CS 229 - PS 1: Problem 5

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 1 shows an example of a quasar light spectrum, where the wavelengths are measured in Angstroms (Å), where $1Å = 10^{-10}$ meters.

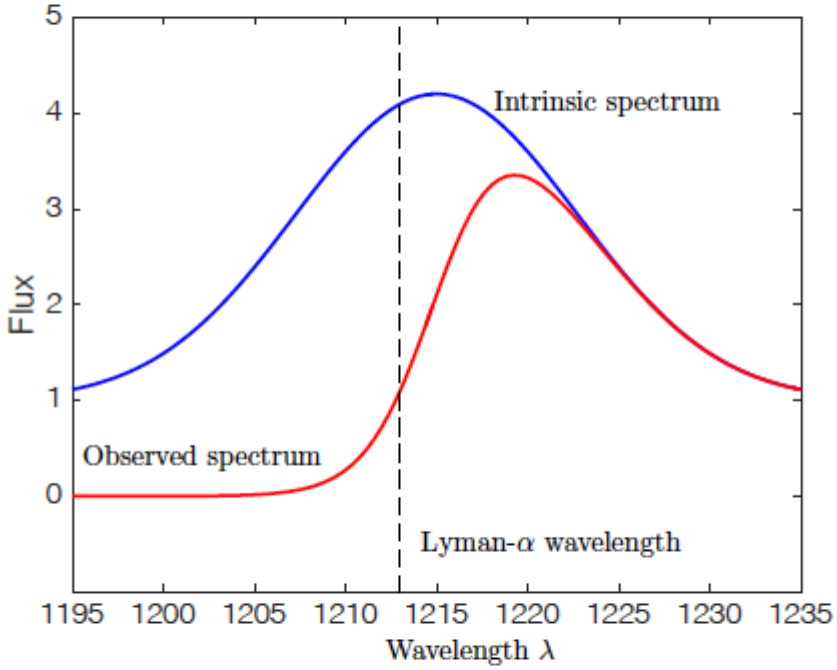


Figure 1: Light spectrum of a quasar. The blue line shows the intrinsic (i.e. original) flux spectrum emitted by the quasar. The red line denotes the observed spectrum here on Earth. To the left of the Lyman-α line, the observed flux is damped and the intrinsic (unabsorbed) flux continuum is not clearly 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 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`

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`

Part (a)

Part (a) essentially asks us to show that it can be written as follows and to derive the analytical solution to locally weighted linear regression. My solution is as follows:

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

$$W_{ii} = \frac{w^{(i)}}{2} \text{ and } W_{jk} = 0 \text{ if } j \neq k$$

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

In [25]:

```
# imports
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
```

Part (b)

(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. Remember the intercept term (your optimal θ should lie in $\mathbb{R}^2$).

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 $\tau = 5$.

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.

In [3]:

```
#load data
data = pd.read_csv('quasar_train.csv', header = None)
data
```

Out[3]:

	0	1	2	3	4	5	6	7	
0	1150.000	1151.000	1152.000	1153.000	1154.000	1155.000	1156.000	1157.000	1158.00
1	0.629	1.910	0.976	2.161	0.964	2.068	1.630	1.716	2.37
2	-0.161	1.830	0.609	1.932	0.932	0.857	0.978	1.032	1.79
3	-0.085	-1.304	0.691	-0.800	-1.090	1.087	0.734	1.198	1.12
4	1.283	-0.487	0.660	0.167	-1.217	1.535	2.264	0.220	0.86
...	...	...	...	...	...	...	...	...	
196	0.541	1.128	0.309	1.549	2.142	0.573	0.861	-0.064	0.94
197	1.237	0.712	1.407	1.155	-0.538	-1.360	2.321	1.598	1.44
198	0.212	0.102	0.452	2.887	3.236	0.674	-1.290	0.954	0.79
199	0.193	1.063	0.292	0.699	-1.152	1.738	1.987	1.434	3.07
200	0.142	0.591	3.317	2.280	0.973	-0.702	0.935	0.605	0.15

201 rows × 450 columns

In this case, our x is the flux and y is the corresponding wavelength given in the first row

In [10]:

```
n=450
```

In [34]:

```
data_nump = data.to_numpy()
x = data_nump[0] #select first row
x = np.expand_dims(x,1)
i = np.ones((450,1)) #add col of 1's to it
x = np.concatenate((i,x),1)
#x
```

In [35]:

```
y = data_nump[1]
```

Analytic Solution to Unweighted Linear Regression

$$(X^T X)^{-1} X^T \vec{y}.$$

In [36]:

```
xt = np.transpose(x)
theta = np.matmul(np.matmul(np.linalg.inv(np.matmul(xt,x)),xt),y)
theta
```

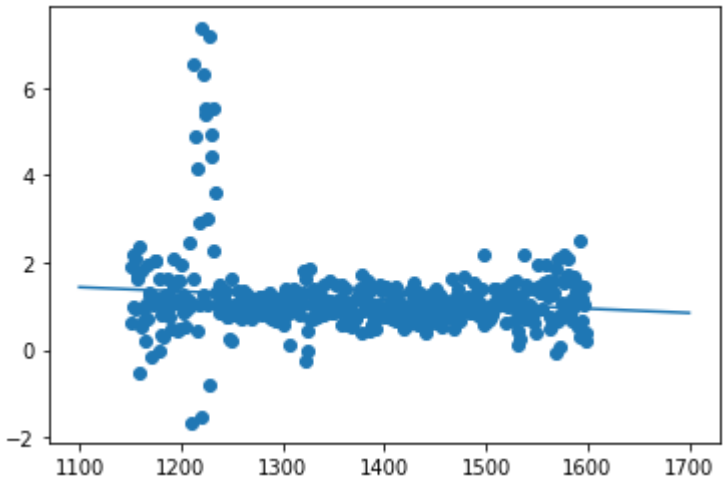
Out[36]:

```
array([ 2.51339906e+00, -9.81122145e-04])
```

Plot of Unweighted Linear Regression Solution and Data

In [46]:

```
#plot line
xvals = np.arange(1100,1700,0.01)
yvals = xvals*(theta[1]) + theta[0] #linear equation with parameters found
plt.scatter(xt[1],y)
plt.plot(xvals, yvals)
plt.show()
```



Analytic Solution to Weighted Linear Regression

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

where

$$W_{ii} = \frac{w^{(i)}}{2} \text{ and } W_{jk} = 0 \text{ if } j \neq k$$

and

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

Steps:

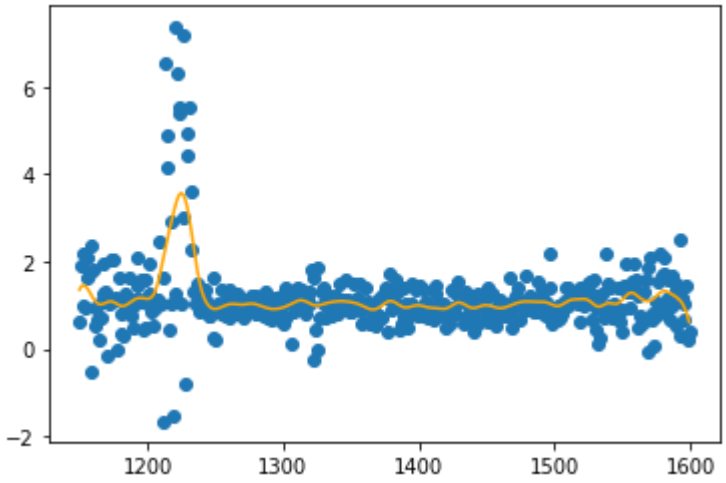
1. For each x in range that I want to draw curve for, compute theta a. Compute W and then, solve for theta
2. Use theta to predict y

In [114]:

```
#Clean this up

tau = 5 #change me
xvals = np.arange(1150,1600,1)
yvals = []
for row in range(xvals.shape[0]):
    yvals.append(calc_y(xvals[row],tau))

plt.scatter(xt[1],y)
plt.plot(xvals, yvals,color='orange')
plt.show()
```



In [117]:

```
#Calculate y for an x
def calc_y(x_,tau,y=y):
    t = calc_theta(x_,tau,y)
    return np.dot(t,np.array([1,x_]))
```

In [118]:

```
#Calculates theta
def calc_theta(x_,tau,y=y):
    w = calc_w(x_,tau)
    a = np.linalg.inv(np.matmul(np.matmul(xt,w),x))
    b = np.matmul(xt,np.matmul(w,y))
    return np.matmul(a,b)
```

In [107]:

```
#Calculate W
def calc_weight(x_,tau,x_i):
    return math.exp((-1*((x_-x_i)**2))/(2*(tau**2)))
def calc_w(x_,tau):
    w_ = np.zeros((n,n))
    for i in range(n):
        w_[i][i] = calc_weight(x_,tau,x[i][1])/2
    return w_
```

Part (c)

(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} (note that f_{left} and f_{right} don't cover the entire spectrum). 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_{right} to f_{left}:

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

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. Apply smoothing to the entire spectrums (including both f_{left} and f_{right}) for both train and test. We will now operate on these smoothed spectra.

Steps

1. Go through each row of flux values
2. For each, calculate the estimate at the x value based on weighted linear regression with $\tau = 5$
3. Do this for both the train & test set

In [122]:

```
smooth_train = np.zeros((data_nump.shape[0]-1,n))
for row in range(1,data_nump.shape[0]):
    tau = 5
    for i in range(n):
        smooth_train[row-1][i] = calc_y(x[i][1],tau,data_nump[row])
smooth_train
```

Out[122]:

```
array([[ 1.35457751,  1.411428  ,  1.44418291, ...,  0.83468706,
         0.73537353,  0.61880777],
       [ 0.8630299 ,  0.93499234,  0.98748083, ...,  0.88094242,
         0.7530859 ,  0.58686922],
       [-0.54199083, -0.3879195 , -0.23759803, ...,  1.1474556 ,
         1.18788252,  1.22661737],
       ...,
       [ 0.84491185,  0.91335579,  0.94612793, ...,  0.96263275,
         0.89165294,  0.80694689],
       [ 0.2365519 ,  0.40504492,  0.5759311 , ...,  1.06663278,
         1.0357148 ,  0.99772697],
       [ 1.25215608,  1.21257714,  1.15577569, ...,  0.59479467,
         0.37155891,  0.09915474]])
```

In [129]:

```
data = pd.read_csv('quasar_test.csv', header = None)
test_nump = data.to_numpy()
```

In [130]:

```
smooth_test = np.zeros((test_nump.shape[0]-1,n))
for row in range(1,test_nump.shape[0]):
    tau = 5
    for i in range(n):
        smooth_train[row-1][i] = calc_y(x[i][1],tau,test_nump[row])
smooth_test
```

Out[130]:

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [131]:

```
smooth_test.shape
```

Out[131]:

```
(50, 450)
```

C.ii)

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 $i \in \{1, 2, \ldots, m\}$ of the training set 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

$$\widehat{f_{\text{left}}}(\lambda) = \frac{\sum_{i \in \text{neighb}_k(f_{\text{right}})} \ker(d(f_{\text{right}}^{(i)}, f_{\text{right}})/h) f_{\text{left}}^{(i)}(\lambda)}{\sum_{i \in \text{neighb}_k(f_{\text{right}})} \ker(d(f_{\text{right}}^{(i)}, f_{\text{right}})/h)}. \quad (1)$$

Include f_{right} from training in its own neighborhood. Recall that $f_{\text{right}}^{(i)}$ is the *smoothed* (weighted linear regression) estimate of the ith training spectrum.
Construct the functional regression estimate (1) 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 (1) 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?

Steps

1. Define the relevant functions
2. Split data into right and left
3. Construct estimates f_left^ using f_right^ and of size f_left
4. Compute average error between estimated f_left^ and actual f_left

In [132]:

```
def d(f1,f2,right=True):
    s = 0
    k = 1300 if right else 1100 #if f_right, 1300-1599 only
    for i in range(1600-k):
        s+= (f1[i]-f2[i])**2
    return s
```

In [133]:

```
def ker(t):
    return np.maximium(1-t,0)
```

In [142]:

```
#k nearest neighbors to f & h
def neighb(f_right,f,k=3):
    ds = []
    for i in range(n):
        ds.append(d(f_right,f[i])) #add all distancs
    ds = np.array(ds)
    nn = []
    h = np.argmax(ds) #max of all
    for i in range(k):
        new_n = np.argmax(ds) #max
        nn.append(new_n) #add the index to list
        ds = np.delete(ds,new_n) #drop new_nth entry in ds
    return (np.array(nn),h)
```

If you are wondering where the rest of this problem's solution is, I think that I have run out of interest. I regret to not complete it, but if this were a real problem set, I think I would be mostly satisfied with the grade it gets me.

I may come back and finish this for clarity