X);
  theta = theta - H \ grad;
end
```

(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 -1). Also plot on the same figure the decision boundary fit by logistic regression. (This should be a straight line showing the boundary separating the region where $h_\theta(x) > 0.5$ from where $h_\theta(x) \leq 0.5$.)

Answer:

2. **[15 points] Poisson regression and the exponential family**

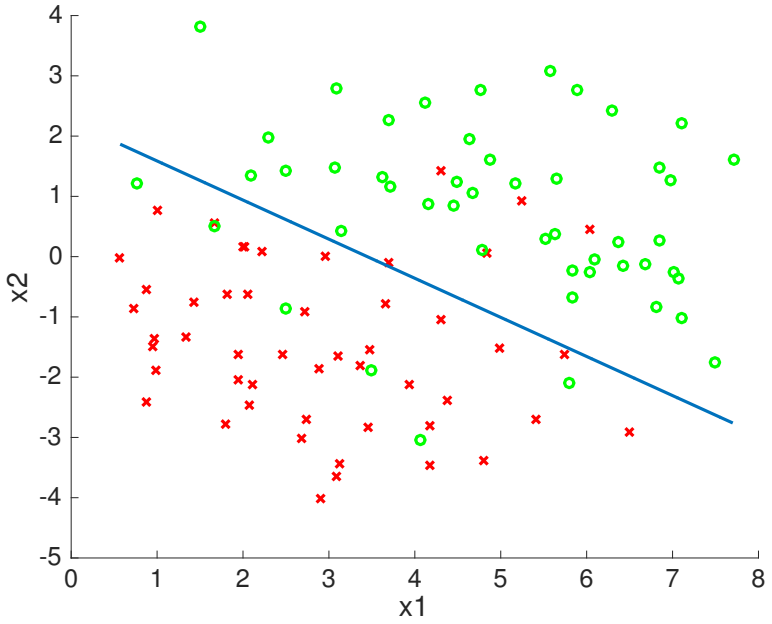


Figure 1: Separating hyperplane for logistic regression (question 1c).

(a) [5 points] 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 $b(y)$, η, $T(y)$, and $a(\eta)$.

Answer: Rewrite the distribution function as:

$$\begin{aligned} p(y;\lambda) &= \frac{e^{-\lambda}e^{y\log\lambda}}{y!} \\ &= \frac{1}{y!}\exp(y\log\lambda - \lambda) \end{aligned}$$

Comparing with the standard form for the exponential family:

$$\begin{aligned} b(y) &= \frac{1}{y!} \\ \eta &= \log\lambda \\ T(y) &= y \\ a(\eta) &= e^{\eta} \end{aligned}$$

(b) [3 points] 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.)

Answer: The canonical response function for the GLM model will be:

$$\begin{aligned} g(\eta) &= E[y;\eta] \\ &= \lambda \\ &= e^{\eta} \end{aligned}$$

(c) [7 points] For a training set $\{(x^{(i)}, y^{(i)});\ i = 1, \ldots, m\}$, let the log-likelihood of an example be $\log p(y^{(i)}|x^{(i)};\theta)$. 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.

Answer: The log-likelihood of an example $(x^{(i)}, y^{(i)})$ is defined as $\ell(\theta) = \log p(y^{(i)}|x^{(i)};\theta)$. To derive the stochastic gradient ascent rule, use the results in part (a) and the standard GLM assumption that $\eta = \theta^T x$.

$$\begin{aligned} \frac{\partial \ell(\theta)}{\partial \theta_j} &= \frac{\partial \log p(y^{(i)}|x^{(i)};\theta)}{\partial \theta_j} \\ &= \frac{\partial \log \left(\frac{1}{y^{(i)}!} \exp(\eta^T y^{(i)} - e^{\eta}) \right)}{\partial \theta_j} \\ &= \frac{\partial \log \left(\exp((\theta^T x^{(i)})^T y^{(i)} - e^{\theta^T x^{(i)}}) \right)}{\partial \theta_j} + \frac{\partial \log \left(\frac{1}{y^{(i)}!} \right)}{\partial \theta_j} \\ &= \frac{\partial \left((\theta^T x^{(i)})^T y^{(i)} - e^{\theta^T x^{(i)}} \right)}{\partial \theta_j} \\ &= \frac{\partial \left((\sum_k \theta_k x_k^{(i)}) y^{(i)} - e^{\sum_k \theta_k x_k^{(i)}} \right)}{\partial \theta_j} \\ &= x_j^{(i)} y^{(i)} - e^{\sum_k \theta_k x_k^{(i)}} x_j^{(i)} \\ &= (y^{(i)} - e^{\theta^T x^{(i)}}) x_j^{(i)} \end{aligned}$$

Thus the stochastic gradient ascent update rule should be:

$$\theta_j := \theta_j + \alpha \frac{\partial \ell(\theta)}{\partial \theta_j}$$

which reduces here to:

$$\theta_j := \theta_j + \alpha (y^{(i)} - e^{\theta^T x}) x_j^{(i)}$$

(d) **[3 extra credit points]** Consider using GLM with a response variable from any member of the exponential family in which $T(y) = y$, and the canonical response function $h(x)$ for the family. Show that stochastic gradient ascent on the log-likelihood $\log p(\vec{y}|X;\theta)$ results in the update rule $\theta_i := \theta_i - \alpha(h(x) - y)x_i$.

Answer: As in the previous part, consider the derivative of the likelihood of a training example

(x, y) with respect to the parameter θ_j:

$$\begin{aligned}
\frac{\partial \ell(\theta)}{\partial \theta_j} &= \frac{\partial \log p(y|x;\theta)}{\partial \theta_j} \\
&= \frac{\partial \log \left(b(y)\exp(\eta^T y - a(\eta))\right)}{\partial \theta_j} \\
&= \frac{\partial \left(\eta^T y - a(\eta))\right)}{\partial \theta_j} \\
&= x_j y - \frac{\partial a(\eta)}{\partial \eta} x_j \\
&= \left(y - \frac{\partial a(\eta)}{\partial \eta}\right) x_j
\end{aligned}$$

Thus, it only remains to show that $\frac{\partial a(\eta)}{\partial \eta} = h(x) = E[y|x;\theta]$. To prove this consider the fact that $p(y|x;\theta)$ is a probability distribution and must thus sum to 1.

$$\begin{aligned}
\int_y p(y|x;\theta)dy &= 1 \\
\int_y b(y)\exp(\eta^T y - a(\eta))dy &= 1 \\
\int_y b(y)\exp(\eta^T y)dy &= \exp(a(\eta))
\end{aligned}$$

Differentiating both sides with respect to η:

$$\begin{aligned}
\int_y b(y) y \exp(\eta^T y)dy &= \exp(a(\eta))\frac{\partial a(\eta)}{\partial \eta} \\
\frac{\partial a(\eta)}{\partial \eta} &= \int_y b(y) y \exp(\eta^T y - a(\eta))dy \\
&= \int_y y p(y|x;\theta)dy \\
&= E[y|x;\theta]
\end{aligned}$$

where the last step follows from the definition of the (conditional) expectation of a random variable. Substituting this into the expression for $\frac{\partial \ell(\theta)}{\partial \theta_j}$ gives the required gradient ascent update rule.

3. **[15 points] Gaussian discriminant analysis**

Suppose we are given a dataset $\{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$ consisting of m independent examples, where $x^{(i)} \in \mathbb{R}^n$ are n-dimensional vectors, and $y^{(i)} \in \{-1, 1\}$. We will model the joint

distribution of (x, y) according to:

$$\begin{aligned} p(y) &= \begin{cases} \phi & \text{if } y = 1 \\ 1-\phi & \text{if } y = -1 \end{cases} \\ p(x|y=-1) &= \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_{-1})^T\Sigma^{-1}(x-\mu_{-1})\right) \\ p(x|y=1) &= \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right) \end{aligned}$$

Here, the parameters of our model are ϕ, Σ, μ_{-1} and μ_1. (Note that while there're two different mean vectors μ_{-1} and μ_1, there's only one covariance matrix Σ.)

(a) [5 points] Suppose we have already fit ϕ, Σ, μ_{-1} 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

$$p(y \mid x; \phi, \Sigma, \mu_{-1}, \mu_1) = \frac{1}{1+\exp(-y(\theta^T x + \theta_0))},$$

where $\theta \in \mathbb{R}^n$ and the bias term $\theta_0 \in \mathbb{R}$ are some appropriate functions of $\phi, \Sigma, \mu_{-1}, \mu_1$. (Note: the term θ_0 corresponds to introducing an extra coordinate $x_0^{(i)} = 1$, as we did in class.)

Answer: For shorthand, we let $\mathcal{H} = \{\phi, \Sigma, \mu_{-1}, \mu_1\}$ denote the parameters for the problem. Since the given formulae are conditioned on y, use Bayes rule to get:

$$\begin{aligned} p(y=1|x;\phi,\Sigma,\mu_{-1},\mu_1) &= \frac{p(x|y=1;\phi,\Sigma,\mu_{-1},\mu_1)p(y=1;\phi,\Sigma,\mu_{-1},\mu_1)}{p(x;\phi,\Sigma,\mu_{-1},\mu_1)} \\ &= \frac{p(x|y=1;\mathcal{H})p(y=1;\mathcal{H})}{p(x|y=1;\mathcal{H})p(y=1;\mathcal{H}) + p(x|y=-1;\mathcal{H})p(y=-1;\mathcal{H})} \\ &= \frac{\exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right)\phi}{\exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right)\phi + \exp\left(-\frac{1}{2}(x-\mu_{-1})^T\Sigma^{-1}(x-\mu_{-1})\right)(1-\phi)} \\ &= \frac{1}{1+\frac{1-\phi}{\phi}\exp\left(-\frac{1}{2}(x-\mu_{-1})^T\Sigma^{-1}(x-\mu_{-1}) + \frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right)} \\ &= \frac{1}{1+\exp\left(\log(\frac{1-\phi}{\phi}) - \frac{1}{2}(x-\mu_{-1})^T\Sigma^{-1}(x-\mu_{-1}) + \frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right)}. \end{aligned}$$

Now, we expand and rearrange the difference of quadratic terms in the preceding expression, finding that

$$\begin{aligned} &(x-\mu_{-1})^T\Sigma^{-1}(x-\mu_{-1}) - (x-\mu_1)^T\Sigma^{-1}(x-\mu_1) \\ &= x^T\Sigma^{-1}x - \mu_{-1}^T\Sigma^{-1}x - x^T\Sigma^{-1}\mu_{-1} + \mu_{-1}^T\Sigma^{-1}\mu_{-1} - x^T\Sigma^{-1}x + \mu_1^T\Sigma^{-1}x + x^T\Sigma^{-1}\mu_1 - \mu_1^T\Sigma^{-1}\mu_1 \\ &= -2\mu_{-1}^T\Sigma^{-1}x + \mu_{-1}^T\Sigma^{-1}\mu_{-1} + 2\mu_1^T\Sigma^{-1}x - \mu_1^T\Sigma^{-1}\mu_1 \\ &= 2(\mu_1-\mu_{-1})^T\Sigma^{-1}x + \mu_{-1}^T\Sigma^{-1}\mu_{-1} - \mu_1^T\Sigma^{-1}\mu_1. \end{aligned}$$

Thus, we have

$$p(y=1 \mid x; \mathcal{H}) = \frac{1}{1+\exp\left(\log\frac{1-\phi}{\phi} + \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 - \frac{1}{2}\mu_{-1}^T\Sigma^{-1}\mu_{-1} + (\mu_{-1}-\mu_1)^T\Sigma^{-1}x\right)}.$$

and setting

$$\theta = \Sigma^{-1}(\mu_1 - \mu_{-1}) \text{ and } \theta_0 = \frac{1}{2}(\mu_{-1}^T\Sigma^{-1}\mu_{-1} - \mu_1^T\Sigma^{-1}\mu_1) - \log\frac{1-\phi}{\phi}$$

gives that

$$p(y \mid x; \phi, \Sigma, \mu_{-1}, \mu_1) = \frac{1}{1+\exp(-y(\theta^T x + \theta_0))}.$$

(b) [10 points] 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

$$\begin{aligned}
\phi &= \frac{1}{m}\sum_{i=1}^m 1\{y^{(i)} = 1\} \\
\mu_{-1} &= \frac{\sum_{i=1}^m 1\{y^{(i)} = -1\}x^{(i)}}{\sum_{i=1}^m 1\{y^{(i)} = -1\}} \\
\mu_1 &= \frac{\sum_{i=1}^m 1\{y^{(i)} = 1\}x^{(i)}}{\sum_{i=1}^m 1\{y^{(i)} = 1\}} \\
\Sigma &= \frac{1}{m}\sum_{i=1}^m (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T
\end{aligned}$$

The log-likelihood of the data is

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

By maximizing ℓ with respect to the four parameters, prove that the maximum likelihood estimates of ϕ, μ_{-1}, 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_{-1} and μ_1 above are non-zero.)

Answer: The derivation follows from the more general one for the next part.

(c) **[3 extra credit points]** Without assuming that $n = 1$, show that the maximum likelihood estimates of ϕ, μ_{-1}, μ_1, and Σ are as given in the formulas in part (b). [Note: If you're fairly sure that you have the answer to this part right, you don't have to do part (b), since that's just a special case.]

Answer: First, derive the expression for the log-likelihood of the training data:

$$\begin{aligned}
\ell(\phi, \mu_{-1}, \mu_1, \Sigma) &= \log\prod_{i=1}^m p(x^{(i)}|y^{(i)}; \mu_{-1}, \mu_1, \Sigma)p(y^{(i)}; \phi) \\
&= \sum_{i=1}^m \log p(x^{(i)}|y^{(i)}; \mu_{-1}, \mu_1, \Sigma) + \sum_{i=1}^m \log p(y^{(i)}; \phi) \\
&\simeq \sum_{i=1}^m \big[\frac{1}{2}\log\frac{1}{|\Sigma|} - \frac{1}{2}(x^{(i)} - \mu_{y^{(i)}})^T\Sigma^{-1}(x^{(i)} - \mu_{y^{(i)}}) + y^{(i)}\log\phi + (1-y^{(i)})\log(1-\phi)\big]
\end{aligned}$$

where constant terms indepedent of the parameters have been ignored in the last expression.

Now, the likelihood is maximized by setting the derivative (or gradient) with respect to each of the parameters to zero.

$$\begin{aligned}\frac{\partial \ell}{\partial \phi} &= \sum_{i=1}^{m} \big[\frac{y^{(i)}}{\phi} - \frac{1-y^{(i)}}{1-\phi}\big] \\ &= \frac{\sum_{i=1}^{m} 1\{y^{(i)}=1\}}{\phi} - \frac{m - \sum_{i=1}^{m} 1\{y^{(i)}=1\}}{1-\phi}\end{aligned}$$

Setting this equal to zero and solving for ϕ gives the maximum likelihood estimate.

For μ_{-1}, take the gradient of the log-likelihood, and then use the same kinds of tricks as were used to analytically solve the linear regression problem.

$$\begin{aligned}\nabla_{\mu_{-1}} \ell &= -\frac{1}{2} \sum_{i:y^{(i)}=-1} \nabla_{\mu_{-1}} (x^{(i)} - \mu_{-1})^T \Sigma^{-1} (x^{(i)} - \mu_{-1}) \\ &= -\frac{1}{2} \sum_{i:y^{(i)}=-1} \nabla_{\mu_{-1}} \big[\mu_{-1}^T \Sigma^{-1} \mu_{-1} - {x^{(i)}}^T \Sigma^{-1} \mu_{-1} - \mu_{-1}^T \Sigma^{-1} x^{(i)}\big] \\ &= -\frac{1}{2} \sum_{i:y^{(i)}=-1} \nabla_{\mu_{-1}} tr\big[\mu_{-1}^T \Sigma^{-1} \mu_{-1} - {x^{(i)}}^T \Sigma^{-1} \mu_{-1} - \mu_{-1}^T \Sigma^{-1} x^{(i)}\big] \\ &= -\frac{1}{2} \sum_{i:y^{(i)}=-1} \big[2\Sigma^{-1}\mu_{-1} - 2\Sigma^{-1} x^{(i)}\big]\end{aligned}$$

The last step uses matrix calculus identities (specifically, those given in page 8 of the lecture notes), and also the fact that Σ (and thus Σ^{-1}) is symmetric.

Setting this gradient to zero gives the maximum likelihood estimate for μ_{-1}. The derivation for μ_1 is similar to the one above.

For Σ, we find the gradient with respect to $S = \Sigma^{-1}$ rather than Σ just to simplify the derivation (note that $|S| = \frac{1}{|\Sigma|}$). You should convince yourself that the maximum likelihood estimate S_m found in this way would correspond to the actual maximum likelihood estimate Σ_m as $S_m^{-1} = \Sigma_m$.

$$\begin{aligned}\nabla_S \ell &= \sum_{i=1}^{m} \nabla_S \big[\frac{1}{2}\log|S| - \frac{1}{2} \underbrace{(x^{(i)} - \mu_{y^{(i)}})^T}_{b_i^T} S \underbrace{(x^{(i)} - \mu_{y^{(i)}})}_{b_i}\big] \\ &= \sum_{i=1}^{m} \big[\frac{1}{2|S|}\nabla_S |S| - \frac{1}{2}\nabla_S b_i^T S b_i\big]\end{aligned}$$

But, we have the following identities:

$$\nabla_S |S| = |S|(S^{-1})^T$$

$$\nabla_S b_i^T S b_i = \nabla_S tr\left(b_i^T S b_i\right) = \nabla_S tr\left(S b_i b_i^T\right) = b_i b_i^T$$

In the above, we again used matrix calculus identities, and also the commutatitivity of the trace operator for square matrices. Putting these into the original equation, we get:

$$\begin{aligned}\nabla_S \ell &= \sum_{i=1}^{m} \big[\frac{1}{2}S^{-1} - \frac{1}{2}b_i b_i^T\big] \\ &= \frac{1}{2}\sum_{i=1}^{m} \big[\Sigma - b_i b_i^T\big]\end{aligned}$$

Setting this to zero gives the required maximum likelihood estimate for Σ.

4. **[10 points] Linear invariance of optimization algorithms**

Consider using an iterative optimization algorithm (such as Newton's method, or gradient descent) to minimize some continuously differentiable function $f(x)$. Suppose we initialize the algorithm at $x^{(0)} = \vec{0}$. When the algorithm is run, it will produce a value of $x \in \mathbb{R}^n$ for each iteration: $x^{(1)}, x^{(2)}, \ldots$.

Now, let some non-singular square matrix $A \in \mathbb{R}^{n\times n}$ be given, and define a new function $g(z) = f(Az)$. Consider using the same iterative optimization algorithm to optimize g (with initialization $z^{(0)} = \vec{0}$). If the values $z^{(1)}, z^{(2)}, \ldots$ produced by this method necessarily satisfy $z^{(i)} = A^{-1}x^{(i)}$ for all i, we say this optimization algorithm is **invariant to linear reparameterizations**.

(a) [7 points] Show that Newton's method (applied to find the minimum of a function) is invariant to linear reparameterizations. Note that since $z^{(0)} = \vec{0} = A^{-1}x^{(0)}$, it is sufficient to show that if Newton's method applied to $f(x)$ updates $x^{(i)}$ to $x^{(i+1)}$, then Newton's method applied to $g(z)$ will update $z^{(i)} = A^{-1}x^{(i)}$ to $z^{(i+1)} = A^{-1}x^{(i+1)}$.[3]

Answer: Let $g(z) = f(Az)$. We need to find $\nabla_z g(z)$ and its Hessian $\nabla_z^2 g(z)$.

By the chain rule:

$$\begin{aligned}\frac{\partial g(z)}{\partial z_i} &= \sum_{k=1}^{n} \frac{\partial f(Az)}{\partial (Az)_k}\frac{\partial (Az)_k}{\partial z_i} &(1)\\ &= \sum_{k=1}^{n} \frac{\partial f(Az)}{\partial (Az)_k} A_{ki} &(2)\\ &= \sum_{k=1}^{n} \frac{\partial f(Az)}{\partial x_k} A_{ki} &(3)\end{aligned}$$

Notice that the above is the same as :

$$\frac{\partial g(z)}{\partial z_i} = A_{\bullet i}^{\top}\nabla_x f(Az) \quad (4)$$

where $A_{\bullet i}$ is the i'th column of A. Then,

$$\nabla_z g(z) = A^{\top}\nabla_x f(Az) \quad (5)$$

[3] Note that for this problem, you must explicitly prove any matrix calculus identities that you wish to use that are not given in the lecture notes.

where $\nabla_x f(Az)$ is $\nabla_x f(\cdot)$ evaluated at Az.

Now we want to find the Hessian $\nabla_z^2 g(z)$.

$$\begin{aligned}\frac{\partial^2 g(z)}{\partial z_i \partial z_j} &= \frac{\partial}{\partial z_j} \sum_{k=1}^{n} \frac{\partial f(Az)}{\partial (Az)_k} A_{ki} &(6)\\ &= \sum_l \sum_k \frac{\partial^2 f(Az)}{\partial x_l \partial x_k} A_{ki} A_{lj} &(7)\end{aligned}$$

If we let $H_f(y)$ denote the Hessian of $f(\cdot)$ evaluated at some point y, and let $H_g(y)$ be the Hessian of $g(\cdot)$ evaluated at some point y, we have from the previous equation that:

$$H_g(z) = A^\top H_f(Az) A \tag{8}$$

We can now put this together and find the update rule for Newton's method on the function $f(Ax)$:

$$\begin{aligned}z^{(i+1)} &= z^{(i)} - H_g(z^{(i)})^{-1} \nabla_z g(z^{(i)}) &(9)\\ &= z^{(i)} - (A^\top H_f(Az^{(i)})A)^{-1} A^\top \nabla_x f(Az^{(i)}) &(10)\\ &= z^{(i)} - A^{-1} H_f(Az^{(i)})^{-1} (A^\top)^{-1} A^\top \nabla_x f(Az^{(i)}) &(11)\\ &= z^{(i)} - A^{-1} H_f(Az^{(i)})^{-1} \nabla_x f(Az^{(i)}) &(12)\end{aligned}$$

Now we have the update rule for $z^{(i+1)}$, we just need to verify that $z^{(i+1)} = A^{-1}x^{(i+1)}$ or equivalently that $Az^{(i+1)} = x^{(i+1)}$. From Eqn. (12) we have

$$\begin{aligned}Az^{(i+1)} &= A\left(z^{(i)} - A^{-1} H_f(Az^{(i)})^{-1} \nabla_x f(Az^{(i)})\right) &(13)\\ &= Az^{(i)} - H_f(Az^{(i)})^{-1} \nabla_x f(Az^{(i)}) &(14)\\ &= x^{(i)} - H_f(x^{(i)})^{-1} \nabla_x f(x^{(i)}) &(15)\\ &= x^{(i+1)}, &(16)\end{aligned}$$

where we used in order: Eqn. (12); rewriting terms; the inductive assumption $x^{(i)} = Az^{(i)}$; the update rule $x^{(i+1)} = x^{(i)} - H_f(x^{(i)})^{-1} \nabla_x f(x^{(i)})$.

(b) [3 points] Is gradient descent invariant to linear reparameterizations? Justify your answer.

Answer:

No. Using the notation from above, gradient descent on $g(z)$ results in the following update rule:

$$z^{(i+1)} = z^{(i)} - \alpha A^\top \nabla_x f(Az^{(i)}). \tag{17}$$

The update rule for $x^{(i+1)}$ is given by

$$x^{(i+1)} = x^{(i)} - \alpha \nabla_x f(x^{(i)}). \tag{18}$$

The invariance holds if and only if $x^{(i+1)} = Az^{(i+1)}$ given $x^{(i)} = Az^{(i)}$. However we have

$$\begin{aligned} Az^{(i+1)} &= Az^{(i)} - \alpha AA^\top \nabla_x f(Az^{(i)}) && (19) \\ &= x^{(i)} - \alpha AA^\top \nabla_x f(x^{(i)}). && (20) \end{aligned}$$

The two expressions in Eqn. (18) and Eqn. (20) are not necessarily equal ($AA^T = I$ requires that A be an orthogonal matrix), and thus gradient descent is not invariant to linear reparameterizations.

5. **[35 points] Regression for denoising quasar spectra**[4]

Introduction. In this problem, we will apply a supervised learning technique to estimate the light spectrum of *quasars*. Quasars are luminous distant galactic nuclei that are so bright, their light overwhelms that of stars in their galaxies. Understanding properties of the spectrum of light emitted by a quasar is useful for a number of tasks: first, a number of quasar properties can be estimated from the spectra, and second, properties of the regions of the universe through which the light passes can also be evaluated (for example, we can estimate the density of neutral and ionized particles in the universe, which helps cosmologists understand the evolution and fundamental laws governing its structure). The *light spectrum* is a curve that relates the light's intensity (formally, lumens per square meter), or *luminous flux*, to its wavelength. Figure 2 shows an example of a quasar light spectrum, where the wavelengths are measured in Angstroms (Å), where 1Å$= 10^{-10}$ meters.

The Lyman-α wavelength is a wavelength beyond which intervening particles at most negligibly interfere with light emitted from the quasar. (Interference generally occurs when a photon is absorbed by a neutral hydrogen atom, which only occurs for certain wavelengths of light.) For wavelengths greater than this Lyman-α wavelength, the observed light spectrum f_{obs} can be modeled as a smooth spectrum f plus noise:

$$f_{\text{obs}}(\lambda) = f(\lambda) + \text{noise}(\lambda)$$

For wavelengths below the Lyman-α wavelength, a region of the spectrum known as the Lyman-α forest, intervening matter causes attenuation of the observed signal. As light emitted by the quasar travels through regions of the universe richer in neutral hydrogen, some of it is absorbed, which we model as

$$f_{\text{obs}}(\lambda) = \text{absorption}(\lambda) \cdot f(\lambda) + \text{noise}(\lambda)$$

Astrophysicists and cosmologists wish to understand the absorption function, which gives information about the Lyman-α forest, and hence the distribution of neutral hydrogen in otherwise unreachable regions of the universe. This gives clues toward the formation and evolution of the universe. Thus, it is our goal to estimate the spectrum f of an observed quasar.

Getting the data. We will be using data generated from the Hubble Space Telescope Faint Object Spectrograph (HST-FOS), Spectra of Active Galactic Nuclei and Quasars.[5] We have provided two comma-separated data files located at:

- Training set: `http://cs229.stanford.edu/ps/ps1/quasar_train.csv`
- Test set: `http://cs229.stanford.edu/ps/ps1/quasar_test.csv`

[4] Ciollaro, Mattia, et al. "Functional regression for quasar spectra." arXiv:1404.3168 (2014).
[5] `https://hea-www.harvard.edu/FOSAGN/`

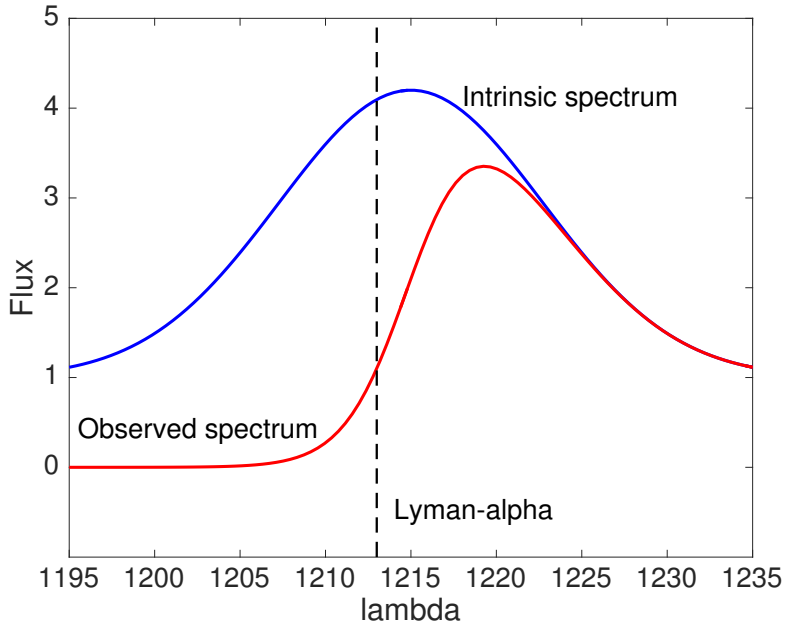


Figure 2: Light spectrum of a quasar. The blue line shows the intrinsic (i.e. original) flux spectrum emitted by the quasar. The red line denotes the observed spectrum here on Earth. To the left of the Lyman-α line, the observed flux is damped and the intrinsic (unabsorbed) flux continuum is not clearly recognizable (red line). To the right of the Lyman-α line, the observed flux approximates the intrinsic spectrum.

Each file contains a single header row containing 450 numbers corresponding integral wavelengths in the interval $[1150, 1600]$ Å. The remaining lines contain relative flux measurements for each wavelength. Specifically, `quasar_train.csv` contains 200 examples and `quasar_test.csv` contains 50 examples. You may use the helper file `load_quasar_data.m` to load the data in Matlab: `http://cs229.stanford.edu/ps/ps1/load_quasar_data.m`

(a) [10 points] Locally weighted linear regression

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

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

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.

i. [2 points] Show that $J(\theta)$ 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. State clearly what W is.

Answer: Let $W_{ii} = \frac{1}{2}w^{(i)}, W_{ij} = 0$ for $i \neq j$, let $\vec{z} = X\theta - \vec{y}$, i.e. $z_i = \theta^T x^{(i)} - y^{(i)}$. Then we have:

$$\begin{aligned}(X\theta - \vec{y})^T W(X\theta - \vec{y}) &= \vec{z}^T W \vec{z} \\ &= \frac{1}{2}\sum_{i=1}^{m} w^{(i)} z_i^2 \\ &= \frac{1}{2}\sum_{i=1}^{m} w^{(i)}(\theta^T x^{(i)} - y^{(i)})^2 \\ &= J(\theta)\end{aligned}$$

ii. [4 points] 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 give the new value of θ that minimizes $J(\theta)$ in closed form as a function of X, W and $\vec{y}$.

Answer:

$$\nabla_\theta J(\theta) = \nabla_\theta(\theta^T X^T W X\theta + \vec{y}^T W\vec{y} - 2\vec{y}^T W X\theta) = 2(X^T W X\theta - X^T W\vec{y}),$$

so we have $\nabla_\theta J(\theta) = 0$ if and only if

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

These are the normal equations, from which we can get a closed form formula for θ.

$$\theta = (X^T W X)^{-1} X^T W\vec{y}$$

iii. [4 points] 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

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

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.

Answer:

$$\begin{aligned}
\arg\max_\theta \prod_{i=1}^m p(y^{(i)}|x^{(i)};\theta) &= \arg\max_\theta \sum_{i=1}^m \log p(y^{(i)}|x^{(i)};\theta) \\
&= \arg\max_\theta \sum_{i=1}^m \left(\log \frac{1}{\sqrt{2\pi}\sigma^{(i)}} - \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2(\sigma^{(i)})^2} \right) \\
&= \arg\max_\theta - \sum_{i=1}^m \frac{(y^{(i)} - \theta^T x^{(i)})^2}{2(\sigma^{(i)})^2} \\
&= \arg\min_\theta \frac{1}{2} \sum_{i=1}^m \frac{1}{(\sigma^{(i)})^2} (y^{(i)} - \theta^T x^{(i)})^2 \\
&= \arg\min_\theta \frac{1}{2} \sum_{i=1}^m w^{(i)} (y^{(i)} - \theta^T x^{(i)})^2
\end{aligned}$$

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

(b) [6 points] Visualizing the data

i. [2 points] Use the normal equations to implement (unweighted) linear regression ($y = \theta^T x$) on the *first* training example (i.e. first non-header row). On one figure, plot both the raw data and the straight line resulting from your fit. State the optimal θ resulting from the linear regression.
Answer: $\theta^* = (2.5134, -0.0010)$

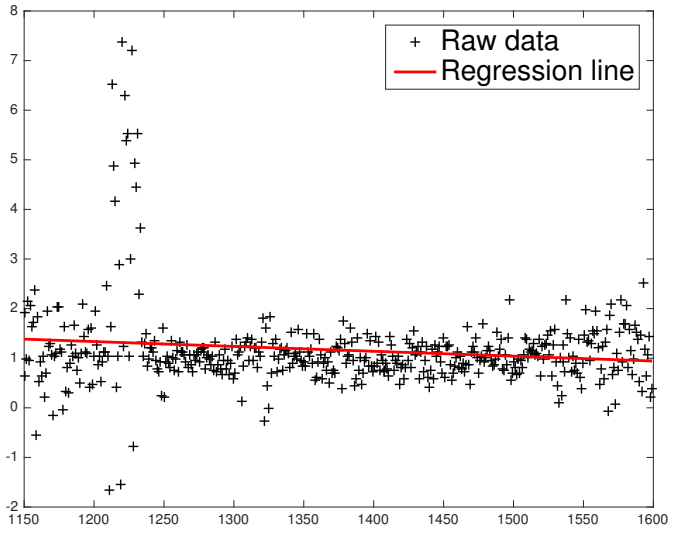


Figure 3: Unweighted linear regression

ii. [2 points] Implement locally weighted linear regression on the *first* training example. Use the normal equations you derived in part (a)(ii). On a different figure, plot both the raw data and the smooth curve resulting from your fit. When evaluating $h(\cdot)$ at a

query point x, use weights

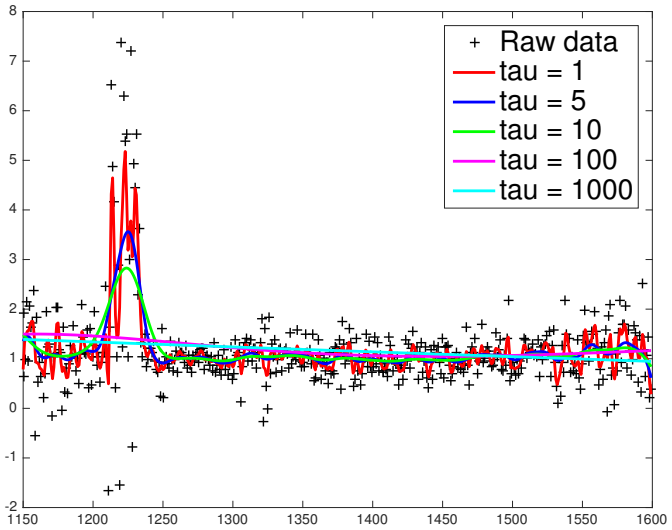
$$w^{(i)} = \exp\left(-\frac{(x - x^{(i)})^2}{2\tau^2}\right),$$

with bandwidth parameter $\tau = 5$.

Answer: See figure below for $\tau = 5$.

iii. [2 points] Repeat (b)(ii) four more times with $\tau = 1, 10, 100$ and 1000. Plot the resulting curves. You can submit one plot with all four τ values or submit four separate plots. If you submit one plot, make sure all curves are visible. Additionally, in **2-3 sentences**, comment on what happens to the locally weighted linear regression line as τ varies.

Answer: As τ becomes large, locally weighted linear regression becomes unweighted linear regression. As τ becomes small, it begins to overfit the data.



(c) [19 points] Predicting quasar spectra with functional regression

We now go a step beyond what we have covered explicitly in class, and we wish to predict an entire part of a spectrum—a curve—from noisy observed data. We begin by supposing that we observe a random sample of m absorption-free spectra, which is possible for quasars very close (in a sense relative to the size of the universe!) to Earth. For a given spectrum f, define f_{right} to be the spectrum to the right of the Lyman-α line. Let f_{left} be the spectrum within the Lyman-α forest region, that is, for lower wavelengths. To make the results cleaner, we define:

$$f(\lambda) = \begin{cases} f_{\text{left}}(\lambda) & \text{if } \lambda < 1200 \\ f_{\text{right}}(\lambda) & \text{if } \lambda \geq 1300 \end{cases}$$

We will learn a function r (for regression) that maps an observed f_{right} to an unobserved target f_{left}. This is useful in practice because we observe f_{right} with *only* random noise: there is no systematic absorption, which we cannot observe directly, because hydrogen does not absorb photons with higher wavelengths. By predicting f_{left} from a noisy version of f_{right}, we can estimate the unobservable spectrum of a quasar as well as the absorption function. Imaging systems collect data of the form

$$f_{\text{obs}}(\lambda) = \text{absorption}(\lambda) \cdot f(\lambda) + \text{noise}(\lambda)$$

for $\lambda \in \{\lambda_1, \ldots, \lambda_n\}$, a *finite* number of points λ, because they must quantize the information. That is, even in the quasars-close-to-Earth training data, our observations of f_{left} and f_{right} consist of noisy evaluations of the true spectrum f at multiple wavelengths. In our case, we have $n = 450$ and $\lambda_1 = 1150, \ldots, \lambda_n = 1599$.

We formulate the functional regression task as the goal of learning the function r mapping f_{left} to f_{right}:

$$r(f_{\text{right}})(\lambda) = \mathbb{E}(f_{\text{left}} \mid f_{\text{right}})(\lambda)$$

for λ in the Lyman-α forest.

i. [1 points] First, we must smooth the data in the training dataset to make it more useful for prediction. For each $i = 1, \ldots, m$, define $f^{(i)}(\lambda)$ to be the weighted linear regression estimate the i^{th} spectrum. Use your code from part (b)(ii) above to smooth all spectra in the training set using $\tau = 5$. Do the same for the test set. We will now operate on these smoothed spectra.

ii. [14 points] Using your estimated regression functions $f^{(i)}$ for $i = 1, \ldots, m$, we now wish to estimate the unobserved spectrum f_{left} of a quasar from its (noisy) observed spectrum f_{right}. To do so, we perform a weighted regression of the *locally weighted regressions*. In particular, given a new noisy spectrum observation:

$$f_{\text{obs}}(\lambda) = f(\lambda) + \text{noise}(\lambda) \quad \text{for } \lambda \in \{1300, \ldots, 1599\}.$$

We define a metric d which takes as input, two spectra f_1 and f_2, and outputs a scalar:

$$d(f_1, f_2) = \sum_i \Big(f_1(\lambda_i) - f_2(\lambda_i)\Big)^2.$$

The metric d computes squared distance between the new datapoint and previous datapoints. If f_1 and f_2 are right spectra, then we take the preceding sum only over $\lambda \in \{1300, \ldots, 1599\}$, rather than the entire spectrum.

Based on this distance function, we may define the nonparametric *functional* regression estimator, which is a locally weighted sum of *functions* f_{left} from the training data (this is like locally weighted linear regression, except that instead of predicting $y \in \mathbb{R}$ we predict a function f_{left}). Specifically, let f_{right} denote the right side of a spectrum, which we have smoothed using locally weighted linear regression (as you were told to do in the previous part of the problem). We wish to estimate the associated *left* spectrum f_{left}. Define the function $\ker(t) = \max\{1 - t, 0\}$ and let $\mathsf{neighb}_k(f_{\text{right}})$ denote the k indices $i \in \{1, 2, \ldots, m\}$ that are closest to f_{right}, that is

$$d(f^{(i)}_{\text{right}}, f_{\text{right}}) < d(f^{(j)}_{\text{right}}, f_{\text{right}}) \quad \text{for all } i \in \mathsf{neighb}_k(f_{\text{right}}),\ j \notin \mathsf{neighb}_k(f_{\text{right}})$$

and $\mathsf{neighb}_k(f_{\text{right}})$ contains exactly k indices. In addition, let

$$h := \max_{i \in \{1, \ldots, m\}} d(f^{(i)}_{\text{right}}, f_{\text{right}}).$$

Then define the estimated function $\widehat{f_{\text{left}}} : \mathbb{R} \to \mathbb{R}$ by

$$\widehat{f_{\text{left}}}(\lambda) = \frac{\sum_{i \in \mathsf{neighb}_k(f_{\text{right}})} \ker(d(f^{(i)}_{\text{right}}, f_{\text{right}})/h) f^{(i)}_{\text{left}}(\lambda)}{\sum_{i \in \mathsf{neighb}_k(f_{\text{right}})} \ker(d(f^{(i)}_{\text{right}}, f_{\text{right}})/h)}. \tag{21}$$

Recall that $f_{\text{right}}^{(i)}$ is the *smoothed* (weighted linear regression) estimate of the ith training spectrum.
Construct the functional regression estimate (21) for each spectrum in the entire training set using $k = 3$ nearest neighbors: for each $j = 1, \ldots, m$, construct the estimator $\widehat{f_{\text{left}}}$ from (21) using $f_{\text{right}} = f_{\text{right}}^{(j)}$. Then compute the error $d(f_{\text{left}}^{(j)}, \widehat{f_{\text{left}}})$ between the true spectrum $f_{\text{left}}^{(j)}$ and your estimated spectrum $\widehat{f_{\text{left}}}$ for each j, and return the average over the training data. What is your average training error?
Answer: We achieve a training set error of 1.0664.

iii. [4 points] Perform functional regression on the test set using the same procedure as in the previous subquestion. What is your average test error? For test examples 1 and 6, include a plot with both the entire smooth spectrum and the fitted curve $\widehat{f_{\text{left}}}$ curve on the same graph. You should submit two plots: one for test example 1 and one for test example 6.
Answer: We achieve a test set error of 2.7100.

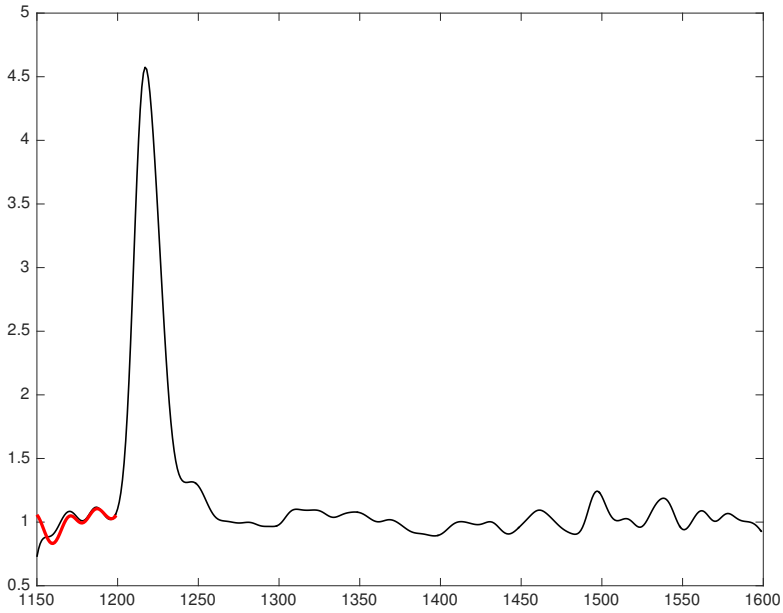


Figure 4: Resulting functional regression for test set example 1.

Code for the entire problem is included below
Answer:

```
function yhat = local_linear_regression(x, y, tau)
% LOCAL_LINEAR_REGRESSION Performs a local linear regression to smooth the
% given input signal.
%
% yhat = local_linear_regression(x, y, tau) takes as input the vectors x
% and y, both of the same dimension. Then, at each point x in the given
% vector, fits a local linear regression using the features (1, x) at that
```

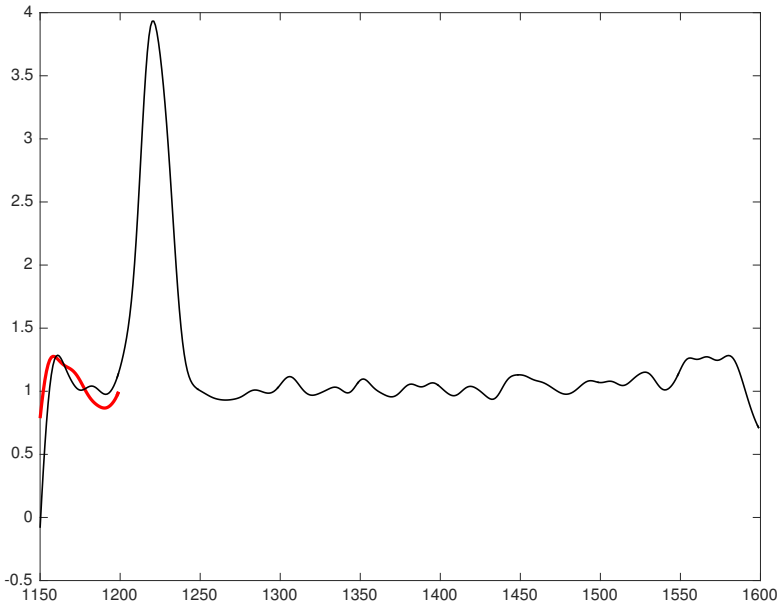


Figure 5: Resulting functional regression for test set example 6.

```
% point, with weights given by
%
% w^i(x) = exp(-(x - x^i)^2 / (2 * tau^2)),
%
% that is, transforms the input so that
%
% yhat(i) = [1, x(i)] * theta^(i)
%
% where theta^(i) minimizes
%
% J_i(theta) = sum_{j=1}^m w^j(x(i)) * (y(j) - [1 x(j)] * theta)^2.

if (length(x) ~= length(y))
  error('Length of x (%d) not same as y (%d)', length(x), length(y));
end
nn = length(x);

X = [ones(nn, 1), x];
yhat = zeros(nn, 1);

for ii = 1:nn
  w = exp(-(x - x(ii)).^2 / (2 * tau^2));
  XwX = X' * ([w, w] .* X);
  XtWy = X' * (w .* y);
  theta = XwX \ XtWy;
  yhat(ii) = [1 x(ii)] * theta;
```

```
end

% Full quasar solution, including plots

load_quasar_data;

[mm, nn] = size(train_qso);

mtest = size(test_qso, 1);

%% Part (a)i: Linear regression

y = train_qso(1, :)';
X = [ones(nn, 1), lambdas];
theta = X\y;  % Solves linear regression directly

figure;
h = plot(lambdas, train_qso(1, :), 'k+');
set(h, 'linewidth', 1);
hold on;
h = plot(lambdas, theta(1) + lambdas * theta(2), 'r-');
set(h, 'linewidth', 2);
h = legend('Raw data', 'Regression line');
set(h, 'fontsize', 20);

print -depsc2 quasar_linear_regression.eps;

%% Part (a)ii/iii: Smoothing the quasars with LWLR

figure;
X = [ones(nn, 1), lambdas];
y = train_qso(1, :)';
h = plot(lambdas, y, 'k+');
set(h, 'linewidth', 1);
hold on;
colors = {'r-', 'b-', 'g-', 'm-', 'c-'};
taus = [1, 5, 10, 100, 1000];
for tau_ind = 1:5
  tau = taus(tau_ind);
  y_smooth = local_linear_regression(lambdas, y, tau);
  h = plot(lambdas, y_smooth, char(colors(tau_ind)));
  set(h, 'linewidth', 2);
end

h = legend('Raw data', 'tau = 1', 'tau = 5', 'tau = 10', ...
           'tau = 100', 'tau = 1000');
set(h, 'fontsize', 20);

print -depsc2 sprintf('quasar_locally_taus.eps', tau);
```

```
%% Part (b)i: Smooth all quasars with LWLR

train_smooth = train_qso;
test_smooth = test_qso;
tau = 5;

X = [ones(nn, 1), lambdas];

for jj = 1:mm
  ytrain = train_qso(jj, :)';
  train_smooth(jj, :) = local_linear_regression(lambdas, ytrain, tau)';
end

for jj = 1:mtest
  ytest = test_qso(jj, :)';
  test_smooth(jj, :) = local_linear_regression(lambdas, ytest, tau)';
end

%% Find the right-most function parts
right_trains = train_smooth(:, 151:end);
left_trains = train_smooth(:, 1:50);
right_tests = test_smooth(:, 151:end);
left_tests = test_smooth(:, 1:50);

%% Construct matrix of all pairs of distances between training quasar spectra
train_dists = zeros(mm, mm);
for ii = 1:mm
  for jj = (ii + 1):mm
    train_dists(ii, jj) = norm(right_trains(ii, :) - right_trains(jj, :))^2;
  end
end
train_dists = train_dists + train_dists';
train_dists = train_dists / max(train_dists(:));

%% Reconstruct training curves

f_left_estimates = zeros(mm, 50);
num_nearest = 3;

for ii = 1:mm
  [train_dist_sort, inds] = sort(train_dists(:, ii), 1, 'ascend');
  close_inds = ones(mm, 1);
  close_inds(inds((num_nearest + 1):end)) = 0;
  h = max(train_dists(:, ii));
  kerns = max(1 - train_dists(:, ii) / h, 0); % An m-by-1 vector
  kerns = kerns .* close_inds;
  f_left_estimates(ii, :) = left_trains' * kerns / sum(kerns);
end
```

```
% Compute error rate of estimates
err = sum((left_trains(:) - f_left_estimates(:)).^2);
err = err / mm;
fprintf(1, 'Average training error: %1.4f\n', err);

%% Reconstruct test curves

% Construct matrix of all pairs of distances between training and testing
% quasar spectra

train_to_test_dists = zeros(mm, mtest);
for ii = 1:mm
  for jj = 1:mtest
    train_to_test_dists(ii, jj) = ...
        norm(right_trains(ii, :) - right_tests(jj, :))^2;
  end
end
train_to_test_dists = train_to_test_dists / max(train_to_test_dists(:));

f_left_estimates = zeros(mtest, 50);
for ii = 1:mtest
  [tttd_sorted, inds] = sort(train_to_test_dists(:, ii), 1, 'ascend');
  close_inds = ones(mm, 1);
  close_inds(inds((num_nearest + 1):end)) = 0;
  h = max(train_to_test_dists(:, ii));
  kerns = max(1 - train_to_test_dists(:, ii) / h, 0);
  kerns = kerns .* close_inds;
  f_left_estimates(ii, :) = left_trains' * kerns / sum(kerns);
end

%% Compute error rate of estimates
err = sum((left_tests(:) - f_left_estimates(:)).^2);
err = err / mtest;
fprintf(1, 'Average testing error: %1.4f\n', err);

%% Final plots
figure;
plot(lambdas, test_smooth(1, :), 'k-', 'linewidth', 1);
hold on;
plot(lambdas(1:50), f_left_estimates(1, :), 'r-', 'linewidth', 2);
print -depsc2 'quasar_test_1.eps';

figure;
plot(lambdas(1:50), f_left_estimates(6, :), 'r-', 'linewidth', 2)
hold on;
plot(lambdas, test_smooth(6, :), 'k-', 'linewidth', 1);
print -depsc2 'quasar_test_6.eps';
```

Reminder: Please include in your submission a printout of your code and figures for the programming questions.