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

Godfray Qiu

January 9, 2017

1. Logistic Regression

(a) 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.

We know $g'(x) = g(x)(1-g(x))$ and $g(x) + g(-x) = 1$ which may help.

$$\frac{\partial J(\theta)}{\partial \theta_k} = -\frac{1}{m}\sum_{i=1}^{m}\frac{\log(h_\theta(y^{(i)}x^{(i)}))}{\partial\theta_k} = -\frac{1}{m}\sum_{i=1}^{m}\left[1-h_\theta(y^{(i)}x^{(i)})\right]y^{(i)}x_k^{(i)};$$

$$\frac{\partial^2 J(\theta)}{\partial\theta_k\partial\theta_l} = \frac{\partial}{\partial\theta_l}\left(-\frac{1}{m}\sum_{i=1}^{m}\left[1-h_\theta(y^{(i)}x^{(i)})\right]y^{(i)}x_k^{(i)}\right) = \frac{\partial}{\partial\theta_l}\left(\frac{1}{m}\sum_{i=1}^{m}\left[h_\theta(y^{(i)}x^{(i)})\right]y^{(i)}x_k^{(i)}\right)$$

By rule of chain,

$$H_{kl} = \frac{\partial^2 J(\theta)}{\partial\theta_k\partial\theta_l} = \frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)2}h_\theta(y^{(i)}x^{(i)})\left(1-h_\theta(y^{(i)}x^{(i)})\right)x_k^{(i)}x_l^{(i)}\right]$$

After vectorization,

$$H = \frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)2}h_\theta(y^{(i)}x^{(i)})\left(1-h_\theta(y^{(i)}x^{(i)})\right)x^{(i)}x^{(i)T}\right]$$

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

Take vector $w \in \mathbb{R}^n$,

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

For $0 < h(t) < 1$ for all $t \in \mathbb{R}$, that is, $h(t)(1 - h(t)) > 0$, and $w^T x^{(i)} x^{(i)T} w = (w^T x^{(i)})^2 \geq 0$, H is PSD.

(b) We have provided two data files:

- logistic_x.txt
- 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 $\theta = \overrightarrow{0}$ (the vector of all zeros). What are the coefficients θ resulting from your fit? (Remember to include the intercept term.)

The Newton's method says $\theta := \theta - H^{-1} \nabla_\theta \ell(\theta)$, from (a), after 6 iterations, $\theta = [-2.6205, 0.7604, 1.1719]^T$.

```
X = load('logistic_x.txt');
y = load('logistic_y.txt');

pos = find(y == 1);
neg = find(y ~= 1);

X = [ones(size(y)), X];
m = size(X,1);
n = size(X,2);

figure;hold on;
plot(X(pos,2:2), X(pos, 3:3), 'k+');
plot(X(neg,2:2), X(neg, 3:3), 'ko', 'MarkerFaceColor', 'y');

theta = zeros(n,1);
max_iter = 100;
grad_l = zeros(n,1);
all_one = ones(m,1);
H = zeros(n);

for i = 1:max_iter
    g = sigmoid(y .* (X* theta));  % sigmoid function
    grad_l = X'*((all_one - g) .* y) /(-m);
    H = (X'* diag(g .* (all_one - g)) *X)/m;
    old_theta = theta;
    fprintf('iteration %d', i);
    theta = theta - pinv(H)*grad_l
    if (old_theta - theta)'*(old_theta - theta)<0.0000001
        break;
    end
end

x1 = min(X(:,2)):.01:max(X(:,2));
```

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

```
x2 = -(theta(1) / theta(3)) - (theta(2) / theta(3)) * x1;
plot(x1,x2, 'linewidth', 2);
xlabel('x1');
ylabel('x2');
```

(c) 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 $h_\theta(x) > 0.5$.)

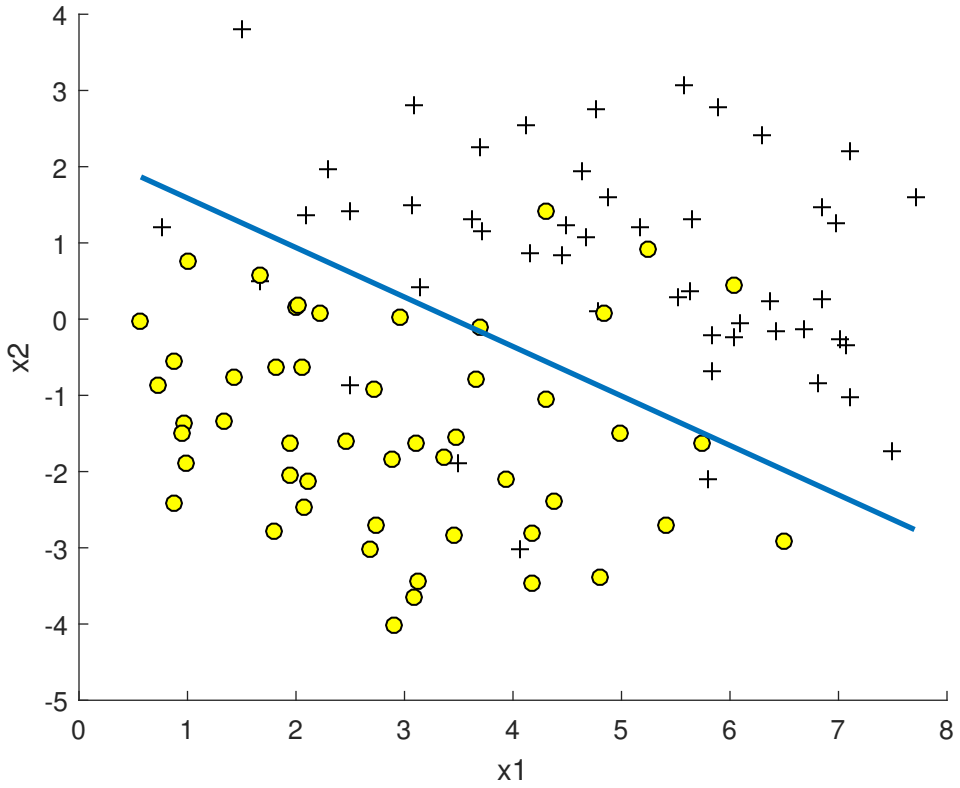

Figure 1: Separating hyperplane for logistic regression (question 1c).

2. Poisson regression and the exponential family

(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 $b(y)$, η, $T(y)$, and $a(\eta)$.

Usually, $p(y;\eta) = b(y)\exp\left(\eta^T T(y) - a(\eta)\right)$, and rewrite p,

$$p(y;\lambda) = \frac{e^{-\lambda}\lambda^y}{y!} = \frac{\exp\left(-\lambda + y\log\lambda\right)}{y!}$$

So, $b(y) = 1/(y!)$, $T(y) = y$, $\eta = \log\lambda$, and $a(\eta) = \lambda = e^\eta$.

(b) Consider 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.)

Canonical response function $g(\eta) = E[T(y);\eta] = E[y;\eta] = \lambda = e^\eta$.

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

Commonly, in stochastic gradient ***ascent***, $\theta_j := \theta_j + \alpha\frac{\partial}{\partial\theta_j}\log p(y^{(i)}|x^{(i)};\theta)$.

In a GLM model, $h_\theta(x) = E[y|x] = \lambda = e^\eta = e^{\theta^T x}$, and the gradient of θ_j is $(y - h_\theta(x))x_j$.

To derive, we have $\eta = \theta^T x$, and $\frac{\partial\eta}{\partial\theta_j} = x_j$

$$\frac{\partial}{\partial\theta_j}\log p = \frac{\partial}{\partial\theta_j}\left(\eta^T T(y) - a(\eta) + \log b(y)\right) = \frac{\partial}{\partial\theta_j}\left(y\eta - a(\eta)\right) = (y - a'(\eta))\frac{\partial\eta}{\partial\theta_j},$$

Common conclusion:

$$\frac{\partial}{\partial\theta_j}\log p = (y - a'(\eta))x_j.$$

So,

$$\frac{\partial}{\partial\theta_j}\log p = (y - e^{\theta^T x})x_j,$$

And finally, $\theta_j := \theta_j + \alpha(y - e^{\theta^T x})x_j$ is the stochastic gradient ***ascent*** rule for learning.

(d) 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 $\log p(\overrightarrow{y}|X;\theta)$ results in the update rule $\theta_i := \theta_i - \alpha(h(x) - y)x_i$.

That is to say, the gradient of θ_i is always $(y - h_\theta(x))x_i$. In the exponential family of $T(y) = y$,

$$\frac{\partial}{\partial\theta_i}\log p = \frac{\partial}{\partial\theta_i}\left(\eta^T T(y) - a(\eta) + \log b(y)\right) = \frac{\partial}{\partial\theta_i}\left(y\eta - a(\eta)\right) = (y - \frac{\partial}{\partial\eta}a(\eta))\frac{\partial\eta}{\partial\theta_i} = (y - \frac{\partial}{\partial\eta}a(\eta))x_i,$$

We have to prove that $\frac{\partial}{\partial\eta}a(\eta)$ equals to $h_\theta(x) = g(\eta) = E[y|x;\theta]$.

Next, take from the SOLUTION MANUAL:

$$\int_y p(y|x;\theta)dy = 1$$

$$\int_y b(y)\exp(\eta^T y - a(\eta))dy = 1$$

$$\int_y b(y)\exp(\eta^T y)dy = \exp(a(\eta))$$

Differentiating both sides with respect to η:

$$\int_y b(y)y\exp(\eta^T y)dy = \exp(a(\eta))\frac{\partial a(\eta)}{\partial \eta}$$

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

3. Gaussian discriminant analysis

Suppose we are given a data set $\{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$ 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) 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

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

where $\theta \in \mathbb{R}^n$ and the bias term $\theta_0 \in \mathbb{R}$ are some appropriate functions of ϕ, Σ, μ_{-1} and μ_1. (Note: the term θ_0 corresponds to introducing an extra coordinate $x_0^{(i)} = 1$, as we did in class.)

By Bayes rule, we have

$$p(y|x;\theta) = \frac{p(x|y)p(y)}{p(x)}$$

By law of total probability, we have

$$p(x) = p(x|y=1)p(y=1) + p(x|y=-1)p(y=-1)$$

Considering

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

We make the numerator 1,

$$p(y=1|x;\theta) = \frac{1}{1+\exp\left(\log\frac{1-\phi}{\phi} + \frac{1}{2}\left[(x-\mu_1)^T\Sigma^{-1}(x-\mu_1) - (x-\mu_{-1})^T\Sigma^{-1}(x-\mu_{-1})\right]\right)}$$

and Σ is symmetric,

$$\frac{1}{2}\left[(x-\mu_1)^T\Sigma^{-1}(x-\mu_1) - (x-\mu_{-1})^T\Sigma^{-1}(x-\mu_{-1})\right] = \frac{1}{2}\left(\mu_1^T\Sigma^{-1}\mu_1 - \mu_{-1}^T\Sigma^{-1}\mu_{-1}\right) + (\mu_{-1}^T - \mu_1^T)\Sigma^{-1}x$$

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

Considering factor of $-y = -1$, we get $\theta_0 = \log\frac{\phi}{1-\phi} - \frac{1}{2}\left(\mu_1^T\Sigma^{-1}\mu_1 - \mu_{-1}^T\Sigma^{-1}\mu_{-1}\right)$
and $\theta^T = -(\mu_{-1} - \mu_1)^T\Sigma^{-1}$, that is $\theta = \Sigma^{-1}(\mu_1 - \mu_{-1})$.

(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 data set, 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, Σ, μ_{-1} and μ_1, 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.)

(c) Without assuming that $n = 1$, show that the maximum likelihood estimates of ϕ, Σ, μ_{-1} and μ_1 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.]

Taken from **SOLUTION MANUAL**:

$\ell(\phi, \mu_{-1}, \mu_1, \Sigma) \simeq$

$$\sum_{i=1}^{m}\left[\frac{1}{2}\log\frac{1}{|\Sigma|} - \frac{1}{2}(x^{(i)} - \mu_{y^{(i)}})^T\Sigma^{-1}(x^{(i)} - \mu_{y^{(i)}})\right] + \sum_{i=1}^{m}\left[y^{(i)}\log\phi + (1-y^{(i)})\log(1-\phi)\right]$$

Maximizing the likelihood by setting the derivative (gradient) with respect to each of the parameters to 0.

$$\frac{\partial \ell}{\partial \phi} = \sum_{i=1}^{m} \left[\frac{y^{(i)}}{\phi} + \frac{1-y^{(i)}}{\phi - 1} \right] = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=1\}}{\phi} + \frac{m - \sum_{i=1}^{m} 1\{y^{(i)}=1\}}{\phi - 1} = 0,$$

We get the maximum likelihood estimates of the parameter ϕ. And for μ, by Matrix Book,

$$\begin{aligned}
\nabla_{\mu_1} \ell &= -\frac{1}{2} \sum_{y^{(i)}=1} \nabla_{\mu_1} (x^{(i)} - \mu_1)^T \Sigma^{-1} (x^{(i)} - \mu_1) \\
&= -\frac{1}{2} \sum_{y^{(i)}=1} \nabla_{\mu_1} \left(x^{(i)T} \Sigma^{-1} x^{(i)} - \mu_1^T \Sigma^{-1} x^{(i)} - x^{(i)T} \Sigma^{-1} \mu_1 + \mu_1^T \Sigma^{-1} \mu_1 \right) \\
&= -\frac{1}{2} \sum_{y^{(i)}=1} \left[2\Sigma^{-1} \mu_1 - 2\Sigma^{-1} x^{(i)} \right] = 0;
\end{aligned}$$

Then, we will get the results of μ_1 and μ_{-1} consistent with part(b).

The last step with respect to Σ^{-1}, denoted as $S = \Sigma^{-1}$ and $|S| = 1/|\Sigma|$,

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

while

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

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

set

$$\nabla_S \ell = \frac{1}{2} \sum_{i=1}^{m} \left[S^{-1} - b_i b_i^T \right] = 0$$

and we have

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

4. 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 $x^{(0)} = \overrightarrow{0}$. When the algorithm is run, it will produce a value of $x \in \mathbb{R}^n$ for each iteration: $x^{(1)}, x^{(2)}, \cdots$. 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 $z^{(0)} = \overrightarrow{0}$). If the values $z^{(1)}, z^{(2)}, \cdots$ 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) Show that Newton's method (applied to find the minimum of a function) is invariant to linear reparameterizations. Note that since $z^{(0)} = \overrightarrow{0} = A^{-1}x^{(0)}$, 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]

Taken from **SOLUTION MANUAL**.

First, we do $\nabla_z g(z)$,

$$\frac{\partial g(z)}{\partial z_i} = A_{*i}^T \nabla_x f(Az)$$
$$\nabla_z g(z) = A^T \nabla_x f(Az)$$

where A_{*i}^T means the i-th column of matrix A and $\nabla_x f(Az)$ is the vector of $\nabla_x f(x)$ evaluated at Az.

Then we do $\nabla_z^2 g(z)$,

$$H_{ij} = \frac{\partial^2 g(z)}{\partial z_i \partial z_j} = A_{*i}^T H_f(Az) A_{*j}$$
$$\nabla_z^2 g(z) = A^T H_f(Az) A = H_g(z)$$

And we have

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

So considering $Az^{(i)} = x^{(i)}$,

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

(b) Is gradient descent invariant to linear reparameterizations? Justify your answer.

No. We take

$$\begin{aligned} z^{(i+1)} &= z^{(i)} - \alpha \nabla_z g(z^{(i)}) \\ &= z^{(i)} - \alpha A^T \nabla_x f(Az^{(i)}) \\ Az^{(i+1)} &= Az^{(i)} - \alpha AA^T \nabla_x f(Az^{(i)}) \\ &= x^{(i)} - \alpha AA^T \nabla_x f(x^{(i)}) \end{aligned}$$

while

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

If and only if $AA^T = I$ it holds. That is to say A is orthogonal.

[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.

5. 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 $1\text{Å} = 10^{-10}$ meters.

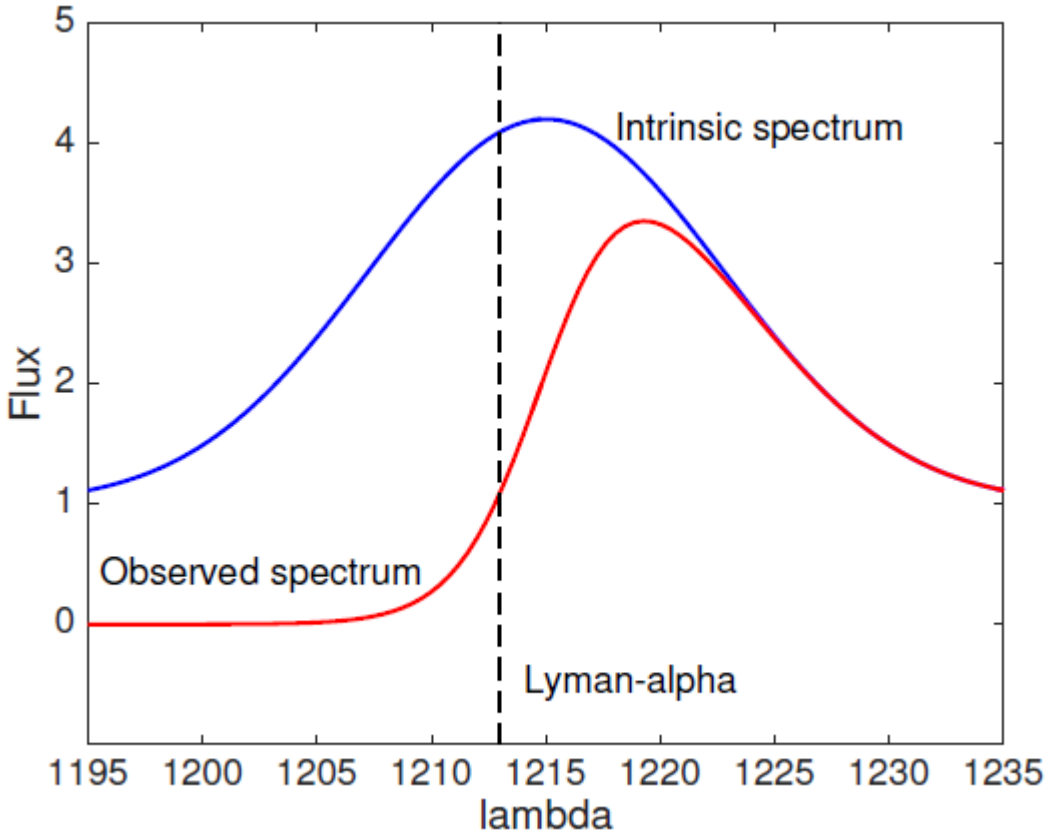

Figure 2: Light spectrum of a quasar. The blue line shows the intrinsic (i.e. original) flux spectrum emitted by the quasar. The red line denotes the observed spectrum here on Earth. To the left of the Lyman-α line, the observed flux is damped and the intrinsic (unabsorbed) flux continuum is not clearly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

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

[4]Ciollaro, Mattia, et al. "Functional regression for quasar spectra." arXiv:1404.3168 (2014).

hydrogen atom, which only occurs for certain wavelengths of light.) For wavelengths greater than this Lyman-α wavelength, the observed light spectrum fobs 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`

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: `load_quasar_data.m`

(a) 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. Show that $J(\theta)$ can also be written

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

for an appropriate diagonal matrix W, and where X and $\overrightarrow{y}$ are as defined in class. State clearly what W is.

Take $W_{ii} = \frac{1}{2}w^{(i)}$ and $W_{ij} = 0$ for $i \neq j$.

$$\begin{aligned}(X\theta - \overrightarrow{y})^T W (X\theta - \overrightarrow{y}) &= (X\theta - \overrightarrow{y})^T \frac{1}{2}\text{diag}(w^{(1)}, \cdots, w^{(n)})(X\theta - \overrightarrow{y}) \\ &= \frac{1}{2}\sum_{i=1}^{m} w^{(i)} \left(\theta^T x^{(i)} - y^{(i)}\right)^2 = J(\theta)\end{aligned}$$

[5] https://hea-www.harvard.edu/FOSAGN/

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

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

and that the value of θ that minimizes $J(\theta)$ is given by $(X^T X)^{-1} X^T \overrightarrow{y}$. 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 $\overrightarrow{y}$.

$$\begin{aligned}\nabla_\theta J(\theta) &= \nabla_\theta(\theta^T X^T W X\theta - y^T W X\theta - \theta^T X^T W y + y^T W y) \\ &= 2X^T W X\theta - 2X^T W y = 0, (W \text{ is symmetric }, W^T = W)\end{aligned}$$

Thus, $X^T W X\theta = X^T W^T y$, that is $\theta = (X^T W X)^{-1} X^T W y$

iii. Suppose we have a training set $\{(x^{(i)}, y^{(i)}); i = 1, \cdots, m\}$ 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.

We first make $\ell(\theta) = \log p(y^{(i)}|x^{(i)};\theta)$, maximizing $\ell(\theta)$

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

and we have $w^{(i)} = \frac{1}{(\sigma^{(i)})^2}$, and that is a weighted linear regression.

(b) Visualizing the data

i. 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.

See Figure 3.

ii. 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

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

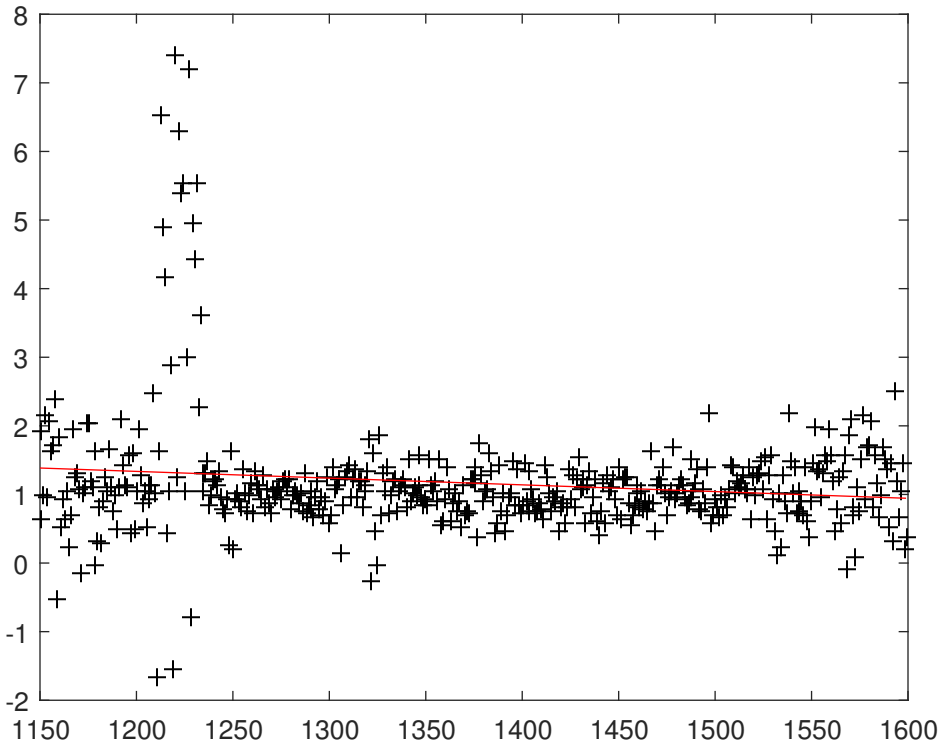

Figure 3: Regression $\theta = [2.5134, -0.0010]^T$.

with bandwidth parameter τ = 5.

See next part of Figure 4.

iii. Repeat (b)(ii) four more times with τ = 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.

See Figure 4.

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

Matlab code for this part:

```
function yhat = weighted_regression(XX, y, tau)
    m = length(y);
    X = [ones(m,1), XX];
    yhat = zeros(m,1);
    w = zeros(m,1);

    for i = 1:m
        w = exp(- (XX-XX(i)).^2 / (2*tau^2));
        W = diag(w);
        XWX = X'*W*X;
        XTWY = X'*W*y;
```

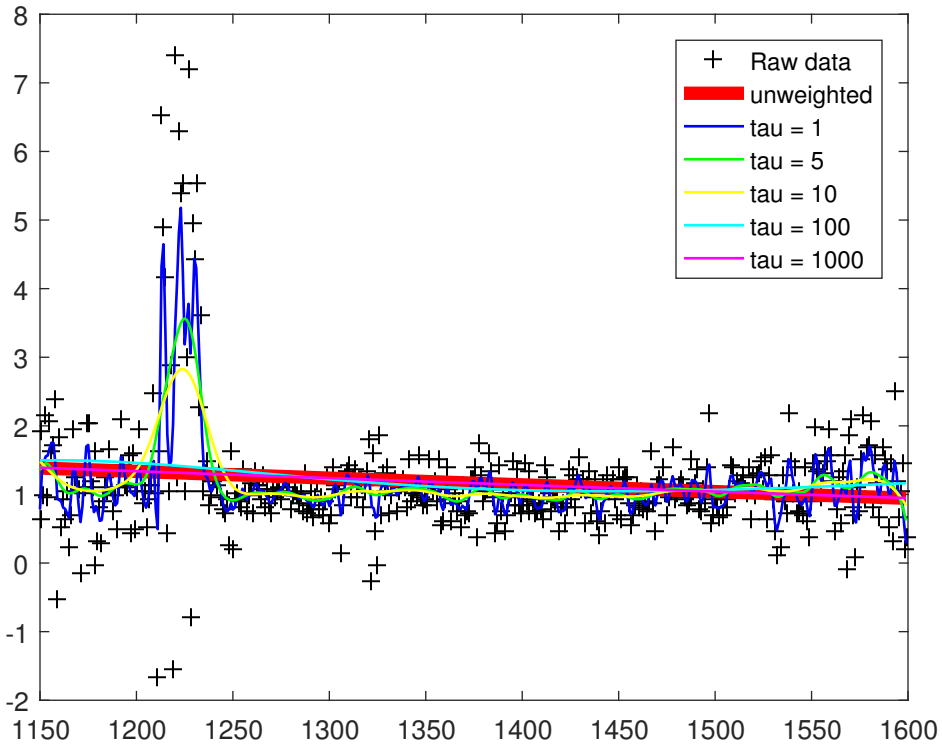

Figure 4: Varing τ.

```
        theta = pinv(XWX)*XTWY;
        yhat(i) = X(i, :)*theta;
    end

end
```

```
% load_quasar_data
%
% Loads the data in the quasar data files
%
% Upon completion of this script, the matrices and data are as follows:
%
% lambdas - A length n = 450 vector of wavelengths {1150, ..., 1599}
% train_qso - A size m-by-n matrix, where m = 200 and n = 450, of noisy
%      observed quasar spectra for training.
% test_qso - A size m-by-n matrix, where m = 50 and n = 450, of noisy observed
%       quasar spectra for testing.


%% part(b)

load quasar_train.csv;
lambdas = quasar_train(1, :)';
train_qso = quasar_train(2:end, :);

[m, n] = size(train_qso);
```

```
21
22 y = train_qso(1,:)';
23 X = [ones(n ,1), lambdas];
24
25 h = plot(lambdas, y, 'k+');
26 hold on;
27
28 theta = pinv(X'*X)*X'*y
29
30 h = plot(lambdas, theta(1)+theta(2)*lambdas, 'r-');
31 set(h, 'linewidth', 5);
32 hold on;
33
34 yhat = weighted_regression(lambdas, y, 1);
35 h = plot(lambdas, yhat, 'b-');
36 set(h, 'linewidth', 1);
37 hold on;
38
39 yhat = weighted_regression(lambdas, y, 5);
40 h = plot(lambdas, yhat, 'g-');
41 set(h, 'linewidth', 1);
42 hold on;
43
44 yhat = weighted_regression(lambdas, y, 10);
45 h = plot(lambdas, yhat, 'y-');
46 set(h, 'linewidth', 1);
47 hold on;
48
49 yhat = weighted_regression(lambdas, y, 100);
50 h = plot(lambdas, yhat, 'c-');
51 set(h, 'linewidth', 1);
52 hold on;
53
54 yhat = weighted_regression(lambdas, y, 1000);
55 h = plot(lambdas, yhat, 'm-');
56 set(h, 'linewidth', 1);
57 hold on;
58
59 h = legend('Raw data', 'unweighted', 'tau = 1', 'tau = 5', 'tau = 10', ...
60 'tau = 100', 'tau = 1000');
61
62 load quasar_test.csv;
63 test_qso = quasar_test(2:end, :);
```

(c) 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 $\lambda \in \{\lambda_1, \cdots, \lambda_n\}$, 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 $\lambda_1 = 1150, \cdots, \lambda_n = 1599$.

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

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

for λ in the Lyman-α forest.

i. First, we must smooth the data in the training data set to make it more useful for prediction. For each $i = 1, \cdots, m$, 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 τ = 5. Do the same for the test set. We will now operate on these smoothed spectra.

ii. Using your estimated regression functions $f^{(i)}$ for $i = 1, \cdots, m$, 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:

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

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

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

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 $\lambda \in \{1300, \cdots, 1599\}$ 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 $\texttt{neighb}_k(f_{\text{right}})$ denote the k indices $i \in \{1, \cdots, m\}$ that are closest to f_{right}, that is

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

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

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

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

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

Recall that $f^{(i)}_{\text{right}}$ 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 $j = 1, \cdots, m$, construct the estimator $\widehat{f_{\text{left}}}$ from (21) using $f_{\text{right}} = f^{(i)}_{\text{right}}$. Then compute the error $d(f^{(j)}_{\text{left}}, \widehat{f_{\text{left}}})$ between the true spectrum $f^{(j)}_{\text{left}}$ 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?

Average training error is 1.0709, a little different with the **SOLUTION MANUAL**...

iii. 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.

Average test error is 3.0937, a little different with the **SOLUTION MANUAL**...

Fitted $\widehat{f_{\text{left}}}$ curve:

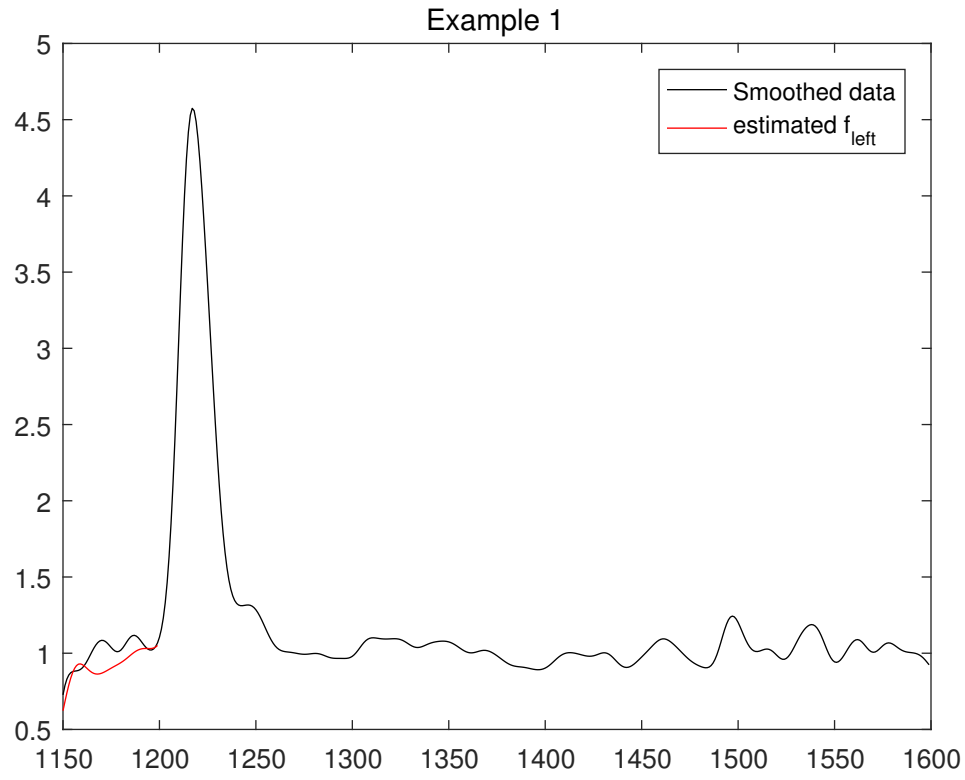

Figure 5: Resulting functional regression for test set example 1.

Matlab code for this part:

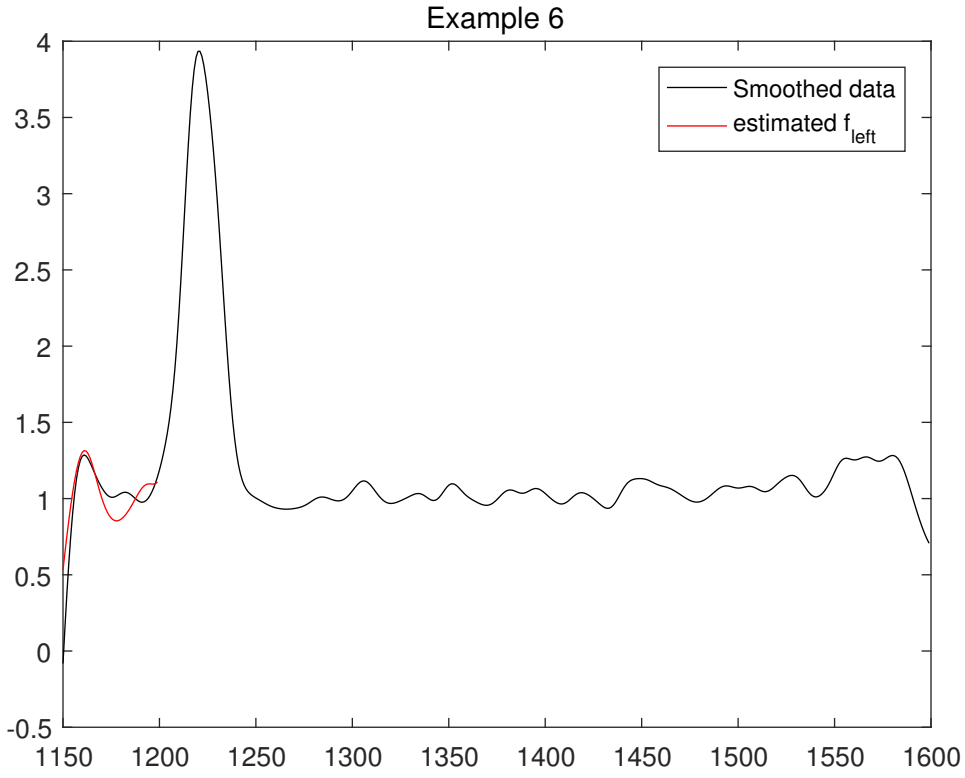

Figure 6: Resulting functional regression for test set example 6.

```
load quasar_train.csv;
lambdas = quasar_train(1, :)';
train_qso = quasar_train(2:end, :);

[m, n] = size(train_qso);

load quasar_test.csv;
test_qso = quasar_test(2:end, :);

mtest = size(test_qso,1);

train_smooth = zeros(size(train_qso));
test_smooth = zeros(size(test_qso));

% for i = 1:m
%     train_smooth(i,:) = (weighted_regression(lambdas, train_qso(i,:)', 5))';
% end
% for i = 1:mtest
%     test_smooth(i,:) = (weighted_regression(lambdas, test_qso(i,:)', 5))';
% end
%
% save('train.mat', 'train_smooth');
% save('test.mat', 'test_smooth');

load train.mat;
load test.mat;
```

```
27
28 train_left = train_smooth(:,1:50);
29 train_right = train_smooth(:, 150:end);
30 test_left = test_smooth(:,1:50);
31 test_right = test_smooth(:, 150:end);
32
33 %% calculate nearest neighbors
34 train_dist = zeros(m,m);
35 for i = 1:m
36     for j = i+1:m
37         diff = train_right(i,:)-train_right(j,:);
38         train_dist(i,j) = diff*diff';
39     end
40 end
41
42 train_dist = train_dist+train_dist';
43 train_dist = train_dist/max(train_dist(:));
44
45 nearest = 3;
46 fleft = zeros(m, 50);
47 for i = 1:m
48 %     [train_dist_sort, inds] = sort(train_dist(:, i), 1, 'ascend');
49 %     close_inds = ones(m, 1);
50 %     close_inds(inds(nearest+1:end)) = 0;
51 %     h = max(train_dist(:, i));
52 %     kerns = max(1-train_dist(:, i) /h, 0);
53 %     kerns = kerns .* close_inds;
54 %     fleft(i, :) = train_left'*kerns /sum(kerns);
55     [train_dist_sort, ind] = sort(train_dist(i,:), 2, 'ascend');
56     h = max(train_dist(i, :));
57     ker = zeros(m,1);
58     for j = 1:nearest
59         ker(ind(j)) = max(1-train_dist(i, ind(j))/h, 0);
60     end
61     fleft(i, :) = (train_left'*ker/sum(ker))';
62 end
63
64 %% error of part(c)(ii)
65 err = sum( (fleft(:)-train_left(:)).^2 )
66 errrate = err/m
67
68 %% test example 1 and 6
69 train_test_dist = zeros(mtest, m);
70 for i = 1:mtest
71     for j = 1:m
72         diff =train_right(j) - test_right(i);
73         train_test_dist(i, j) = diff*diff';
74     end
75 end
76 train_test_dist = train_test_dist/max(train_test_dist(:));
77 % train_test_dist = train_test_dist';
78
79 flefttest = zeros(mtest, 50);
80
81 for i=1:mtest
82     [train_test_dist_sort, ind] = sort(train_test_dist(i,:), 2, 'ascend');
83     h = max(train_test_dist(i, :));
84     ker = zeros(m,1);
85     for j = 1:nearest
```

```
        ker(ind(j)) = max(1-train_test_dist(i, ind(j))/h, 0);
    end
    flefttest(i, :) = (train_left'*ker/sum(ker))';
%     [train_dist_sort, inds] = sort(train_test_dist(:, i), 1, 'ascend');
%     close_inds = ones(m, 1);
%     close_inds(inds(nearest+1:end)) = 0;
%     h = max(train_test_dist(:, i));
%     kerns = max(1-train_test_dist(:, i) /h, 0);
%     kerns = kerns .* close_inds;
%     flefttest(i, :) = train_left'*kerns /sum(kerns);
end

err = sum( (flefttest(:)-test_left(:)).^2 )
errrate = err/mtest

%% plot pictures
figure;
h = plot(lambdas, test_smooth(1,:), 'k-');
hold on;
h = plot(lambdas(1:50), flefttest(1,:), 'r-');
h = legend('Smoothed data', 'estimated f_{left}');
h = title('Example 1');

figure;
h = plot(lambdas, test_smooth(6,:), 'k-');
hold on;
h = plot(lambdas(1:50), flefttest(6,:), 'r-');
h = legend('Smoothed data', 'estimated f_{left}');
h = title('Example 6');
```