Concordia University

Computer Science and Software Engineering Department

COMP 6321

Machine Learning

Assignment 2

Deep Pandya

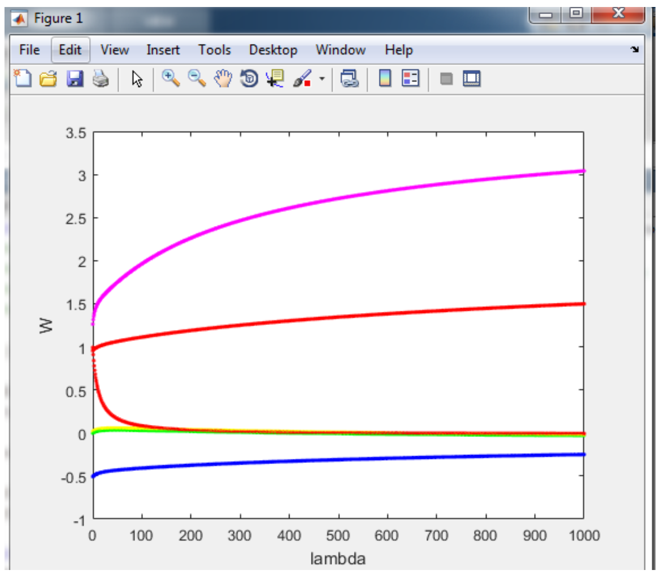
27162391

Question 1:

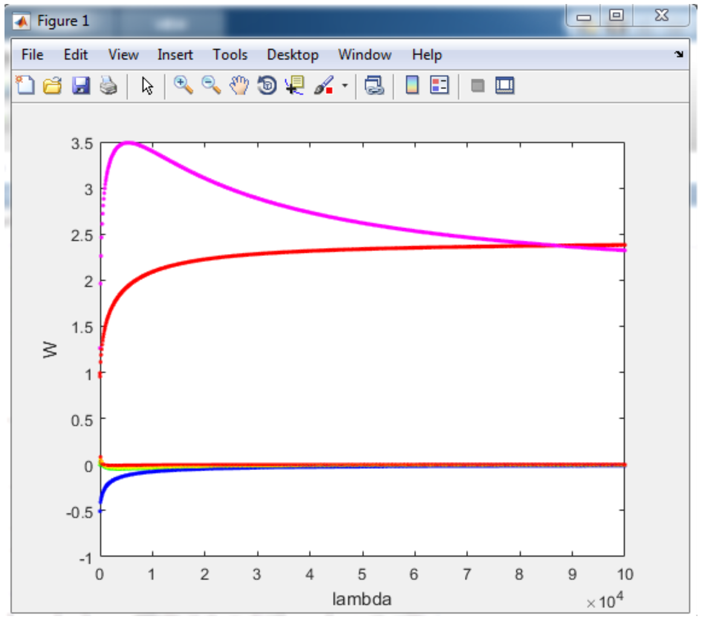
(A)

Please check Q1a.m file for the results.

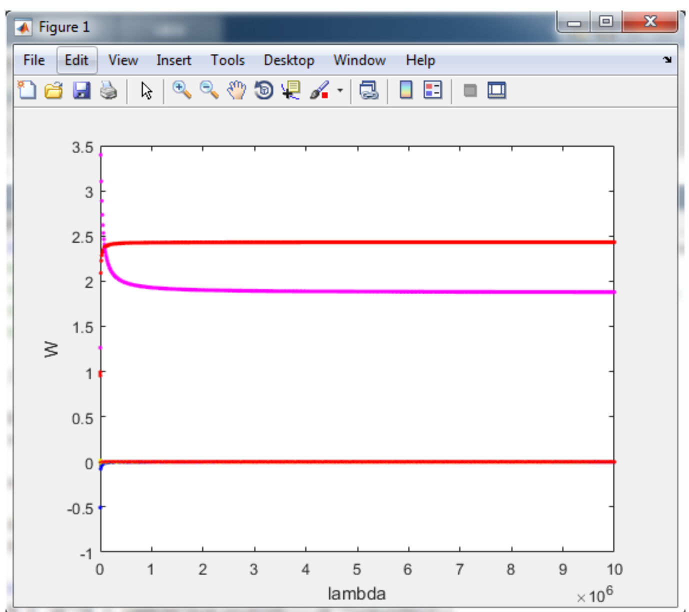
I am taking the first 90 data as training set, and the rest 10 as testing set. At first I chose λ in the interval of 0 to 1000. As shown in below figure, testing error is larger than training error, and they both go higher as goes λ higher.



Then I chose λ in the interval of 0 to 100,000. See below figure, the plots have one intersection that after gets to really large (80,000 in this case), MSE for test set goes lower than that for train set.



And for the conclusion, I chose λ in the interval of 0 to 10,000,000. See below figure, that both error gets to converge there.



Question 1 (b):

Please check File Q1b.m for the results.

Q:-1 (b)

For L1 regularization; we have,

$$\omega = \arg\min_{\omega} \left(\omega^T X^T X \omega - 2 y^T X \omega \right)$$

subject to $A\omega \leq t$

According to the question,

w has four parameters in all from ω_0 to ω_3 so, A should be $16 * 4$ matrix, that $A\omega$ should include all combinations of ω_0 to ω_3.

Being more specific, the constraint should be as below:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & & \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \omega_0 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \leq \begin{bmatrix} t \\ t \\ t \\ \cdots \end{bmatrix}$$

In matlab,

ω = quadprog (H, F, A, b).

ω subject to $A\omega \leq b$

$$J(\omega) = \frac{1}{2} \omega^T H \omega + F^T \omega$$

In the problem,

H equals to $X^T X$, f equals to $-(y^T X)^T$.

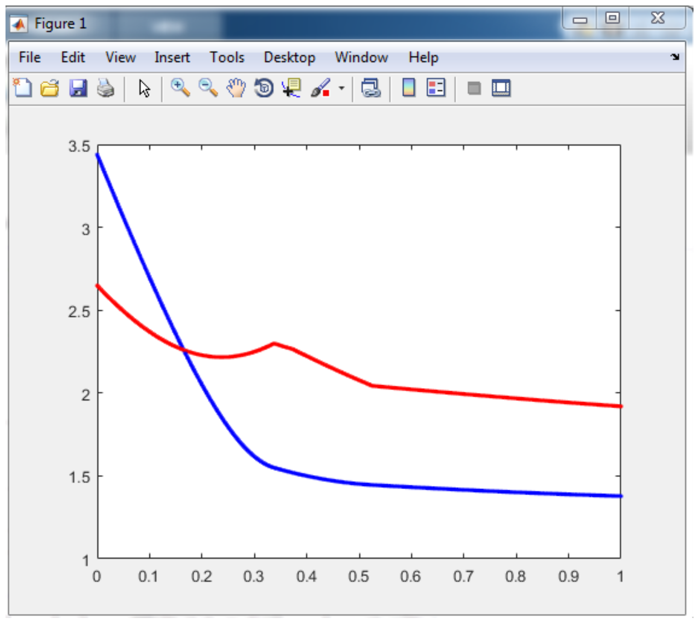
A is a $16*4$ matrix that includes all combination of 1 and -1.

b is $16*1$ vector where all elements are equal to t which represents the regularization parameter 'μ'.

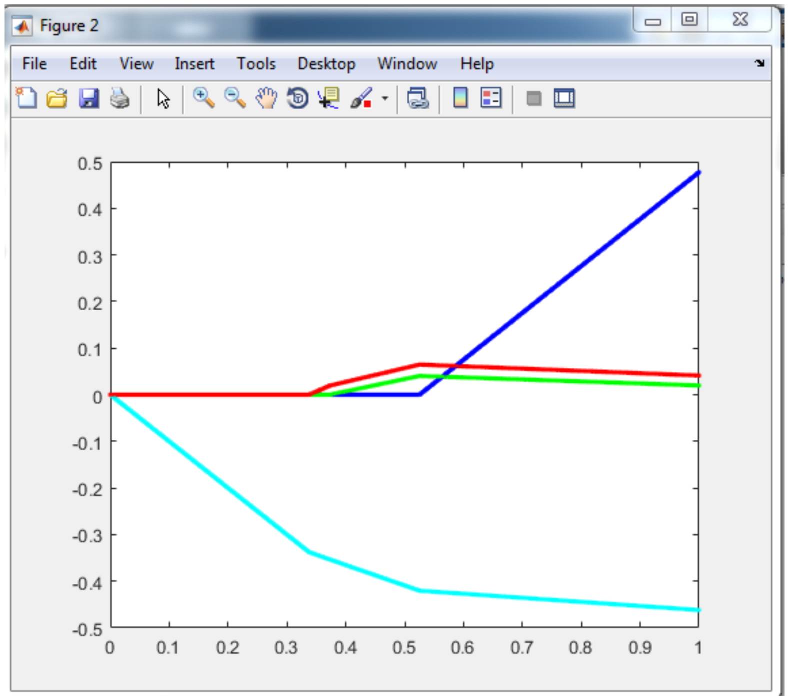
Question 1(c):

Please Check File Q1c.m for the results.

Below figure shows how error changes according to η —regularization parameter. As η is shrinking smaller to 0, training error is getting higher than testing error. They have an intersection point between 0.1 and 0.2.



Below figure shows how parameters of w change with regard to η. As η is shrinking smaller to 0, all the four parameters will converge to 0.



How the data is generated:

Regularization is to attach a cost function to another component related with parameters w:

$$\arg\min_{w} \frac{1}{2}(\Phi w - y)^T(\Phi w - y) + \frac{\lambda}{2}\sum_{k=0}^{K-1} |w_k|^q$$

Which equals to:

$$\min_{w} J_D(w) = \min_{w}(\Phi w - y)^T(\Phi w - y)$$

$$such\ that \sum_{i=1}^{n} |w_i|^q \le 1/\lambda$$

For the first part, it is a homogenous object in spatial space centered at $\Phi^{-1}y$ (circle in 2D plane, sphere in 3D, so on so forth.)

The Error Tendency:

For L2 Regularization, the error for testing set is higher than training set when λ is small. As I have mentioned above, w is a well-calculated set of parameters that optimizes the cost, and since it is trained using training set data, it makes sense to get lower error rate for training than for testing;

When λ gets very large, w began to converge to 0, and in this case, it is **underfitting,** so training error goes higher. And it will finally converge to some value when w gets to be fixed at 0;

For testing set, since the polynomial curve is getting smoother (because it is underfitting), it will better fit the testing set.

For L1 Regularization, same things happen. The only difference is $\mu \propto 1/\lambda$, so check it in the inverse direction along the horizontal axis, it is the same as I have mentioned above.

Question: 2

Q:- 2 (a)

⇒ According to the slide (30) from lecture 3, the conditional distribution of X can be expressed as below:

$$P(X|y=0) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(\frac{-1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\right)$$

$$P(X|y=1) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(\frac{-1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right)$$

We can divide the two conditional probability of $y=1$ to $y=0$, to find the decision boundry,

$$\frac{P(y=1/x)}{P(y=0/x)} = \frac{P(x|y=1)\,P(y=1)}{P(x|y=0)\,P(y=0)}$$

Apply log,

$$\ln \frac{P(x|y=1)\,P(y=1)}{P(x|y=0)\,P(y=0)}$$

$$= \ln\frac{P(y=1)}{P(y=0)} + \ln \frac{\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\right)}{\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right)} \quad \ldots\ldots (1)$$

$$\propto -\sum_{i=0}^{n}\left(\frac{(x_i-\mu_{0,i})^2}{\sigma_i^2} - \frac{(x_i-\mu_{1,i})^2}{(\sigma_i)^2}\right)$$

$$= -\sum_{i=0}^{n-1}\left(\frac{(x_i-\mu_{0,i})^2}{\sigma_i^2}-\frac{(x_i-\mu_{1,i})^2}{\sigma_i^2}\right)$$

$$-\left(\frac{(x_n-\mu_{0,n})^2}{\sigma_n^2}-\frac{(x_n-\mu_{1,n})^2}{\sigma_n^2}\right)$$

In order to find the best decision boundary which divides the two classes of y, we need to make the log ratio as extreme as possible.

Since $x_1, x_2, \ldots, x_{n-1}$ are known, plug them into equation(1) the first part is fixed, the only factor that can change its value is x_n.
If $x_n = \mu_{0,n}$, which is $x_n = E(x_n|y=0)$ will get the target ratio, or while $x_n = \mu_{1,n}$ ($x_n = E(x_n|y=1)$) will get the smallest ratio.
The two classes are divided as far as possible.

Q:-3 (a)

⟹ For every feature x, two parameters are required. And one parameter for the probability of y is required.

⟹ Since there are two features in initial ~~active~~ Naive Bayer, so, in total ~~to~~ that's $2*2+1=5$ features,

⟹ And they are,

$$\theta_{1,1} = P(x_1=1 \mid y=1)$$
$$\theta_{2,1} = P(x_2=1 \mid y=1)$$
$$\theta_{1,0} = P(x_1=1 \mid y=0)$$
$$\theta_{2,0} = P(x_2=1 \mid y=0)$$
$$\theta_1 = P(y=1)$$

* When the classifier gets a new feature x_3, accordingly the system should have 7 parameters, but since x_3 is dependent on x_2, parameters for x_3 are duplicates of that for x_2.

So there are still 5 independent parameters.

(b)

⇒ We can express the conditional probability of X given y, using its features:

$$P(X|y)$$
$$= P(x_1, x_2, x_3 | y)$$
$$= P(x_1|y) \, P(x_2|y, x_1) \, P(x_3|y, x_1, x_2)$$
$$= P(x_1|y) \, P(x_2|y) \, P(x_3|y, x_2)$$
$$= P(x_1|y) \, P^2(x_2|y)$$

The first three derivations are due to Bayes theorem, the last line is because x_3 is dependent on x_2.
Then accordingly, the decision boundary is,

$$\log\left(\frac{P(y=1|x)}{P(y=0|x)}\right)$$

$$= \log\left(\frac{P(y=1)}{P(y=0)}\right) + \sum_{i=1}^{n} \log \frac{P(x_i|y=1)}{P(x_i|y=0)}$$

* In case of two independent features, the boundary can be further deduced to,

$$\omega_0 + \omega_{1,0} + \omega_{2,0} + (\omega_{1,1} - \omega_{1,0})x_1 + (\omega_{2,1} - \omega_{2,0})x_2$$

Here, $\omega_0 = \log \frac{\theta_1}{1-\theta_1}$, $\omega_{i,1} = \log \frac{\theta_{i,1}}{\theta_{i,0}}$ | $\omega_{i,0} = \log \frac{(1-\theta_{i,1})}{(1-\theta_{i,0})}$

.... (1)

Known the data will give these parameters concrete values.

So, In case of three features,

$$\omega_0 + \omega_{1,0} + 2\omega_{2,0} + (\omega_{1,1} - \omega_{1,0})x_1 + 2(\omega_{2,1} - \omega_{2,0})x_2$$

..... (2)

So, according to equation (1) and (2),

the slope of the boundary changes by one half.

$$x_2 = -\frac{\omega_{1,1} - \omega_{1,0}}{\omega_{2,1} - \omega_{2,0}} x_2 + \alpha$$

.... (3)

So, let's re-write the boundary for three-features,

$$x_2' = -\frac{1}{2} * \frac{\omega_{1,1} - \omega_{1,0}}{\omega_{2,1} - \omega_{2,0}} x_1 + \alpha$$

.... (4)

which is half of slope of that of original classifier.

⟹ The worst-case scenario happen when the slope of (3) is 180°, in which case, the slope of (4) is only 90°, that makes for the biggest discrepancy ever between the two decision boundaries.

⟹ The notiable thing is the decision boundry still locates in 2D plane, rather than 3D space, but ~~still~~ just change its slope, which just lower the accuracy since it put more emphasis on one features than the other.
That's Introducing factors that lower robustness.

Q :- 4

⇒ for ML estimation,

According to the logistic regression, The difference between y_i and its logistic regression value $\sigma(x_i)$ should be minimal.

Using the cross entropy to express this error, and apply log trick will get the equivalent expression,

$$\arg\min_{\omega} J = -\left(\sum_{i=1}^{m} y_i \log \sigma(\omega^T x_i) + (1-y_i)\log(1-\sigma(\omega^T x_i))\right)$$

To solve, the argument we need to find out its first order derivative,

$$\nabla_\omega J = \sum_i (y_i - \sigma_\omega(x_i))\, x_i$$

The optimal W should make its first order derivative equals to one.

Thus,

$$\sum_i (y_i - \sigma_{\omega_{ML}}(x_i))\, x_i = 0 \quad \ldots\ldots (1)$$

* For MAP estimation,

W_{MAP} is a set of parameter that make the posterion estimation to maamum.
To compare the relationship between

$\|W_{MAP}\|_2$ and $\|W_{ML}\|_2$.

we should use cross entropy error and apply log.

$$P(\omega) = \frac{1}{\sqrt{2\pi c}} * e^{-\frac{w^2}{2\tau^2}}$$

It's cost function should be,

$$J = -\left(\sum_{i=1}^{m} y_i \log 6(\omega^T x_i) + (1-y_i)\log(1-6(\omega^T x_i))\right) - m\frac{w^2}{c^2} - \log\left(\sqrt{2\pi c}\right)$$

apply first derivative,

$$\nabla_\omega J = \sum_i (y_i - 6_\omega(x_i))\, x_i - \frac{w}{\tau^2}$$

and W_{MAP} should satisfy,

$$\sum_i (y_i - 6_{W_{MAP}}(x_i))\, x_i - \frac{W_{MAP}}{\tau^2}$$

..... (2)

According to 1 and 2 ,

$$\sum_i \left(2y_i - \frac{1}{1+e^{-W_{max}X_i}} - \frac{1}{1+e^{-W_{max}X_i}} \right) X_i = m \frac{W_{MAP}}{\tau^2}$$

Question 5:

(a)

Please check LogisticRegression.m file for the results.

To implement logistic regression, first add bias term for all the samples. For every sample, the logistic function is:

$$h(X) = \frac{1}{1+e^{-W^T X}}$$

To find the optimal set of W, I have used iterative recursive least-squares (Newton method). The idea is to divide W's first derivative over that of second derivative, update this W by subtracting this division:

$$J' = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)x_i$$

$$J'' = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i)(1-h_\theta(x_i))x_i(x_i)^T)$$

$$w = w - \frac{J'}{J''}$$

In my case, I initialized *W* as vector of zeros, and take 20 iterations of updating. It turned out that the first derivative converges to 0 and all *W* converge to some certain values finally.

(b)

Please check GaussianNaiveBayes.m and TenFolderCrossValidation.m file for the results.

To implement Guassian Naive Bayes, we need to find both *W* and bias term *W0* separately as following:

$$w = \Sigma^{-1}(\mu_1 - \mu_2)$$

$$w_0 = -\frac{1}{2}\mu_1{}^T\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2{}^T\Sigma^{-1}\mu_2 + \ln\frac{p(C_1)}{p(C_2)} *$$

When only the first features are employed, w is -1.2073, w0 is 0.3838.

To implement 10-fold cross validation, just divide all samples into 10 bins and each of them takes turn to be the testing set, with all resting being training set. The average error for Guassian naive bayes model is 0.5515 (for all the four sub-questions, I used cross entropy as the error, and all errors are compared based on this same criteria. This will not be claimed later).

Take only the first feature and use 10 fold cross validation in Logistic Regression will get error of 0.5517. They are very close when only one feature is employed.

(c)

Please check GuassianNaiveBayes.m and LogisticRegression.m files for the results.

The idea is the same as that in (b). Only when implement this question, the input argument x should be at its full size, which is 194*32 in this case.

In this problem, W is a 32*1 vector that can be checked when you run *hw5.m*. And w0 is -1.55.

(D)

All files for this question are the same as for previous (b). Just need to modify the input argument of x to extend it to its full size. All the differences from this change will be taken care of in the program.

The average error for Guassian model is 0.5589, for logistic is 0.7272.

Turned out that when all the features are used, Guassian naïve bayes model has a smaller error rate than Logistic. Guassian error doesn’t change too much from question (b), but for Logisitc, the error goes higher with more features.

In Below Table, I’ve listed all the errors for both models. As we can checked there, when only one feature is used, the error rate for each fold are very close. While all features are used, some fold of logistic regression is close to Guassian Naïve Bayes, but there are three folds in particular that get very large error rate, and as a result, raise the average error. For Guassian model, the errors for 10 fold tend to be more even than that of Logistic Regression.

TABLE: Cross-Entropy Error over Guassian and Logistic Model

		Guassian Naïve Bayes		Logistic Regression	
		One feature	All features	One feature	All features
Error for 10 fold	1	0.6655	0.7938	0.6656	1.0748
	2	0.4100	0.4057	0.4100	0.4828
	3	0.7431	0.9151	0.7434	1.4534
	4	0.5188	0.5498	0.5187	0.8139

	5	0.4286	0.5093	0.4293	0.5580
	6	0.5557	0.5003	0.5558	0.5283
	7	0.6550	0.6343	0.6552	0.9029
	8	0.5960	0.5795	0.5959	0.6048
	9	0.4840	0.3536	0.4841	0.3986
	10	0.4579	0.3473	0.4591	0.4547
Average		0.5515	0.5589	0.5517	0.7272

References:

- Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006
- Kevin P. Murphy, *Machine Learning. A Probabilistic Perspective,* MIT Press, 2012.
- http://luthuli.cs.uiuc.edu/~daf/courses/CS-498-DAF-PS/Lecture%206%20-%20Gaussian%20Classifiers.pdf
- M. Jordan, J. Kleinberg, B. Scholkopf. Pattern Recognition and Machine Learning. Springer, 2006. Pp: 199.