CS 542, Spring 2014

Assignment 2

Shan Sikdar

1 3.3

Find an expression for the solution w* that minimizes this error function. Give two alternative interpretations of the weighted sum-of-errors function in terms of (i) data dependant noise and (ii) replicated data points.

Take the new sum -of-squares error function:
$E_D(w) = \frac{1}{2}\sum_{n=1}^{N} r_n\{t_n - w^T\phi(x_n)\}^2$
$\nabla E_D(w) = \sum_{n=1}^{N} r_n\{t_n - w^T\phi(x_n)\}\phi(x_n)^T$
$0 = \sum_{n=1}^{N} r_n t_n\phi(x_n)^T - \sum_{n=1}^{N} w^T\phi(x_n)\ \phi(x_n)^T$
$\sum_{n=1}^{N} w^T\phi(x_n)\ \phi(x_n)^T = \sum_{n=1}^{N} r_n t_n\phi(x_n)^T$
Since $\phi = (\phi_0, \phi_1...., \phi_{m-1})^T$:
$\sum_{n=1}^{N} w^T(\phi_0, \phi_1...., \phi_{m-1})((\phi_0, \phi_1 m...., \phi_{m-1})^T)^T = \sum_{n=1}^{N} r_n t_n\phi(x_n)^T$
Now $r_1, ..., r_n$ can be represented by a diagnol matrix where the coefficents make up the diaganol and everywhere else is 0.
Then using that fact along with the fact that summing $((\phi_0, \phi_1 m...., \phi_{m-1})^T)^T$ from 1 to N gives you Φ you get:
$w^T\Phi^T R\Phi = \sum_{n=1}^{N} r_n t_n\phi(x_n)^T$
$w^T\Phi^T R\Phi = \sum_{n=1}^{N} r_n t_n((\phi_0, \phi_1 m...., \phi_{m-1})^T)^T$
$w^T = \Phi^T Rt$
$w^T = \Phi^{-1}R^{-1}(\Phi^T)^{-1}\Phi^T Rt$
$w^T = (R\Phi)^{-1}(\Phi^T)^{-1}\Phi^T Rt$
$w^* = (\Phi^T R\Phi)^{-1}\Phi^T Rt$
(i) In terms of data dependent noise variation r_n can be seen as an inverse varianve parameter to a datapoint (x_n, t_n) which modifies the precision matrix.

(ii) replicated data points: r_n can be regarded as an effective number of replicated observations of a data point (x_n, t_n).

2 3.11

Show that the uncertainty $\sigma_N(x)$ associtated with linear regression function given by (3.59) statisfies $\sigma_{N+1}(x) \leq \sigma_N(x)$
Fromula 3.59: $\sigma_N^2(x) = \frac{1}{\beta} + \phi(x)^T S_N \phi(x)$
The formula for 3.59 for N+1 case:
$\sigma_{N+1}^2(x) = \frac{1}{\beta} + \phi(x)^T S_{N+1} \phi(x)$
It turns out from problem 3.8 the posterior distribution is given by 3.49 but with S_N replaced with S_{N+1} and m_n replaced with m_{n+1}. So:
$S_{N+1}^{-1} = S_N^{-1} + \beta \phi_{N+1} \phi_{N+1}^T$
$S_{N+1} = (S_N^{-1} + \beta \phi_{N+1} \phi_{N+1}^T)^{-1}$
Then using the matrix identity from appendix C: where $M = S_N^{-1}$ and $v = \beta^{\frac{1}{2}} \phi_{N+1}$:
$S_{N+1} = S_N - \frac{(S_N \phi_{N+1} \beta^{\frac{1}{2}})(\beta^{\frac{1}{2}} \phi_{N+1}^T S_N)}{1+\beta \phi_{N+1} S_N \phi_{N+1}}$
$S_{N+1} = S_N - \frac{\beta (S_N \phi_{N+1})(\phi_{N+1}^T S_N)}{1+\beta \phi_{N+1} S_N \phi_{N+1}}$
So plugging this value of S_{N+1} back into formula 3.59:
$\sigma_{N+1}^2(x) = \frac{1}{\beta} + \phi(x)^T (S_N - \frac{\beta (S_N \phi_{N+1})(\phi_{N+1}^T S_N)}{1+\beta \phi_{N+1} S_N \phi_{N+1}}) \phi(x)$
$\sigma_{N+1}^2(x) = \frac{1}{\beta} + \phi(x)^T S_N \phi(x) - \frac{\phi(x)^T \beta (S_N \phi_{N+1})(\phi_{N+1}^T S_N) \phi(x)}{1+\beta \phi_{N+1} S_N \phi_{N+1}}$
Using the definition of σ_N^2 from above:
$\sigma_{N+1}^2(x) = \sigma_N^2 - \frac{\phi(x)^T \beta (S_N \phi_{N+1})(\phi_{N+1}^T S_N) \phi(x)}{1+\beta \phi_{N+1} S_N \phi_{N+1}}$
Now if the fraction on the left is always positive then we are done. Since S_N is postive definite, the numerator and demoniator will both be positive. Therefore:
$\sigma_{N+1}(x) \leq \sigma_N(x)$

3 3.14

Show for $\alpha = 0$, the equivelent kernel cna be writtn as $k(x, x') = \psi(x)^T \psi(x')$
Using equation 3.54, when $\alpha = 0$, $S_N^{-1} = \beta \Phi^T \Phi$
Since $\psi_j(x)$ is our new othronormal basis passning the same space, there must be some function that can transform the original basis, to the orthonormal one. So in other words:
$\psi(x) = V\phi(x)$
where V is the matrix that represents the function that transforms our original basis to the new one. (similar to gram-schmidt?). Since the result of the transformation cover the same space, there must be another function that takes the orthonormal one back to the original space. Therefore this matrix must be the inverse of V. Then V has an inverse.
Before I can work with wquation 3.54, I first need to transform Φ into the new orthonormal basis so:
$\Phi V^T = \Psi$ (using transpose since multiplying on left instead of right of Φ)
Solving back for Φ, $\Phi = \Psi(V^T)^{-1}$:
So going back to equation 3.54:
$S_N^{-1} = \beta \Phi^T \Phi$
$S_N = (\beta \Phi^T \Phi)^{-1}$
$S_N = \beta^{-1}(\Phi^T \Phi)^{-1}$
$= \beta^{-1}(((\Psi(V^T)^{-1})^T(\Psi(V^T)^{-1}))^{-1}$
$= \beta^{-1}(\Psi(V^T)^{-1})^{-1}(((\Psi(V^T)^{-1})^T)^{-1}$
$= \beta^{-1}((V^T)^{-1})^{-1}\Psi^{-1})(((V^T)^{-1})^T\Psi^T)^{-1}$
$= \beta^{-1}((V^T))\Psi^{-1})(((V^T)^{-1})^T\Psi^T)^{-1}$
$= \beta^{-1}((V^T))\Psi^{-1})(((\Psi^T)^{-1}((V^T)^{-1})^T)^{-1}$
$= \beta^{-1}((V^T))\Psi^{-1})(\Psi^T)^{-1}V) = \beta^{-1}((V^T))V)$
(Ψ is an ortho basis, transpose and inverse will negate each other.)
$= \beta^{-1}((V^T))V)$
Plugging in S_N into the equivelent kernel formula:
$K(x, x') = \beta\phi(x)^T S_N \phi(x') = \beta\phi(x)^T \beta^{-1}((V^T))V)\phi(x') = \phi(x)^T((V^T))V)\phi(x')$
Now applying ϕ to the linear transformation V and V^T:
$= \psi(x)^T \psi(x')$
Now looking at the summation of the equivelence kernels:
$\sum_{n=1}^N k(x, x_n) = \sum_{n=1}^N \psi(x)^T \psi(x_n) = \sum_{n=1}^N \sum_{i=1}^M \psi_i(x)^T \psi_i(x_n)$
Using 3.115: $= \sum_{i=1}^M \psi_i(x_n)\psi_i(x_n) = 1$

4 3.21

Prove 3.117 and then make use of 3.117 to derive 3.92 starting drom 3.86.
Let A be a real, symetric matrix A.
By definition eigen values and vectors define how a vetor under the transformation of A can be recreated by simply multiplying the original vector by some constant λ_i(the eigenvalues). Let $\{u_i\}$ be a set of orthonormal vectors then we know be defintion of eignvalues/vectors:
$Au_i = \lambda_i u_i$. Then we can also define A by its eigenvalues.
So:
$ln|A| = ln\Pi_{i=1}^{M}\lambda_i = \sum_{i=1}^{M} ln\lambda_i$
Let α be some random optimized value:

$\frac{\delta ln|A|}{\delta\alpha} = \sum_{i=1}^{M} \frac{1}{\lambda_i}\frac{\delta\lambda_i}{\delta\alpha}$
Now show that the right hand side of equation 3.117 is equal to this.
From C.45 and C.46 in the Appendix on matrices, given eigenvalues we can recreate matrices and their inverse with their eigenvalues:
$A = \sum_{i=1}^{M} \lambda_i u_i u_i^T$ (c.45)
$A^{-1} = \sum_{i=1}^{M} \frac{1}{\lambda_i} u_i u_i^T$(c.46)
Taking the derivative of A using the product rule:
$\frac{\delta}{\delta\alpha}A = \sum_{i=1}^{M} \frac{\delta\lambda_i}{\delta\alpha} u_i u_i^T + \lambda_i(\frac{\delta u_i}{\delta\alpha}u_i^T + u_i \frac{\delta u_i}{\delta\alpha}^T)$
If the length of u_i is always constant this implies the derivative of a vector is orthognal to the vector . Then the multiplication u_i and $(\frac{\delta u_i}{\delta\alpha})$ will be zero. ("Courtesy of http://www.physicsforums.com/showthread.php?t=523876" and some calc III). So the last term cancels out leaving:
$\frac{\delta}{\delta\alpha}A = \sum_{i=1}^{M} \frac{\delta\lambda_i}{\delta\alpha} u_i u_i^T$
Now plugging in A^{-1}, $\frac{\delta}{\delta\alpha}A$ ino the right hand side of 3.117:
$Tr(A^{-1}\frac{\delta}{\delta\alpha}A) = Tr(\sum_{i=1}^{M} \frac{1}{\lambda_i} u_i u_i^T \sum_{j=1}^{M} \frac{\delta\lambda_j}{\delta\alpha} u_j u_j^T)$
$= tr(\sum_{i=1}^{M}\sum_{j=1}^{M} \frac{1}{\lambda_i}\frac{\delta\lambda_j}{\delta\alpha} u_i u_i^T u_j u_j^T)$
Now using the fact that the multipcation of orthognal vectors will be 0 leaves: $Tr(\frac{1}{\lambda_i}\frac{\delta\lambda_j}{\delta\alpha}\sum_{i=1}^{M} u_i u_i^T)$
But $\sum_{i=1}^{M} u_i u_i^T = I$ So we we are left with: $Tr((\frac{1}{\lambda_i}\frac{\delta\lambda_j}{\delta\alpha}) = \sum_{i=1}^{M} \frac{1}{\lambda_i}\frac{\delta\lambda_i}{\delta\alpha}$
Since Both sides of equation 3.117 come out to $\sum_{i=1}^{M} \frac{1}{\lambda_i}\frac{\delta\lambda_i}{\delta\alpha}$ the must be equal and therefore the equation holds.
To prove 3.92 from starting with 3.86:
$ln(t|\alpha,\beta) = \frac{M}{2}ln\alpha + \frac{N}{2}ln\beta - E(m_N) - \frac{1}{2}ln|A| - \frac{N}{2}ln(2\pi)$

$\frac{dln(t|\alpha,\beta)}{\alpha} = \frac{M}{2}\frac{1}{\alpha} - \frac{1}{2}m_N^T m_N - \frac{1}{2}Tr(A^{-1}\frac{d}{d\alpha}A)$
$= \frac{1}{2}(\frac{M}{\alpha} - m_N^T m_N - Tr(A^{-1}\frac{d}{d\alpha}A))$
Since $A = S_N^{-1}$ it represents the mean of the prior distribution. In otherwords it is α So $\frac{d}{d\alpha}A = 1$
$= \frac{1}{2}(\frac{M}{\alpha} - m_N^T m_N - Tr(A^{-1}))$
$= \frac{1}{2}(\frac{M}{\alpha} - m_N^T m_N - \sum \frac{1}{\lambda_i+\alpha}))$
This is the right hand side of equation 3.89. Since 3.89 is used to derive 3.92, then 3.117 can be used to derive 3.92 using 3.89 as an intermdiary step.

Programming Part A:

(graphs on next page)

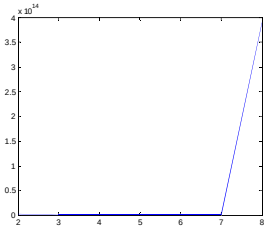
For this programming part I ended up using Regularized Least Squares where I varied lambda between 0, .1, .5 , 10, 500,100, and a 1000 . For the basis, I ended up using a variation of a polynomial basis with some additional terms (explained below). I determined the goodness of these functions by plotting the Total Error function for Regularized least squares as I tried out different variables. The Mathematical step by step logical and formula build up is as follows:

I explain how I chose my functional basis below. But assuming that I had a functional basis I decided that using a regularized least squares would be the best approach. I decided to first implement the equations from 3.1 since that represented the case lambda = 0. After I had that first case down I then would extend the formulas to their more general equation. So for obtaining the weights I first used equation 3.15 and then later changed it to 3.28 in order to incorporate lambda. For the total error function, I first I first used (3.12) to describe the error and then changed it to formula 3.27 when moving to different values of lambda. In my code when changing to the more generalized version, I commented out the original code so you can still see my original implementation. Then as I ran through all possible choices of the missing third value, I would keep track of the error and plot the error.

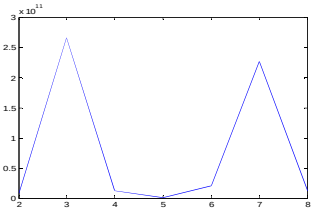
Since we were given that FTP (feature x1) and WE(feature x2) for now are good predictors with one other variable, I assumed that the basis should include those variables somehow. Then since we have to find a third variable I just called it x_3. Since the variables might interact with each other in some way, just using a form of the three variables by themselves might not describe the functional space. I decided that linear combinations of this variable should also be expressed in this basis. So I originally decided to make three of the basis functions : {phi0 = 1, phi1 =x1 phi2 = x2, phi3 = x3}. But when I plotted these errors, the graph produced came out poorly. Therefore I decided to switch the basis to a polynomial format. So then the basis was {phi0 = 1, phi1 = x1, phi2 = x2^2 , phi3 = x3^3}. I tried to also experiment with gaussian and logistic sigmoid functions but was running into problems of the errors being either infinity or 0. Therefore I decided that polynomial basis was the best way to go. Trial's with the lambda varying show that the predictions are fairly consistent and therefore there justifies using a polynomial basis.

The plots of all the trial runs seem to show that variable x5 generated the lowest total error, followed by variable x4. These variables correspond to LIC and GR respectively. It was very beneficial to vary the lambda as the difference in error between LIC and GR became more prominent as lambda increased.

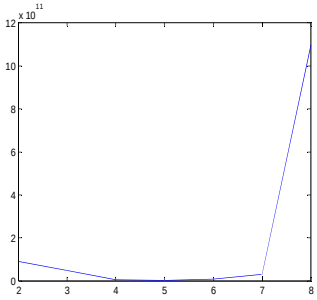
Plots on the next page.



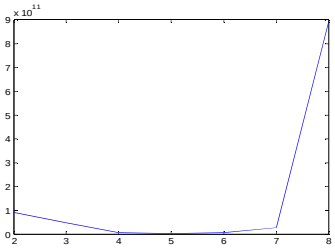
My Naive Radial Basis Attempt?



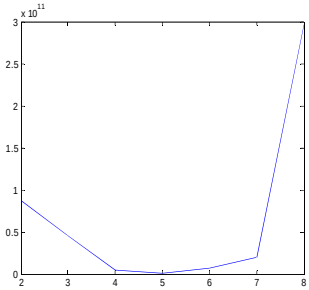
My Polynomial Basis Attempt (lambda = 0)



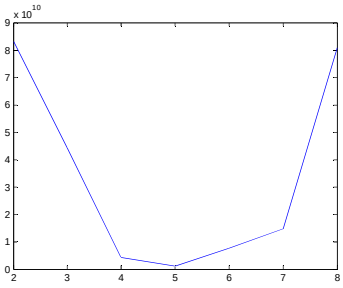
Polynomial Basis attempt (lambda = 1)



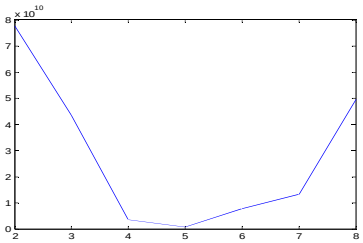
Lambda = 10



Lambda = 100



Lambda = 500



Lambda = 1000

Programming Part B: Nearest Neighbor.

(I) Method for changing the question marks when the features are categories.
Justification for the Mean/Median Imputation. What follows is my reason and justification for what to change the question marks to. In general I used a cut command to see the general distribution of categories, as well as the long grep piped with the cut command to see the distribution for positive and negative labeled samples. The only real different cases were features A9 and A10 where the instances with positive and negative labels had different letter appear the most times. In these cases the question marks should be changed based on the label classification. (At the end of the write up I have attached the output from my shell) I decided to impute based on just the training since I felt the training set is more important in the k nearest neighbor algorithm

1. Feature A1:

Label all question marks to 'b' since no matter the positive or negative classification, 'b' appeared the most times.

2. Feature A4:
   Label all question marks to 'u' since no matter the positive or negative classification 'u' appears the most times
3. Feature A5:

Label question marks to 'g' for the same reasoning as above.

4. Feature A6:

Label question marks to 'c' for the same reasoning as above.

5. Feature A7:
   Label question marks to 'v' for the same reasoning as above.
6. Feature A9:
   'f' occurred more frequently in negative samples and 't' occurred more frequently in positive samples. So any question marks in for this feature in instances with a negative label should be changed to 'f' while positive instances should be changed to 't'.

7. Feature A10:

't' occurred more frequently in negative samples and 'f' occurred more frequently in positive samples. So any question marks in for this feature in instances with a negative label should be changed to 't' while positive instances should be changed to 'f'.

8. Feature A12: Change question marks to f
9. Feature A13: Change question marks to g
10. Field A16: No question marks present

(II) Method for changing Question marks for when the features are continuous variables.
For real value features I replaced the missing values with the label conditioned mean. This meant counting how many instances of each label there were, summing their values and dividing by the amount. I assumed that it was better to replace question marks based on the training data set since for both data sets during the processing since I assumed that it had a higher significance since we are using that as our data base.

(III) Z Scaling: for real values I used the z scaling as described in the homework handout.

For Z scaling the mean and standard deviation are calculated with for each data set separately.

(IV) I wrote the programs in python and named them knn.py and process.py They were written in python 2.7 (tried it on csa2 and it ran) I implemented it based of an algorithm I found from http://www.math.le.ac.uk/people/ag153/homepage/KNN/OliverKNN_Talk.pdf

(V) In order to create a program that ran on the BU Computing enviroment, I made a shell script called process. This shell script took the parameters and passes them into a python program called process.py and runs the program. So the command "./process crx.data.training crx.data.testing" can be used to run the python program that will then produce crx.training processed and crx.testing.processed. For the man/median imputation, I decided to only use the training data. I did a similar analysis on the testing set and changed them seperately.
My Method that the python program takes the file names, and imports them as csv.

Table of Results for K Nearest Neighbors:

For at least 2 different values of k. I wrote it into the knn.py file a smalle script that checks the accuracy in a similar format to the pearl script. The results seem to make sense because the more neighbors you take, the more opportunities there are for an individual point to be mislabeled.

K Value	Lenses Results	CRX results
1	83.00%	100%?
2	67.00%	89.49%
3	83.00%	89.84
10	50.00%	85.00%
100	NA	78.00%
300	NA	69.00%