Problem Set #1 : Supervised Learning

Omer Hazon

October 18, 2017

1 Logistic Regression

(a)

Evaluating $\boldsymbol{H} = (H_{ij}) = (\frac{\partial^2 J(\boldsymbol{\theta})}{\partial\theta_i \partial\theta_j})$ from $J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \log(1+\exp(-y^{(i)}\sum_\alpha \theta_\alpha x_\alpha^{(i)}))$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial\theta_j} = \frac{1}{m}\sum_{i=1}^{m} \frac{-y^{(i)}x_j^{(i)}\exp(-y^{(i)}\sum_\alpha \theta_\alpha x_\alpha^{(i)})}{1+\exp(-y^{(i)}\sum_\alpha \theta_\alpha x_\alpha^{(i)})} = \frac{1}{m}\sum_{i=1}^{m} \frac{-y^{(i)}x_j^{(i)}}{1+\exp(y^{(i)}\sum_\alpha \theta_\alpha x_\alpha^{(i)})}$$

$$\frac{\partial^2 J(\boldsymbol{\theta})}{\partial\theta_j \partial\theta_k} = \frac{1}{m}\sum_{i=1}^{m} y^{(i)}x_j^{(i)}(1+\exp(y^{(i)}\sum_\alpha \theta_\alpha x_\alpha^{(i)}))^{-2}\exp(y^{(i)}\sum_\alpha \theta_\alpha x_\alpha^{(i)})y^{(i)}x_k^{(i)} =$$

$$\frac{1}{m}\sum_{i=1}^{m} \frac{(y^{(i)})^2 x_j^{(i)}x_k^{(i)}\exp(y^{(i)}\sum_\alpha \theta_\alpha x_\alpha^{(i)})}{(1+\exp(y^{(i)}\sum_\alpha \theta_\alpha x_\alpha^{(i)}))^2} = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}x_k^{(i)}\frac{\exp(y^{(i)}\sum_\alpha \theta_\alpha x_\alpha^{(i)})}{(1+\exp(y^{(i)}\sum_\alpha \theta_\alpha x_\alpha^{(i)}))^2} = H_{jk}$$

And for any given vector $\boldsymbol{z} = (z_1, z_2, z_3, \ldots, z_n)$, using

$$\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} = \sum_{j,k} z_j H_{jk} z_k = \sum_{j,k} z_j z_k \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}x_k^{(i)}\frac{\exp(y^{(i)}\sum_\alpha \theta_\alpha x_\alpha^{(i)})}{(1+\exp(y^{(i)}\sum_\alpha \theta_\alpha x_\alpha^{(i)}))^2} =$$

$$= \frac{1}{m}\sum_{i=1}^{m} \frac{\exp(y^{(i)}\sum_\alpha \theta_\alpha x_\alpha^{(i)})}{(1+\exp(y^{(i)}\sum_\alpha \theta_\alpha x_\alpha^{(i)}))^2}\sum_{j,k} z_j z_k x_j^{(i)}x_k^{(i)} = \frac{1}{m}\sum_{i=1}^{m} \frac{\exp(y^{(i)}\sum_\alpha \theta_\alpha x_\alpha^{(i)})}{(1+\exp(y^{(i)}\sum_\alpha \theta_\alpha x_\alpha^{(i)}))^2}(\sum_j z_j x_j^{(i)})(\sum_j z_k x_k^{(i)}) =$$

$$= \frac{1}{m}\sum_{i=1}^{m} \frac{\exp(y^{(i)}\sum_\alpha \theta_\alpha x_\alpha^{(i)})}{(1+\exp(y^{(i)}\sum_\alpha \theta_\alpha x_\alpha^{(i)}))^2}(\sum_j z_j x_j^{(i)})^2$$

The final expression contains factors which are squares (and therefore ≥ 0) and exp which is positive. The total sum over i, or $\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z}$ for any $\boldsymbol{z}$, is therefore ≥ 0.

(b)

Using the formulae for the gradient and the hessian from (a), and the Newton's method formula $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \boldsymbol{H}^{-1}\nabla J(\boldsymbol{\theta})$, the algorithm converges onto

```
theta =
   -2.6205      0.7604      1.1719
```

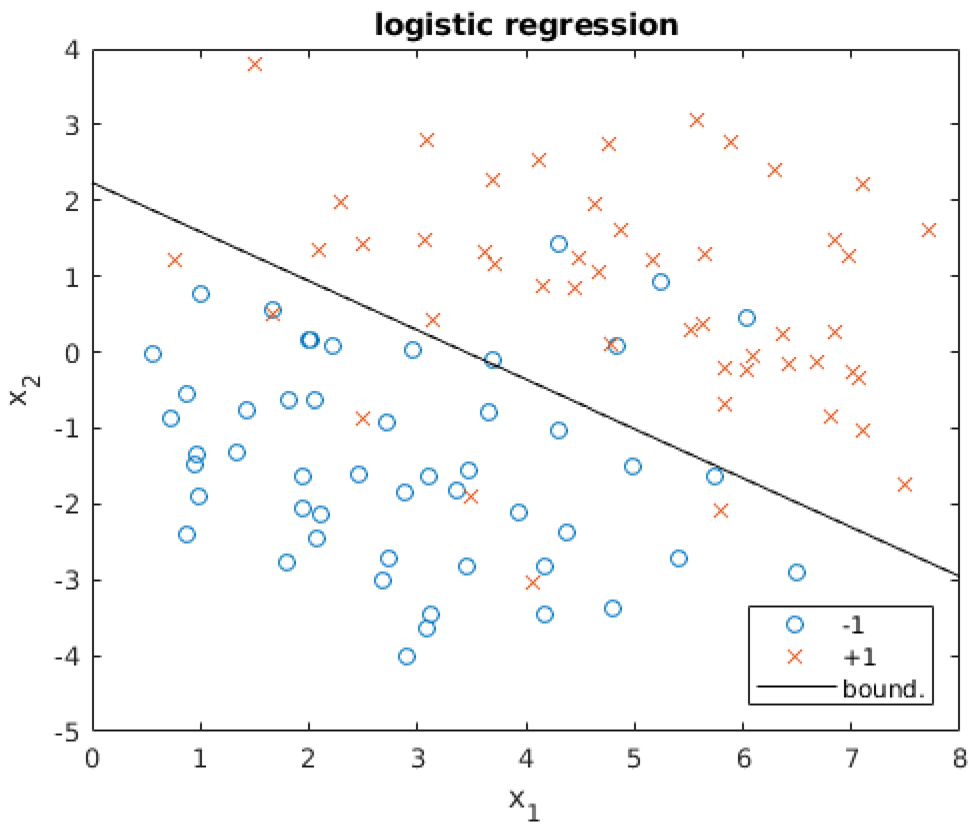
Where the first value is the intercept term, and the second and third correspond to x_1 and x_2 from the given dataset.

Using the code:

Listing 1: Code for logistic regression

```
%Load the data into arrays
X = load('logistic_x.txt');
Y = load('logistic_y.txt');

%Using N=2 but N+1 for the size of theta to account for an intercept term
N = 2;
M = length(X);

%Put in constant one feature into X
X = [ones(M,1), X];

%Initialize theta to be all zeros
theta = zeros(N+1,1);

%Allow sufficient amount of time for convergence
T=10;
for t=1:T
    H = 0;
    grad = 0;
    %Calculate values for hessian 'H' and gradient 'grad' using components
    %from each data point (x^(i), y^(i))
    for i = 1:M
        xi = X(i,:).';
        yi = Y(i);
        expo = exp(yi*theta.'*xi);
        H = H + expo/(1+expo)^2 * xi*(xi.')/M;
        grad = grad + (-yi*xi)/(1+expo)/M;
    end
    disp(t);
    disp(grad.');
    %Apply Netwon's method to theta
    theta = theta - H\grad;
end

%Plotting results
plot(X(Y==-1,2), X(Y==-1,3), 'o');
hold on
plot(X(Y==1,2), X(Y==1,3), 'x');
bound = refline(-theta(2)/theta(3), -theta(1)/theta(3));
bound.Color = 'k';
legend('-1', '+1', 'bound.', 'Location', 'SouthEast');
xlabel x_1
ylabel x_2
title 'logistic␣regression'
display(theta);
```

(c)



2 Poisson regression

(a)

The exponential family is defined as

$$P(y;\eta) = b(y)\exp(\eta T(y) - a(\eta))$$

And the Poisson distribution as

$$P(y;\lambda) = \frac{1}{y!}\exp(-\lambda)\lambda^y = (y!)^{-1}\exp(-\lambda)\exp(y\log\lambda) = (y!)^{-1}\exp(y\log\lambda - \lambda)$$

Define $\eta = \log\lambda$, $b(y) = (y!)^{-1}$, $T(y) = y$, $a(\eta) = \exp(\eta)$

Then Poisson is $P(y;\eta) = b(y)\exp(\eta T(y) - a(\eta))$. This shows that the Poisson distribution is in the exponential family.

(b)

To construct the GLM, we assume $y|x;\theta \sim P(y;\eta)$ as defined above for the Poisson distribution. The model θ predicts y given x with $h_\theta(x) = \boldsymbol{E}[y|x;\theta]$, and that $\eta = \theta^T x$.

Then, using the property of Poisson distributions whereby a parameter of λ leads to a mean of λ, we have $h_\theta(x) = \boldsymbol{E}[y|x;\theta] = \lambda = \exp(\eta) = \exp(\theta^T x)$. The canonical response function is the exponential.

(c)

$$\log p(y^{(i)}|x^{(i)};\theta) = \log\left((y^{(i)}!)^{-1}\exp(y^{(i)}\theta^T x^{(i)} - \exp(\theta^T x^{(i)}))\right) = -\log(y^{(i)}!) + y^{(i)}\theta^T x^{(i)} - \exp(\theta^T x^{(i)})$$

$$\nabla_\theta \log p(y^{(i)}|x^{(i)};\theta) = y^{(i)}x^{(i)} - \exp(\theta^T x^{(i)})x^{(i)} = \left(y^{(i)} - \exp(\theta^T x^{(i)})\right)x^{(i)}$$

Taking the j-th element in the gradient as $\partial/\partial\theta_j$:

$$\frac{\partial \log p(y^{(i)}|x^{(i)};\theta)}{\partial\theta_j} = \left(y^{(i)} - \exp(\theta^T x^{(i)})\right)x_j^{(i)}$$

And the gradient ascent rule:

$$\theta_j \leftarrow \theta_j + \alpha\left(y^{(i)} - \exp(\theta^T x^{(i)})\right)x_j^{(i)}$$

Expressed with the canonical response $h_\theta(x) = \exp(\theta^T x)$:

$$\theta_j \leftarrow \theta_j + \alpha\left(y^{(i)} - h_\theta(x^{(i)})\right)x_j^{(i)}$$

3 Gaussian Discriminant Analysis

(a)

Using $p(y|x) = \frac{p(x|y)p(y)}{p(x)}$:

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

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

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

Where :

$$\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1) - \frac{1}{2}(x-\mu_{-1})^T\Sigma^{-1}(x-\mu_{-1}) =$$

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

And since Σ is symmetric, with term cancellation,

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

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

And the original expression is:

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

The expression for $p(y=-1|x;\phi,\Sigma,\mu_{-1},\mu_1)$, will just be $1-p(y=1|x;\phi,\Sigma,\mu_{-1},\mu_1)$, and since the expression has a logistic form, it will just flip the sign in the exponential. Expressed with the variable y, the result is:

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

$$=\frac{1}{1+\exp\left(-y\left(\theta^T x+\theta_0\right)\right)}$$

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

(b)

$$l=\sum_{i=1}^m \log\frac{1}{(2\pi)^{1/2}}-\log\sigma-\frac{1}{2\sigma^2}(x^{(i)}-\mu_{y^{(i)}})^2+\log p(y^{(i)}|\phi)$$

Maximizing ϕ:
The relevant term is $\sum_{i=1}^m \log p(y^{(i)}|\phi)$

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

$$\frac{\partial p(y|\phi)}{\partial\phi}=1\{y=1\}-1\{y=-1\}=y$$

$$\frac{\partial l}{\partial\phi}=\sum_{i=1}^m\frac{\frac{\partial p(y|\phi)}{\partial\phi}}{p(y|\phi)}=\sum_{i=1}^m\frac{y^{(i)}}{1\{y^{(i)}=1\}\phi+1\{y^{(i)}=-1\}(1-\phi)}=0$$

$$\sum_{i=1}^m 1\{y^{(i)}=1\}/\phi-1\{y^{(i)}=-1\}/(1-\phi)=0$$

$$\left(\sum_{i=1}^m 1\{y^{(i)}=1\}\right)/\phi=\left(\sum_{i=1}^m 1\{y^{(i)}=-1\}\right)/(1-\phi)$$

$$\left(\sum_{i=1}^m 1\{y^{(i)}=1\}\right)(1-\phi)=\left(\sum_{i=1}^m 1\{y^{(i)}=-1\}\right)\phi$$

$$\left(\sum_{i=1}^m 1\{y^{(i)}=1\}\right)=\left(\sum_{i=1}^m 1\{y^{(i)}=-1\}+1\{y^{(i)}=1\}\right)\phi=m\phi$$

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

Maximizing σ:
The revelant terms are $\sum_{i=1}^m -\log\sigma-\frac{1}{2\sigma^2}(x^{(i)}-\mu_{y^{(i)}})^2$

$$\frac{\partial l}{\partial\sigma}=\sum_{i=1}^m -\frac{1}{\sigma}+\frac{(x^{(i)}-\mu_{y^{(i)}})^2}{\sigma^3}=0$$

$$\sum_{i=1}^m -\sigma^2+(x^{(i)}-\mu_{y^{(i)}})^2=0$$

$$m\sigma^2 = \sum_{i=1}^{m}(x^{(i)} - \mu_{y^{(i)}})^2$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu_{y^{(i)}})^2$$

Maximizing μ_α where $\alpha \in \{1, -1\}$:
The relevant term is $\sum_{i=1}^{m} -\frac{1}{2\sigma^2}(x^{(i)} - \mu_{y^{(i)}})^2$

$$\frac{\partial l}{\partial \mu_\alpha} = \sum_{i=1}^{m} \frac{1}{\sigma^2}(x^{(i)} - \mu_{y^{(i)}})1\{y^{(i)} = \alpha\} = 0$$

$$\sum_{i=1}^{m}(x^{(i)}1\{y^{(i)} = \alpha\} - \mu_{y^{(i)}}1\{y^{(i)} = \alpha\}) = \sum_{i=1}^{m}(x^{(i)}1\{y^{(i)} = \alpha\} - \mu_\alpha 1\{y^{(i)} = \alpha\}) = 0$$

$$\mu_\alpha \sum_{i=1}^{m} 1\{y^{(i)} = \alpha\} = \sum_{i=1}^{m} x^{(i)}1\{y^{(i)} = \alpha\}$$

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

4 Linear invariance of optimization algorithms

(a)

The update rule in Newton's method is:

$$x \leftarrow x - H^{-1}\nabla f(x)$$

Given an $x^{(i)}$ it will be updated to $x^{(i+1)} = x^{(i)} - H_x^{-1}|_{x^{(i)}}\nabla_x f(x^{(i)})$.
Using $g(z) = f(Az) = f(x(z))$, then minimizing g wrt z looks like

$$z^{(i+1)} = z^{(i)} - H_z^{-1}|_{z^{(i)}}\nabla_z g(z^{(i)})$$

Matrix Calculus Identities with Proofs Use the notation $f(\boldsymbol{x})$ as short for $f(x_1, x_2, ..., x_N)$

Identity for Gradient: The gradient of g wrt $\boldsymbol{z}$ in terms of the gradient of f wrt $\boldsymbol{x}$ is $\boldsymbol{\nabla_z} g(\boldsymbol{z}) = \boldsymbol{A}^T \boldsymbol{\nabla_x} f|_{\boldsymbol{x}=\boldsymbol{Az}}$

Proof: Given

$$f(\boldsymbol{x}) = f(\boldsymbol{x}(\boldsymbol{z})) = g(\boldsymbol{z})$$

$$x_i = \sum_k A_{ik}z_k$$

We have the partials of f: $\partial f/\partial x_i$; and of g: $\partial g/\partial z_j$
To express the partials of g in terms of the partials of f, use the multivariate chain rule:

$$[\boldsymbol{\nabla_z} g]_j = \frac{\partial g}{\partial z_j} = \sum_i \frac{\partial f}{\partial x_i}(\boldsymbol{x}(\boldsymbol{z}))\frac{\partial x_i}{\partial z_j} = \sum_i \frac{\partial f}{\partial x_i}(\boldsymbol{x}(\boldsymbol{z}))\frac{\partial\left(\sum_k A_{ik}z_k\right)}{\partial z_j} =$$

$$= \sum_i \frac{\partial f}{\partial x_i}(\boldsymbol{x}(\boldsymbol{z})) \sum_k A_{ik}\delta_{kj} = \sum_i \frac{\partial f}{\partial x_i}(\boldsymbol{x}(\boldsymbol{z}))A_{ij} = \sum_i (A^T)_{ji} \frac{\partial f}{\partial x_i}(\boldsymbol{x}(\boldsymbol{z})) = \left[A^T \nabla_x f(\boldsymbol{x}(\boldsymbol{z}))\right]_j =$$

$$= \left[\boldsymbol{A}^T \boldsymbol{\nabla}_{\boldsymbol{x}} f|_{\boldsymbol{x}=\boldsymbol{A}\boldsymbol{z}}\right]_j$$

Proof end.

Identity for Hessian: The hessian of g wrt $\boldsymbol{z}$ in terms of the hessian of f wrt $\boldsymbol{x}$ is $\boldsymbol{H}_{g(\boldsymbol{z})}(\boldsymbol{z}) = \boldsymbol{A}^T \boldsymbol{H}_{f(\boldsymbol{x})}|_{\boldsymbol{x}=\boldsymbol{A}\boldsymbol{z}} \boldsymbol{A}$

Proof: Using the result from the previous proof: $\frac{\partial g}{\partial z_j} = \sum_i \frac{\partial f}{\partial x_i}(\boldsymbol{x}(\boldsymbol{z}))A_{ij}$

$$\left[\boldsymbol{H}_{g(\boldsymbol{z})}\right]_{jl} = \frac{\partial}{\partial z_l}\left(\frac{\partial g}{\partial z_j}\right) = \frac{\partial}{\partial z_l}\left(\sum_i \frac{\partial f}{\partial x_i}(\boldsymbol{x}(\boldsymbol{z}))A_{ij}\right) = \sum_i A_{ij} \frac{\partial}{\partial z_l}\left(\frac{\partial f}{\partial x_i}(\boldsymbol{x}(\boldsymbol{z}))\right) =$$

$$\sum_i A_{ij} \sum_p \frac{\partial^2 f}{\partial x_i \partial x_p}(\boldsymbol{x}(\boldsymbol{z})) \frac{\partial x_p}{\partial z_l} = \sum_i A_{ij} \sum_p \frac{\partial^2 f}{\partial x_i \partial x_p}(\boldsymbol{x}(\boldsymbol{z})) \frac{\partial(\sum_q A_{pq} z_q)}{\partial z_l} =$$

$$\sum_i A_{ij} \sum_p \frac{\partial^2 f}{\partial x_i \partial x_p}(\boldsymbol{x}(\boldsymbol{z})) \sum_q A_{pq}\delta_{ql} = \sum_i A_{ij} \sum_p \frac{\partial^2 f}{\partial x_i \partial x_p}(\boldsymbol{x}(\boldsymbol{z}))A_{pl} = \sum_i \sum_p \left[\boldsymbol{A}^T\right]_{ji} \left[\boldsymbol{H}_{f(\boldsymbol{x})}|_{\boldsymbol{x}=\boldsymbol{A}\boldsymbol{z}}\right]_{ip} \left[\boldsymbol{A}\right]_{pl} =$$

$$= \left[\boldsymbol{A}^T \boldsymbol{H}_{f(\boldsymbol{x})}|_{\boldsymbol{x}=\boldsymbol{A}\boldsymbol{z}} \boldsymbol{A}\right]_{jl}$$

Proof end.
Given $\boldsymbol{z}^{(i)} = \boldsymbol{A}^{-1}\boldsymbol{x}^{(i)}$:
Then

$$\boldsymbol{z}^{(i+1)} = \boldsymbol{z}^{(i)} - \boldsymbol{H}^{-1}_{g(\boldsymbol{z})}|_{\boldsymbol{z}=\boldsymbol{z}^{(i)}} \boldsymbol{\nabla}_{\boldsymbol{z}} g(\boldsymbol{z}^{(i)}) =$$

$$= \boldsymbol{z}^{(i)} - \left(\boldsymbol{A}^T \boldsymbol{H}_{f(\boldsymbol{x})}|_{\boldsymbol{x}=\boldsymbol{A}\boldsymbol{z}^{(i)}} \boldsymbol{A}\right)^{-1} \left(\boldsymbol{A}^T \boldsymbol{\nabla}_{\boldsymbol{x}} f|_{\boldsymbol{x}=\boldsymbol{A}\boldsymbol{z}^{(i)}}\right) =$$

$$= \boldsymbol{z}^{(i)} - \boldsymbol{A}^{-1} \boldsymbol{H}^{-1}_{f(\boldsymbol{x})}|_{\boldsymbol{x}=\boldsymbol{A}\boldsymbol{z}^{(i)}} \boldsymbol{A}^{-T} \boldsymbol{A}^T \boldsymbol{\nabla}_{\boldsymbol{x}} f|_{\boldsymbol{x}=\boldsymbol{A}\boldsymbol{z}^{(i)}} =$$

$$= \boldsymbol{A}^{-1}\boldsymbol{x}^{(i)} - \boldsymbol{A}^{-1} \boldsymbol{H}^{-1}_{f(\boldsymbol{x})}|_{\boldsymbol{x}=\boldsymbol{x}^{(i)}} \boldsymbol{\nabla}_{\boldsymbol{x}} f|_{\boldsymbol{x}=\boldsymbol{x}^{(i)}} = \boldsymbol{A}^{-1}\left(\boldsymbol{x}^{(i)} - \boldsymbol{H}^{-1}_{f(\boldsymbol{x})}|_{\boldsymbol{x}=\boldsymbol{x}^{(i)}} \boldsymbol{\nabla}_{\boldsymbol{x}} f|_{\boldsymbol{x}=\boldsymbol{x}^{(i)}}\right) =$$

$$= \boldsymbol{A}^{-1}\boldsymbol{x}^{(i+1)}$$

And Newton's method is invariant to linear reparametrizations.

(b)

Using the gradient descent rule:

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

And with $z^{(i)} = A^{-1}x^{(i)}$

$$z^{(i+1)} = z^{(i)} - \alpha \nabla_z g(z^{(i)}) = z^{(i)} - \alpha \boldsymbol{A}^T \boldsymbol{\nabla}_{\boldsymbol{x}} f|_{\boldsymbol{x}=\boldsymbol{A}\boldsymbol{z}^{(i)}} = A^{-1}x^{(i)} - A^T \alpha \nabla_x f(x^{(i)})$$

Gradient descent is not invariant to linear parametrization, unless the matrix A has the property that $A^{-1} = A^T$, which is not true in general.

5 Regression for denoising quasar spectra

(a)

(i)

Let $\boldsymbol{y} = [y^{(1)}; y^{(2)}; ...; y^{(m)}]$ (a column vector of size m), $\boldsymbol{X} = [x_0^{(1)}, x_1^{(1)}, ..., x_n^{(1)}; x_0^{(2)}, x_1^{(2)}, ..., x_n^{(2)}; ...; x_0^{(m)}, x_1^{(m)}, ..., x_n^{(m)}]$ (an m by (n+1) matrix), and $\boldsymbol{\theta} = [\theta_0; \theta_1; ...; \theta_n]$ (a column vector of size n+1).

Then

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

$$= \sum_{i=1}^{m} [\boldsymbol{X\theta} - \boldsymbol{y}]^{(i)} \frac{w^{(i)}}{2} [\boldsymbol{X\theta} - \boldsymbol{y}]^{(i)}$$

Given a vector $[a_1; a_2; ...; a_k]$ and $[b_1; b_2; ...; b_k]$, the vector $[a_1b_1; a_2b_2; ...; a_kb_k]$ can be constructed by multiplying

$$\operatorname{diag}\left(\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_k \end{bmatrix}\right)\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_k \end{bmatrix} = \begin{bmatrix} a_1 & 0 & 0 & \cdots & 0 \\ 0 & a_2 & 0 & \cdots & 0 \\ 0 & 0 & a_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_k \end{bmatrix}\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_k \end{bmatrix} = \begin{bmatrix} a_1b_1 \\ a_2b_2 \\ a_3b_3 \\ \vdots \\ a_kb_k \end{bmatrix}$$

Hence

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{m} [\boldsymbol{X\theta} - \boldsymbol{y}]^{(i)} \left[\boldsymbol{W}[\boldsymbol{X\theta} - \boldsymbol{y}]\right]^{(i)} = (\boldsymbol{X\theta} - \boldsymbol{y})^T \boldsymbol{W} (\boldsymbol{X\theta} - \boldsymbol{y})$$

Where $\boldsymbol{W} = \frac{1}{2}\operatorname{diag}\left([w^{(1)}; w^{(2)}; ...; w^{(m)}]\right)$

(ii)

$$\frac{\partial}{\partial\theta_l}J(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{m} w^{(i)}2(\boldsymbol{\theta}^T\boldsymbol{x}^{(i)} - y^{(i)})\frac{\partial}{\partial\theta_l}\left(\sum_j \theta_j x_j^{(i)}\right) = \sum_{i=1}^{m} w^{(i)}(\boldsymbol{\theta}^T\boldsymbol{x}^{(i)} - y^{(i)})x_l^{(i)}$$

$$\nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta}) = \sum_{i=1}^{m} w^{(i)}(\boldsymbol{\theta}^T\boldsymbol{x}^{(i)} - y^{(i)})\boldsymbol{x}^{(i)} = 2(\boldsymbol{X\theta} - \boldsymbol{y})^T\boldsymbol{WX} = 0$$

$$(\boldsymbol{X\theta} - \boldsymbol{y})^T(\boldsymbol{WX}) = 0 \implies (\boldsymbol{WX})^T(\boldsymbol{X\theta} - \boldsymbol{y}) = 0 \implies \boldsymbol{X}^T\boldsymbol{WX\theta} = \boldsymbol{X}^T\boldsymbol{Wy}$$

The value of $\boldsymbol{\theta}$ that minimizes $J(\boldsymbol{\theta})$ is $(\boldsymbol{X}^T\boldsymbol{WX})^{-1}\boldsymbol{X}^T\boldsymbol{Wy}$

(iii)

The log likelihood is:

$$L(\boldsymbol{y}|\boldsymbol{X};\boldsymbol{\theta}) = \sum_{i=1}^{m} L(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta}) = \sum_{i=1}^{m} -\log\left(\sqrt{2\pi}\sigma^{(i)}\right) - \frac{\left(\boldsymbol{y}^{(i)} - \boldsymbol{\theta}^T\boldsymbol{x}^{(i)}\right)^2}{2\left(\sigma^{(i)}\right)^2}$$

Then maximizing L with respect to $\boldsymbol{\theta}$ is equivalent to minimizing $J(\boldsymbol{\theta})$ where

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

Where $w^{(i)} = 1/(\sigma^{(i)})^2$, which is the same as locally weighted linear regression given the values of w in terms of σ.

(b)

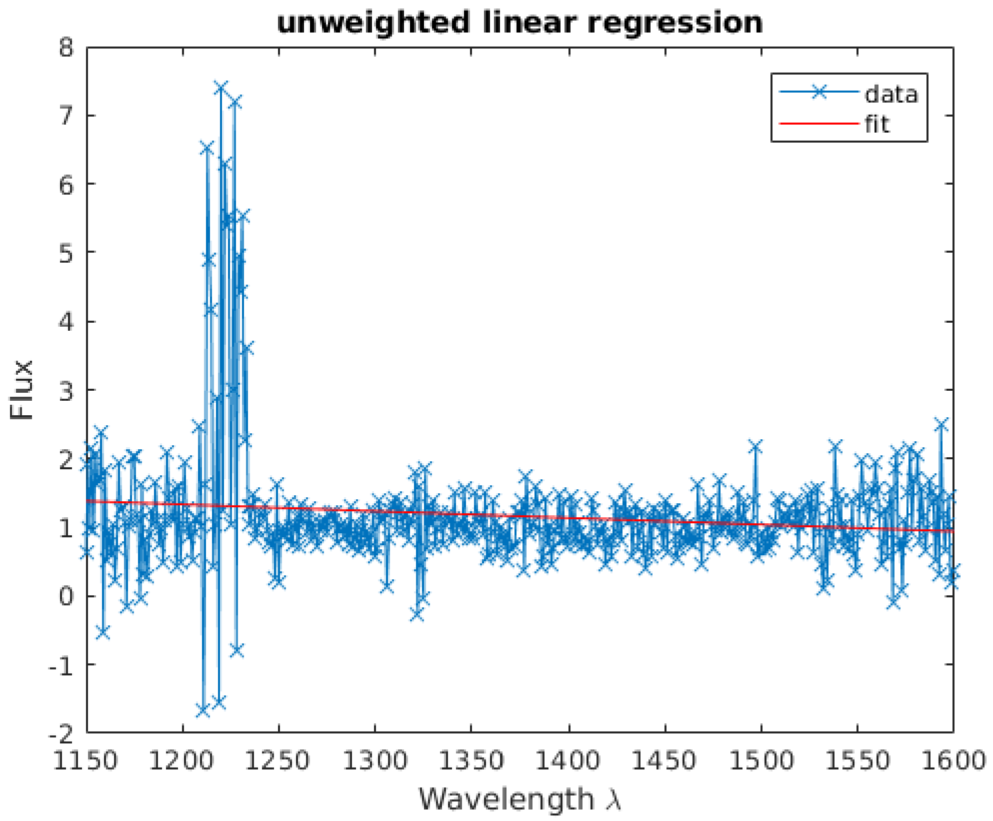
(i)

Used the normal equations:

$$\boldsymbol{\theta} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

To find (intercept first):

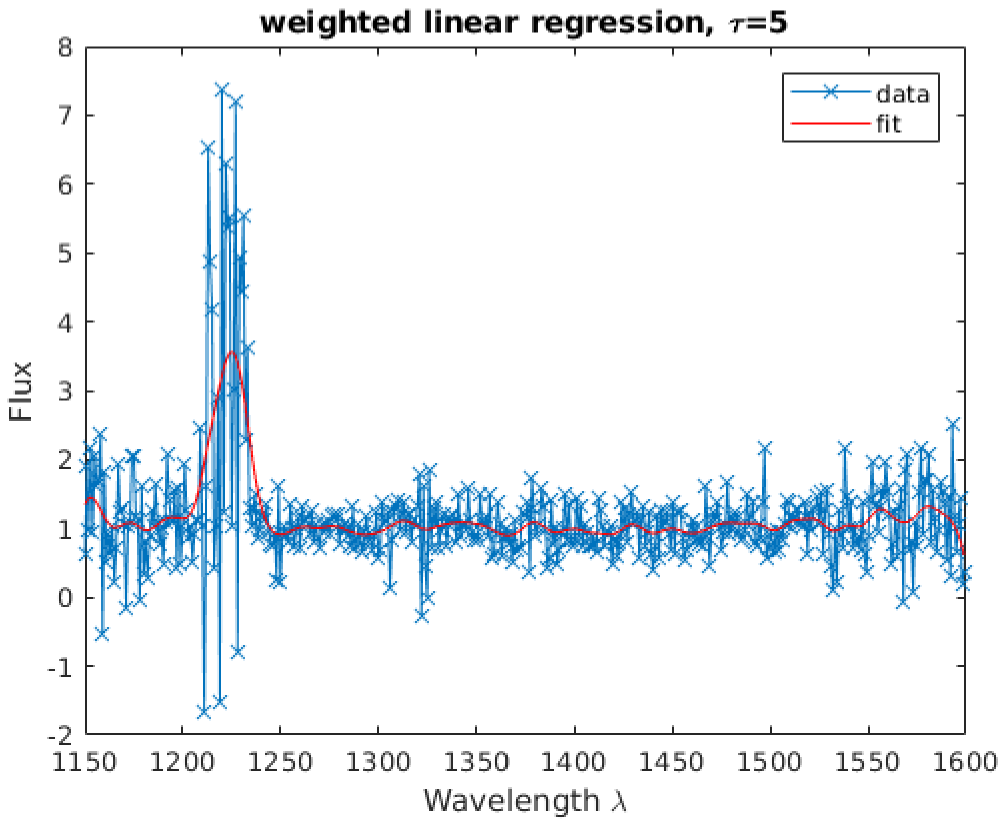
$$\boldsymbol{\theta} = [2.5134; -0.0010]$$



```
load_quasar_data
%first training example
y = train_qso(1,:).';
X = [ones(size(lambdas)), lambdas];
theta = (X.'*X)\(X.'*y);
disp(theta);

%Plotting results
figure
plot(lambdas, y, '-x');
hold on;
ref = refline(flipud(theta));
ref.Color = 'r';
legend data fit
xlabel 'Wavelength␣\lambda'
ylabel 'Flux'
title 'unweighted␣linear␣regression'
```

(ii)

The resulting fit, with $\tau = 5$ is as follows (implemented with the code below):



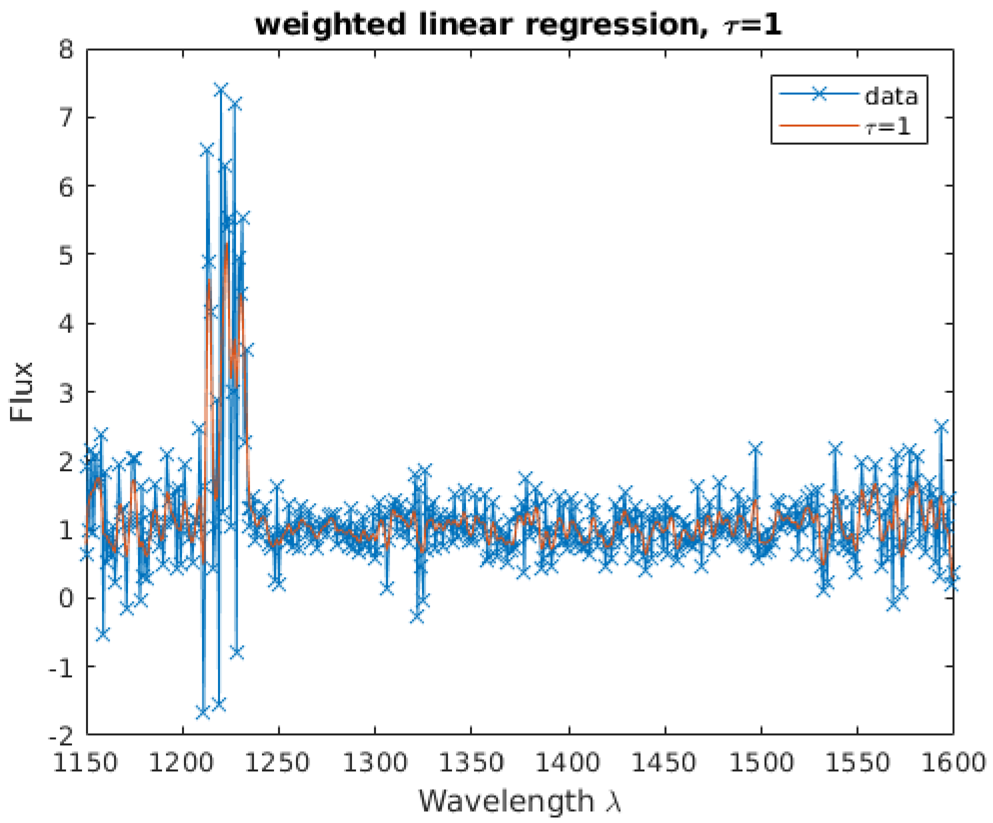
```
load_quasar_data
%first training example
y = train_qso(1,:).';
X = [ones(size(lambdas)), lambdas];

m = length(y);
n = 2;

thetas = zeros(n,m);
tau = 5;
%Different weights per value of lambda
for i=1:m
    x = lambdas(i);
    ws = exp(-(x-lambdas).^2/(2*tau^2));
    W = 1/2*diag(ws);
    thetas(:,i) = (X.'*W*X)\(X.'*W*y);
end

%Evaluating the fit for each value of lambda
y_fit = zeros(1,m);
for i=1:m
    y_fit(i) = thetas(:,i).'*X(i,:).';
end

%Plotting results
figure
```

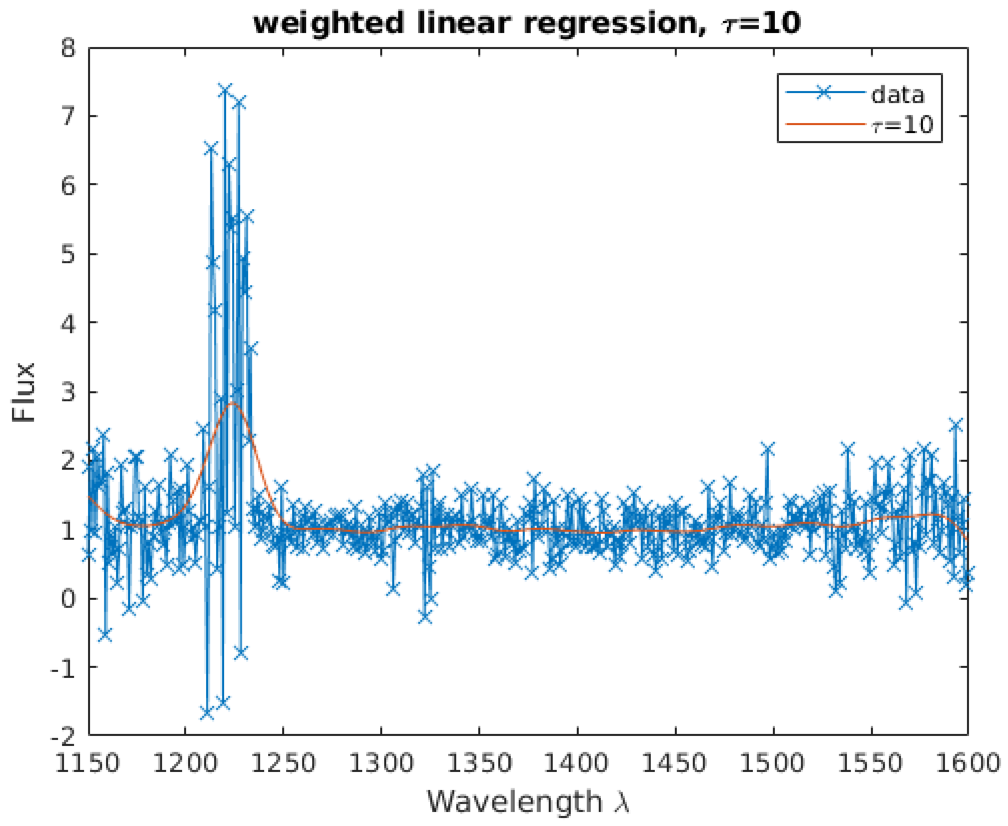
```
plot(lambdas, y, '-x');
hold on
plot(lambdas, y_fit, 'r');
legend data fit
xlabel 'Wavelength␣\lambda'
ylabel 'Flux'
title 'weighted␣linear␣regression,␣\tau=5'
```

(iii)

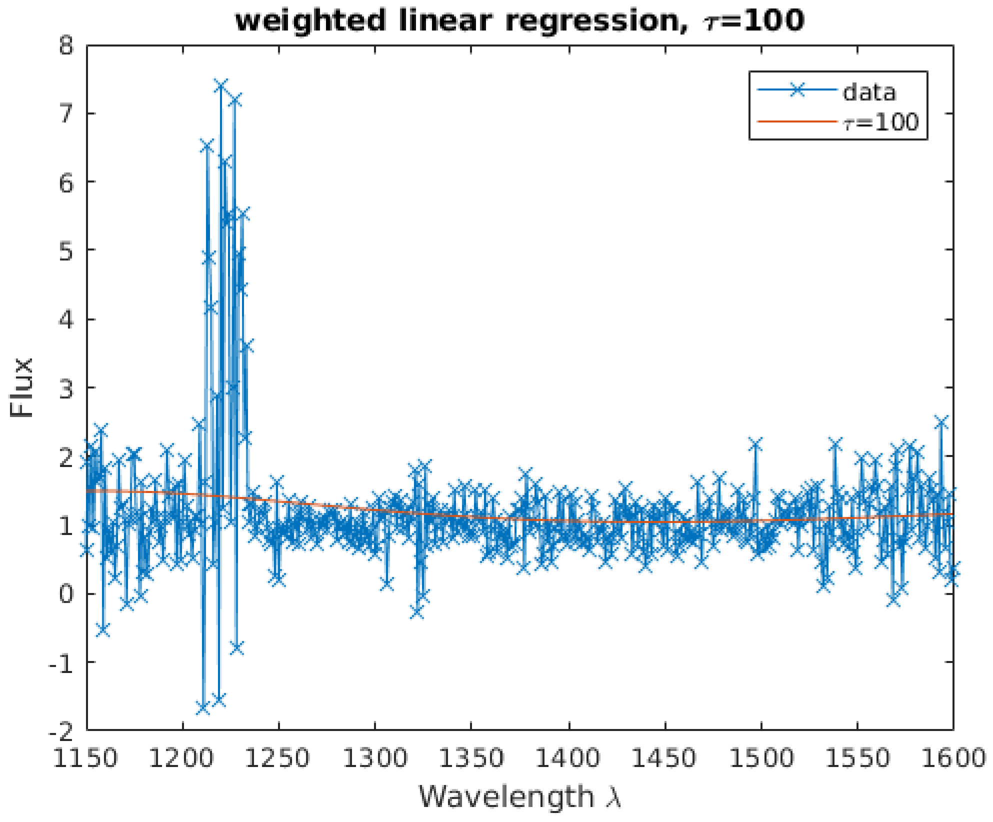
The locally weighted linear regression with a small value of τ such as 1, only applies a small amount of smoothing but the shape of the curve closely tracks the individual data points, resulting in too much remaining noise. As τ is increased more and more points in the vicinity of each data point are considered to determine the shape of the local fit and this results in removing the random fluctuations, but can also underestimate the size of a narrow peak. As τ is further increased the curvature of the fit decreases more and more until it is indistinguishable from a nonweighted linear fit.

The resulting fits for various values of τ are as follows:

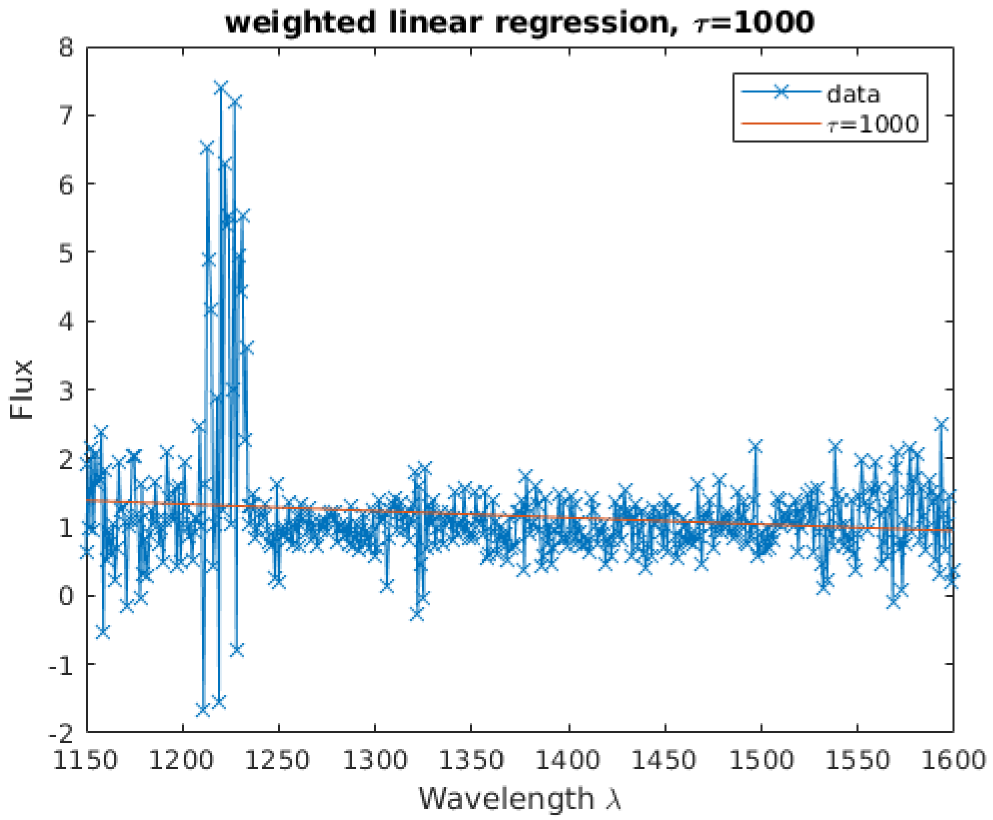




weighted linear regression, τ=10
Flux
Wavelength λ
data
τ=10



weighted linear regression, τ=100
Flux
Wavelength λ
data
τ=100



```
load_quasar_data
%first training example
y = train_qso(1,:).';
X = [ones(size(lambdas)), lambdas];

m = length(y);
n = 2;

thetas = zeros(n,m);
for tau = [1 10 100 1000]
%Different weights per value of lambda
for i=1:m
    x = lambdas(i);
    ws = exp(-(x-lambdas).^2/(2*tau^2));
    W = 1/2*diag(ws);
    thetas(:,i) = (X.'*W*X)\(X.'*W*y);
end

%Evaluating the fit for each value of lambda
y_fit = zeros(1,m);
for i=1:m
    y_fit(i) = thetas(:,i).'*X(i,:).';
end

figure
plot(lambdas, y, '-x');
hold on
plot(lambdas, y_fit);
legend_message = sprintf('\\tau=%d', tau);
legend('data', legend_message);
```

```
xlabel 'Wavelength␣\lambda'
ylabel 'Flux'
title(sprintf('weighted␣linear␣regression,␣%s', legend_message));
end
```

(c)

(i)

The dataset was smoothed with the following function applied to all rows of the training and testing arrays:

```
function f_smooth = smooth_f(f,lambdas,tau)
%SMOOTH_F Applies locally weighted linear regression to smooth f
% operates on f as a row vector
y = f.';
X = [ones(size(lambdas)), lambdas];

m = length(y);
n = 2;

thetas = zeros(n,m);

%Different weights per value of lambda
for i=1:m
    x = lambdas(i);
    ws = exp(-(x-lambdas).^2/(2*tau^2));
    W = 1/2*diag(ws);
    thetas(:,i) = (X.'*W*X)\(X.'*W*y);
end

%Evaluating the fit for each value of lambda
y_fit = zeros(1,m);
for i=1:m
    y_fit(i) = thetas(:,i).'*X(i,:).';
end

f_smooth = y_fit.';
end
```

Applied as such:

```
load_quasar_data

[m,~] = size(train_qso);
train_qso_smooth = zeros(size(train_qso));
for i = 1:m
    train_qso_smooth(i,:) = smooth_f(train_qso(i,:), lambdas, 5);
end

[m,~] = size(test_qso);
test_qso_smooth = zeros(size(test_qso));
for i = 1:m
    test_qso_smooth(i,:) = smooth_f(test_qso(i,:), lambdas, 5);
end
```

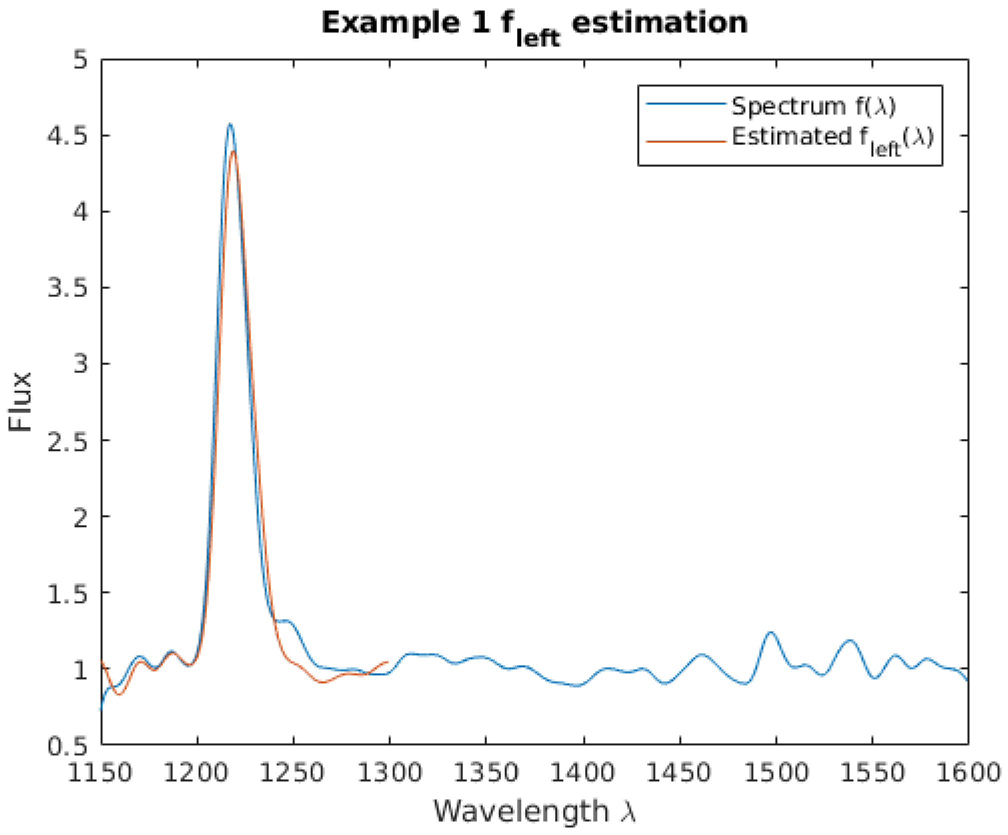
(ii)

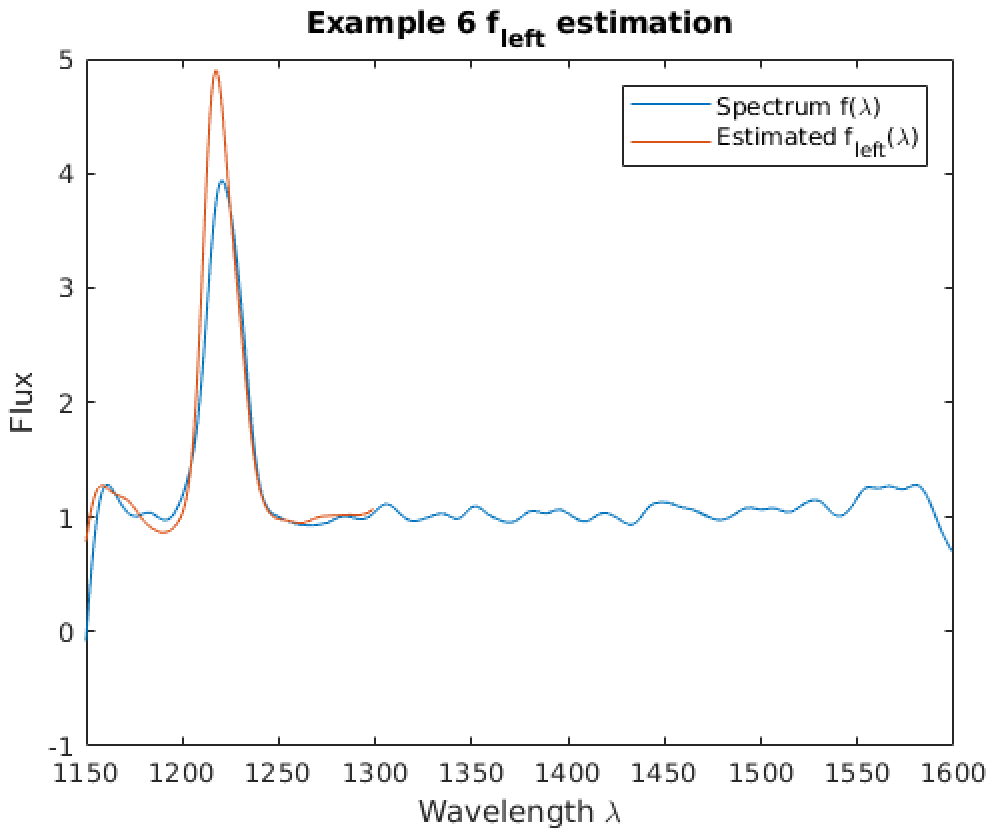
The average training error is 7.5119

(iii)

The average testing error is 16.6916

The following are the estimated spectra for examples 1 and 6 from the testing set:





The code for calculating the estimated spectra and average train/test errors is as follows:

```
%functional regression

%smooth_all %already ran
cutoff_lambda = 1300;
k = 3;
[m, ~] = size(train_qso_smooth);

fleft_trains = train_qso_smooth(:,lambdas < cutoff_lambda);
fright_trains = train_qso_smooth(:,lambdas >= cutoff_lambda);


fleft_tests = test_qso_smooth(:,lambdas < cutoff_lambda);
fright_tests = test_qso_smooth(:,lambdas >= cutoff_lambda);

disp('Average␣error␣for␣training␣set:');
disp(average_error(fleft_trains, fright_trains, fleft_trains, fright_trains, k));

disp('Average␣error␣for␣testing␣set:');
disp(average_error(fleft_tests, fright_tests, fleft_trains, fright_trains, k));

%plotting
for example = [1 6]
    figure
    plot(lambdas, test_qso_smooth(example,:));
    hold on
    fleft_hat = estimate_fleft(fright_tests(example,:), fleft_trains, fright_trains, k);
    plot(lambdas(lambdas < cutoff_lambda), fleft_hat);
    xlabel 'Wavelength␣\lambda'
    ylabel 'Flux'
    legend 'Spectrum␣f(\lambda)' 'Estimated␣f_{left}(\lambda)'
    title(sprintf('Example␣%d␣f_{left}␣estimation', example));
end

%The distance between two functions
function dd = d(f1, f2)
dd = sum((f1-f2).^2);
end

%The auxillary function defined in the problem set
function kk = ker(t)
kk = max(1-t,zeros(size(t)));
end

%Find the maximal distance from function f present in the training set
function hh = h(f, fright_trains)
[m, ~] = size(fright_trains);
hh = 0;
for i=1:m
    dist = d(f, fright_trains(i,:));
    if dist > hh
        hh = dist;
    end
end
end

%Find the k nearest neighbors (may include f) from f in the training set
function nn = neighb(k, f, fright_trains)
[m, ~] = size(fright_trains);
dists = zeros(m,1);
for i=1:m
    dists(i) = d(f, fright_trains(i,:));
end
[~, neighbor_inds] = sort(dists);
nn = neighbor_inds(1:k);
end

%Estimate fleft from fright by the weighted average from the k nearest
%neighbors where the weights are 1 - the distance to the neighbor divided
%by the maximal distance to any function in the training set
function fleft_hat = estimate_fleft(fright, fleft_trains, fright_trains, k)
nn = neighb(k, fright, fright_trains);
hh = h(fright, fright_trains);
numer = 0;
denom = 0;
for ind=nn.'
    fright_ind = fright_trains(ind,:);
    fleft_ind = fleft_trains(ind,:);
    numer = numer + fleft_ind * ker(d(fright, fright_ind)/hh);
    denom = denom + ker(d(fright, fright_ind)/hh);
end
fleft_hat = numer/denom;
end

%Estimate values for fleft from the frights and find the distance from the
%true fleft, then return the average such distance over the given set. This
%function can be used to find the average training error and the average
%testing error
```

```
function avgerr = average_error(flefts, frights, fleft_trains, fright_trains, k)
[m, ~] = size(flefts);
avgerr = 0;
for i=1:m
    fleft_hat = estimate_fleft(frights(i,:), fleft_trains, fright_trains, k);
    avgerr = avgerr + d(flefts(i,:), fleft_hat)/m;
end
end
```