CSE 847 Home Assignment 3

Submitted by: Ritam Guha (MSU ID: guharita)

Date: March 19, 2021

1 Linear Algebra III

1. (10 points) Let $A \in \mathbb{R}^{m \times n}$ be a matrix of rank n. Prove that $\|A(A^TA)^{-1}A^T\|_2 = 1$.

   Response:

   A is a full rank matrix. So, SVD of A looks like:

   $$A = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} \Sigma \\ \\ 0 \end{pmatrix} \begin{pmatrix} V^T \end{pmatrix}$$

   where $U_1 \in \mathbb{R}^{m \times n}$, $U_2 \in \mathbb{R}^{m \times (m-n)}$ have orthogonal columns , $\Sigma \in \mathbb{R}^{n \times n}$ is a diagonal matrix and $V \in \mathbb{R}^{n \times n}$ is orthogonal. Simplifying A gives us:

   $A = U_1 \Sigma V^T$

   Plugging in this form in the given expression, we get:

   $||U_1\Sigma V^T((U_1\Sigma V^T)^T U_1 \Sigma V^T)^{-1}(U_1\Sigma V^T)^T||_2$

   $= ||U_1\Sigma V^T(V\Sigma U_1^T U_1 \Sigma V^T)^{-1} V\Sigma U_1^T||_2$

   $= ||U_1\Sigma V^T(V\Sigma^2 V^T)^{-1} V\Sigma U_1^T||_2$

   $= ||U_1\Sigma V^T(V^T)^{-1}\Sigma^{-2}V^{-1} V\Sigma U_1^T||_2$

   $= ||U_1\Sigma\Sigma^{-2}\Sigma U_1^T||_2$

   $= ||U_1U_1^T||_2$

   $= ||\mathbb{I}||_2$

   According to the definition of 2-norm of matrices, $||A||_2$ is the square root of the largest eigen value of

$A^T A$.

Square root of the Largest eigen value of $\mathbb{III}$ is 1. [Proved]

2. (10 points) Let A and B be two positive semi-definite matrices in $\mathbb{R}^{n\times n}$. Prove or disprove: A and B are PSDs. So, for any $v\epsilon\mathbb{R}^n$, $v^T Av$ and $v^T Bv$ should be greater than or equal to 0.

   (a) $A + B$ is positive semi-definite

   Response:

   $v^T(A + B)v$

   $= v^T Av + v^T Bv$

   where $v^T Av \geq 0$ and $v^T Bv \geq 0$

   So, $v^T(A + B)v \geq 0$

   Hence, it is a PSD.

   (b) AB is positive semi-definite

   Response:

   AB is not always postive semi-definite. While doing multiplication of matrices, the symmetry does not hold always. But according to the definition of PSD matrices, it should be symmetric. For an example, consider the following PSD matrices as A and B.

$$A = \begin{pmatrix} 1 & 4 \\ 4 & 16 \end{pmatrix}, B = \begin{pmatrix} 4 & -6 \\ -6 & 9 \end{pmatrix}$$

$$AB = \begin{pmatrix} -20 & 30 \\ -80 & 120 \end{pmatrix}$$

   So, AB is not symmetric. Now let's consider v as $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$

$$\begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} -20 & 30 \\ -80 & 120 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = -20$$

So, AB is not a PSD matrix. Using this counter example, it is proved that AB may not be a PSD.

(c) B^T is positive semi-definite

Response:

By definition of positive semi-definite matrices, they should be symmetric.

B is a PSD matrix, so it should be symmetric.

$B = B^T$

So, B^T is also a PSD matrix.

2 Linear Classification

Questions in the textbook Pattern Recognition and Machine Learning:

1. (10 points) Page 220, Question 4.1

   Response:

   The convex hull for a set of data points x_n is given by:

   $\mathbf{x} = \sum_n \alpha_n x_n$ where $\sum_n \alpha_n = 1$

   If there are two intersecting convex hulls $\mathbf{x}$, $\mathbf{y}$, for some combinations of $\alpha_n \geq 0$ and $\beta_n \geq 0$, the following equality should hold:

   $\sum_n \alpha_n x_n = \sum_n \beta_n y_n = c$ [A common point on the convex hull]

   Let's consider that these two intersecting convex hulls are linearly separable. Then from the definition of linear separability, we can say:

   $\hat{w}^T x_n + w_0 > 0$

   $\hat{w}^T y_n + w_0 < 0$

$\hat{w}^T c + w_0$

$= \hat{w}^T(\sum_n \alpha_n x_n) + (\sum_n \alpha_n) w_0$ [As $\sum_n \alpha_n = 1$]

$= (\sum_n \alpha_n \hat{w}^T x_n) + \sum_n \alpha_n w_0$

$= \sum_n \alpha_n (\hat{w}^T x_n + w_0)$

As $\hat{w}^T x_n + w_0 > 0$, $\hat{w}^T c + w_0 > 0$

If we use $c = \sum_n \beta_n y_n$, $\hat{w}^T c + w_0 = \sum_n \beta_n (\hat{w}^T y_n + w_0)$.

Then this becomes < 0 because $\hat{w}^T y_n + w_0 < 0$.

This leads to a contradiction. So, our assumption that the intersecting convex hulls are linearly separable was wrong.

If convex hulls of two sets of points intersect, they are not linearly separable.

In the same way, we can show that **If two sets of points are linearly separable, their convex hulls cannot intersect.**

2. (10 points) Page 221, Question 4.5

Response:

The given expressions are:

$y = \mathbf{w^T x}$

$m_k = \mathbf{w^T m_k}$

$s_k^2 = \sum_{n \epsilon C_k} (y_n - m_k)^2$

The fisher criterion can be represented as:

$J(w) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$

From the given expressions, we can derive the follwoing expressions:

$m_1 = \mathbf{w^T m_1}$

$m_2 = \mathbf{w^T m_2}$

$$
\begin{aligned}
(m_2 - m_1)^2 &= (\mathbf{w^T(m_2 - m_1)})^2 \\
&= \mathbf{w^T(m_2 - m_1)(w^T(m_2 - m_1))^T} \\
&= \mathbf{w^T(m_2 - m_1)(m_2 - m_1)^T w} \\
&= \mathbf{w^T S_B w}
\end{aligned}
$$

where $S_B = (m_2 - m_1)(m_2 - m_1)^T$

$$
\begin{aligned}
s_1^2 &= \sum_{n \epsilon C_1} (y_n - m_1)^2 \\
&= \sum_{n \epsilon C_1} (\mathbf{w^T x_n - w^T m_1})^2 \\
&= \sum_{n \epsilon C_1} (\mathbf{w^T(x_n - m_1)})^2 \\
&= \sum_{n \epsilon C_1} (\mathbf{w^T(x_n - m_1)(w^T(x_n - m_1))^T}) \\
&= \sum_{n \epsilon C_1} (\mathbf{w^T(x_n - m_1)(x_n - m_1)^T w}) \\
&= \mathbf{w^T}(\sum_{n \epsilon C_1} \mathbf{(x_n - m_1)(x_n - m_1)^T})\mathbf{w}
\end{aligned}
$$

Similarly $s_2^2 = \mathbf{w^T}(\sum_{n \epsilon C_2} \mathbf{(x_n - m_2)(x_n - m_2)^T})\mathbf{w}$

So,

$$
\begin{aligned}
s_1^2 + s_2^2 &= \mathbf{w^T}(\sum_{n\epsilon C_1}(\mathbf{x_n} - \mathbf{m_1})(\mathbf{x_n} - \mathbf{m_1})^\mathbf{T})\mathbf{w} + \mathbf{w^T}(\sum_{n\epsilon C_2}(\mathbf{x_n} - \mathbf{m_2})(\mathbf{x_n} - \mathbf{m_2})^\mathbf{T})\mathbf{w} \\
&= \mathbf{w^T}(\sum_{n\epsilon C_1}(\mathbf{x_n} - \mathbf{m_1})(\mathbf{x_n} - \mathbf{m_1})^\mathbf{T} + \sum_{n\epsilon C_2}(\mathbf{x_n} - \mathbf{m_2})(\mathbf{x_n} - \mathbf{m_2})^\mathbf{T})\mathbf{w} \\
&= \mathbf{w^T S_w w}
\end{aligned}
$$

The fisher criterion then becomes:

$$
\begin{aligned}
J(w) &= \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} \\
&= \frac{\mathbf{w^T S_B w}}{\mathbf{w^T S_w w}}
\end{aligned}
$$

Thus, the two forms are equivalent.

3. (10 points) Page 221, Question 4.6

Response: Equation 4.3 is given by:
$\sum_{i=1}^{N}(w^T x_n + w_0 - t_n)x_n = 0$ We can simplify it using equations (4.34), (4.35), (4.36) in the following manner:
$\Rightarrow \sum_{i=1}^{N} w^T x_n x_n + \sum_{i=1}^{N} w_0 x_n - \sum_{i=1}^{N} t_n x_n = 0$
$\Rightarrow \sum_{i=1}^{N} w^T x_n x_n - w^T m \sum_{i=1}^{N} x_n - (\sum_{n\epsilon C_1} \frac{N}{N_1} x_n + \sum_{n\epsilon C_2} -\frac{N}{N_2} x_n) = 0$ [Using equation 4.34, we get $w_0 = -w^T m$ and using different representations of t_n for the two classes]
$\Rightarrow \sum_{i=1}^{N} w^T x_n x_n - w^T m N m - (\frac{N}{N_1}\sum_{n\epsilon C_1} x_n - \frac{N}{N_2}\sum_{n\epsilon C_2} x_n) = 0$ [where m is the mean of the entire dataset]
$\Rightarrow \sum_{i=1}^{N} w^T x_n x_n - N w^T m m - (\frac{N}{N_1} N_1 m_1 - \frac{N}{N_2} N_2 m_2) = 0$
$\Rightarrow \sum_{i=1}^{N} x_n x_n^T w - N m m^T w - N(m_1 - m_2) = 0$ [As $w^T x_n$ and $w^t m$ are scalars]
$\Rightarrow \sum_{i=1}^{N} x_n x_n^T w - N m m^T w = N(m_1 - m_2)$

$\Rightarrow (\sum_{i=1}^{N} x_n x_n^T - Nmm^T)w = N(m_1 - m_2)$

$\Rightarrow (\sum_{i=1}^{N} x_n x_n^T - N(\frac{1}{N}(N_1 m_1 + N_2 m_2))(\frac{1}{N}(N_1 m_1 + N_2 m_2)^T))w = N(m_1 - m_2)$ [From equation (4.36), $m = \frac{1}{N}(N_1 m_1 + N_2 m_2)$]

$\Rightarrow (\sum_{i=1}^{N} x_n x_n^T - \frac{1}{N}(N_1 m_1 + N_2 m_2)(N_1 m_1^T + N_2 m_2^T))w = N(m_1 - m_2)$

$\Rightarrow (\sum_{i=1}^{N} x_n x_n^T - \frac{1}{N}(N_1^2 ||m_1||^2 + N_2^2 ||m_2||^2 + 2N_1 N_2 m_1^T m_2))w = N(m_1 - m_2)$

$\Rightarrow (\sum_{i=1}^{N} x_n x_n^T - \frac{N_1^2}{N}||m_1||^2 - \frac{N_2^2}{N}||m_2||^2 - 2\frac{N_1 N_2}{N} m_1^T m_2)w = N(m_1 - m_2)$

$\Rightarrow (\sum_{i=1}^{N} x_n x_n^T - \frac{N_1^2}{N}||m_1||^2 - \frac{N_2^2}{N}||m_2||^2 - \frac{N_1 N_2}{N}||m_1||^2 - \frac{N_1 N_2}{N}||m_2||^2 + \frac{N_1 N_2}{N}||m_1||^2 + \frac{N_1 N_2}{N}||m_2||^2 - 2\frac{N_1 N_2}{N} m_1^T m_2)w = N(m_1 - m_2)$

$\Rightarrow (\sum_{i=1}^{N} x_n x_n^T - \frac{N_1^2}{N}||m_1||^2 - \frac{N_2^2}{N}||m_2||^2 - \frac{N_1 N_2}{N}||m_1||^2 - \frac{N_1 N_2}{N}||m_2||^2 + (\frac{N_1 N_2}{N}||m_1||^2 + \frac{N_1 N_2}{N}||m_2||^2 - 2\frac{N_1 N_2}{N} m_1^T m_2))w = N(m_1 - m_2)$

$\Rightarrow (\sum_{i=1}^{N} x_n x_n^T - \frac{N_1^2}{N}||m_1||^2 - \frac{N_2^2}{N}||m_2||^2 - \frac{N_1 N_2}{N}||m_1||^2 - \frac{N_1 N_2}{N}||m_2||^2 + \frac{N_1 N_2}{N}(m_2 - m_1)(m_2 - m_1)^T)w = N(m_1 - m_2)$

$\Rightarrow (\sum_{i=1}^{N} x_n x_n^T - \frac{N_1}{N}(N_1 + N_2)||m_1||^2 - \frac{N_2}{N}(N_2 + N_1)||m_2||^2 + \frac{N_1 N_2}{N} S_B)w = N(m_1 - m_2)$

$\Rightarrow (\sum_{i=1}^{N} x_n x_n^T - N_1||m_1||^2 - N_2||m_2||^2 + \frac{N_1 N_2}{N} S_B)w = N(m_1 - m_2)$

$\Rightarrow (\sum_{n\epsilon C_1} (x_n x_n^T + ||m_1||^2 - 2m_1 x_n^T) + \sum_{n\epsilon C_2} (x_n x_n^T + ||m_2||^2 - 2m_2 x_n^T) - 2N_1||m_1||^2 - 2N_2||m_2||^2 + \sum_{n\epsilon C_1} 2m_1 x_n^T + \sum_{n\epsilon C_1} 2m_2 x_n^T + \frac{N_1 N_2}{N} S_B)w = N(m_1 - m_2)$

$\Rightarrow (\sum_{n\epsilon C_1} (x_n - m_1)(x_n - m_1)^T + \sum_{n\epsilon C_2} (x_n - m_2)(x_n - m_2)^T - 2N_1||m_1||^2 - 2N_2||m_2||^2 + 2N_1||m_1||^2 + 2N_2||m_2||^2 + \frac{N_1 N_2}{N} S_B)w = N(m_1 - m_2)$

$\Rightarrow (S_w + \frac{N_1 N_2}{N} S_B)w = N(m_1 - m_2)$ [Proved]

4. (10 points) Page 222, Question 4.15

Response:

The hessian matrix is given by:

$\mathbf{H} = \sum_{n=1}^{N} y_n(1-y_n)\phi_n\phi_n^T$

If we want to show that $\mathbf{H}$ is positive definite, for any non-zero vector v in $\mathbb{R}^M$, $v^T\mathbf{H}v$ should be greater than 0.

$v^T\mathbf{H}v$

$= v^T(\sum_{n=1}^{N} y_n(1-y_n)\phi_n\phi_n^T)v$

$= \sum_{n=1}^{N} y_n(1-y_n)(v^T\phi_n)(\phi_n^T v)$

$= \sum_{n=1}^{N} y_n(1-y_n)||v^T\phi_n||_2^2$

In this expression, $y_n \epsilon (0,1)$, so $y_n(1-y_n) > 0$. $v^T\phi_n$ can be treated as a linear combination of columns of Φ. As Φ is the design matrix, we can assume it to have linearly independent columns. Linear combination of linearly independent columns is only zero when the scalars used for the linear combination (here v) are zeros. But, v is a non-zero vector, so the linear combination of the columns of Φ cannot be negative which makes linear combination to be positive. So, there should be at least one n for which $v^T\phi_n$ is greater than 0 which makes $v^T\mathbf{H}v > 0$. So, by definition of PD matrices, $\mathbf{H}$ is a PD matrix.

So, we can see that the second derivative of the error function with respect to w is a PD matrix. This leads to the fact that the error function should be a convex (concave upwards) function of w. Hence, according to the properties of convex functions, it has a unique minimum value.

3 Linear Regression: Experiment

(40 points) In this part of homework you will explore the ridge regression and the effects of ℓ_2-norm regularization. You are to implement a MATLAB solver for ridge regression:

$$\min_w \frac{1}{2}\|Xw - y\|_2^2 + \frac{\lambda}{2}\|w\|_2^2.$$

You are not allowed to use the integrated ridge regression in MATLAB. You will use your solver to investigate the effects of the regularization on the DIABETES[1] dataset, and study the cross validation procedure.

[1] https://github.com/jiayuzhou/CSE847/blob/master/data/diabetes.mat?raw=true

1. Implement the ridge regression solver.

2. Train regression models on the DIABETES dataset using the training data (x_train, y_train variables in the data file). Vary the λ from $1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 10$ (In Matlab $1e-1$ means 0.1). Compute training error (predict y_train given X_train), testing error (predict y_test given X_test) for each λ. The error is measured by mean squared error (MSE):

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2,$$

   where N is the number of samples on which the error is computed, y_i is ground truth, and $\hat{y}_i$ is the prediction from data points given model w.

3. Perform 5-fold cross validation on the training data to estimate the best λ from training data.

In the homework, attach a brief report. In the report you need to discuss your findings in the experiment, include a plot showing how training/testing error changes when you vary the parameter λ (use log scale on λ). In the same plot, show the best λ obtained from your 5-fold cross validation procedure. Submit the MATLAB code (do add some comments in your code for others to understand your code) to the D2L along with your report.

Response: For the regression problem, we can find the expression of the weights as:

$$w = (\Phi^T\Phi + \lambda\mathbb{I})^{-1}\Phi^T t \tag{1}$$

where Φ is the design matrix and t is the target vector. The weights are computed from the training matrix and then applied over test data to compute the predictions. Based on the predictions, every such model provides a MSE which can be used as the quality indicator of the model.

The model has one hyperparamter which is λ. In order to get one optimal value of λ, k-fold cross-validation is performed. After getting an optimal value of λ, that is used to build the model which is finally applied over the test data. The entire result of this experimentation is tabulated in Table 1.

Different types of normalization techniques are also used on the data to measure the change in errors based on

different types of normalizations. For this purpose, 5 types of normalization techniques are used apart from the non-normalized scenario. These normalization techniques are available in *MATLAB* under the names: 'range', 'scale', 'zscore', 'norm', 'center'. Each of these normalization techniques use different methods to scale the data. The description of each type of normalization can be found in the *MathWorks* documentation.

In Table 1, for each of the λs, Train Error and Test Error refer to the MSE values over the training data and testing data for the model generated using Equation 1 using that λ. The Train Error is always less than the Test Error because Training data is used to build the model but the Test data is completely unseen to the model. So, Test data is used to evaluate the generalization of different models. Train Error and Test Error are computed before cross-validation with all the possible models.

After performing cross-validation, the best λ value out of the mentioned possiblilities are calculated for every normalization procedure. We can see that the values change depending on the normalization technique used. So, normalization can affect the selection of the model which is really interesting. We can see that for every scenario, the regularized version is providing better Mean Square error compared to the non-regularised variant. To check the correctness of the code, I also used the *MATLAB* integrated ridge regression for benchmarking and founf out it is performing in a similar manner with little variance.

	Error Type	Normalization Type					
		None	**Range**	**Scale**	**Z-score**	**Norm**	**Center**
λ=1e-6	Train Error	19560.40391	0.021933	3.366411	0.385461	0.002891	2189.03486
	Test Error	93835.00672	3.589247	1005.224421	4.338947	0.056803	3558.254969
λ=1e-5	Train Error	19695.35482	0.021947	3.380637	0.385516	0.002906	2194.171893
	Test Error	76911.07209	1.621412	462.882938	2.244155	0.032902	3514.411567
λ=1e-4	Train Error	20060.11614	0.022058	3.431649	0.385708	0.002956	2209.43153
	Test Error	57531.76027	0.123865	80.669288	0.808274	0.013571	3483.224263
λ=1e-3	Train Error	20384.66425	0.022171	3.451689	0.385887	0.003003	2227.638985
	Test Error	49060.59804	0.107295	40.584322	0.649208	0.0083	3285.303421
λ=1e-2	Train Error	20918.86518	0.022419	3.493135	0.387634	0.003069	2306.837688
	Test Error	44405.1394	0.055242	18.906226	0.644382	0.007227	2951.255302
λ=1e-1	Train Error	22075.71567	0.023389	3.556196	0.390146	0.003194	2511.476379
	Test Error	36994.69673	0.035062	9.140952	0.636367	0.006359	2852.421846
λ=1	Train Error	24730.73933	0.025154	3.620571	0.39539	0.003527	3288.941082
	Test Error	30517.77394	0.029653	7.306174	0.543775	0.005187	3510.851314
λ=10	Train Error	27165.12613	0.03021	3.728976	0.415413	0.003947	4917.706446
	Test Error	30057.3035	0.033097	6.58842	0.470815	0.004905	5390.676364
λ=100	Train Error	27827.97731	0.045432	4.009522	0.465563	0.004107	5556.667563
	Test Error	30414.29643	0.048561	5.430243	0.474178	0.004982	6136.625862
Best λ according to CV-MSE		10	1	100	100	100	0.1
MSE for Non-Regularized version		106775.3616	3.972377	1132.597799	4.838317	1.160637	3630.528184
MSE for Ridge Regularized version		30057.3035	0.029653	6.081624	0.461673	0.004905	2852.421846
MSE for MATLAB integrated Ridge Regularized version		30145.62543	0.094501	4.25334	0.461673	0.004928	5911.960252

Table 1: Results for the Linear Regression Experimentation.

In Figure 1, the average MSE values during cross-valdiation are plotted for different values of λ for various Normalization techniques.

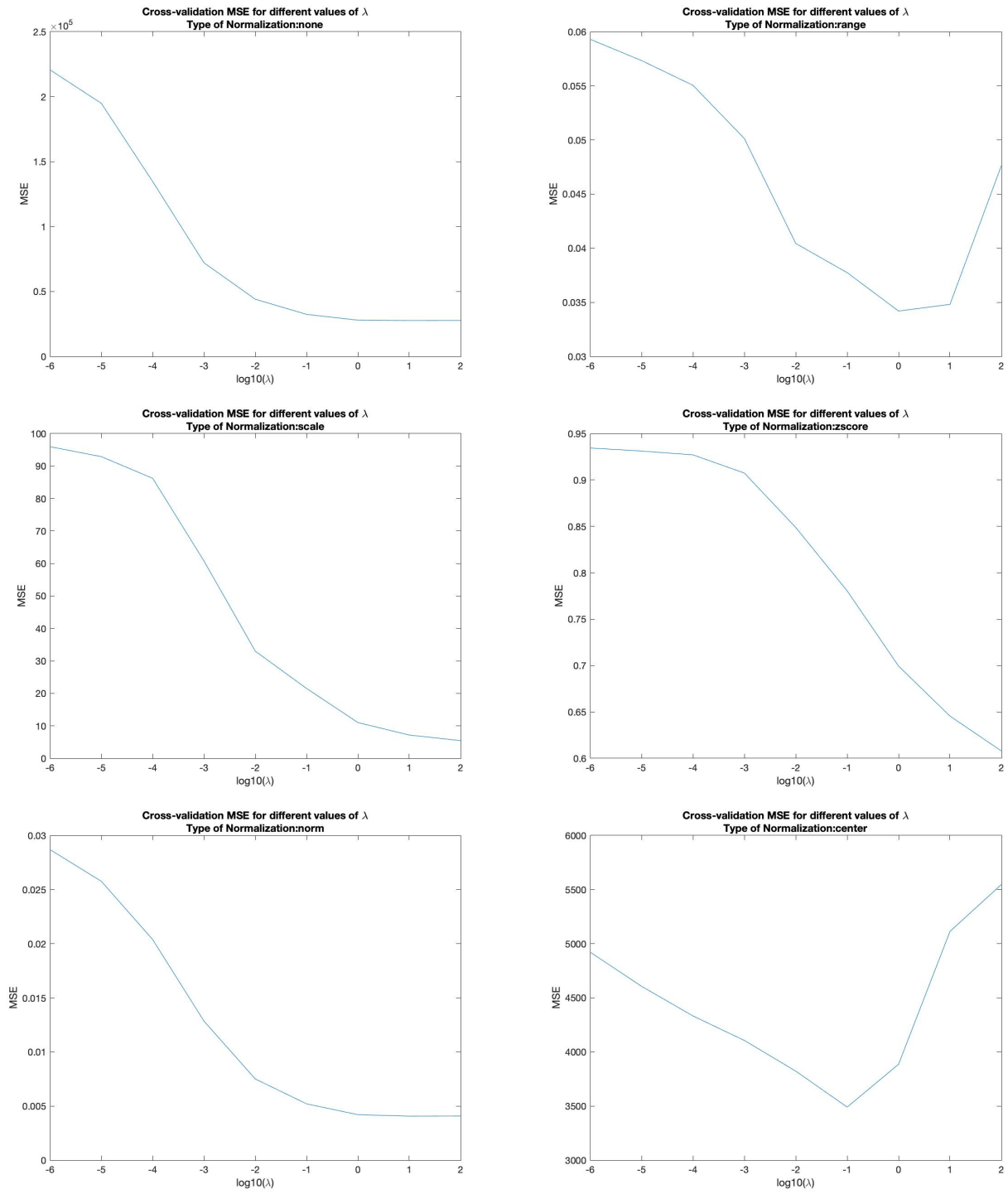

Figure 1: Plot of Cross-Validation MSE values for different λs for different types of normalizations

From the Figure, it is visible that different λ values are preferred for different normalization techniques. In case of 'range' and 'center' normalizations, we get a v-shaped curve, so it is ensured that we have reached a

global optima. In the non-normalized scenario and the 'norm' normalization, the MSE stabilizes after $\lambda = 10$. For rest of the cases, the MSE value keeps on decreasing even after $\lambda = 10$.

The code used to generate the outcome is provided below. It is also attached to *D2L* as **RidgeRegSolver.m**.

```
data = importdata('diabetes.mat');
x_train = data.x_train;
y_train = data.y_train;
x_test = data.x_test;
y_test = data.y_test;
num_folds = 5;
normalization_type = 'none';

if(normalization_type ~= 'none')
    x_train = normalize(x_train, normalization_type);
    y_train = normalize(y_train, normalization_type);
    x_test = normalize(x_test, normalization_type);
    y_test = normalize(y_test, normalization_type);
end


% Calling the Ridge Regression Solver
RidgeReg(x_train, y_train, x_test, y_test, num_folds, normalization_type);

%% Function for solving Ridge Regression
function [] = RidgeReg(x_train, y_train, x_test, y_test, cross_valid_k,
    normalization_type)
    num_feat = size(x_train, 2);
    lambda_vals = [1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 10, 100]; % possible
        lambda values
```

```
24      num_lambdas = size(lambda_vals,2);
25
26      for i=1:num_lambdas
27          [train_error, test_error] = compute_error(x_train, y_train, x_test,
                y_test, lambda_vals(1,i));
28          fprintf('lambda = %f: Train MSE:%f Test MSE:%f\n', lambda_vals(1,i),
                train_error, test_error);
29      end
30      fprintf('\n')
31
32      lambda = cross_validate(x_train, y_train, lambda_vals, cross_valid_k,
            normalization_type);
33      fprintf('Best Lambda: %f\n', lambda);
34
35      phi = x_train;
36      t = y_train;
37
38      w_reg = (phi' * phi + lambda * eye(num_feat, num_feat))^-1 * (phi' * t); %
            weights with regularization
39      w_noreg = (phi' * phi)^-1 * (phi' * t);     % weights without regularization
40      w_reg_matlab = ridge(t, phi, lambda);   % weights for intgreated ridge
            regression in MATLAB
41
42
43      test_phi = x_test;
44      test_t = y_test;
45
46      predictions_reg = test_phi * w_reg; % predictions for regularized version
47      predictions_noreg = test_phi * w_noreg; % predictions for non-regularized
```

```
        version
48      predictions_reg_matlab = test_phi * w_reg_matlab;   % predictions for matlab
            ridge regression
49
50      MSE_reg = mean((predictions_reg - test_t).^2);
51      MSE_noreg = mean((predictions_noreg - test_t).^2);
52      MSE_reg_matlab = mean((predictions_reg_matlab - test_t).^2);
53
54      fprintf('Final MSE for non-regularized variant: %f\n', MSE_noreg);
55      fprintf('Final MSE for ridge regularized variant: %f\n', MSE_reg);
56      fprintf('Final MSE for integrated MATLAB ridge regularized variant: %f\n',
            MSE_reg_matlab);
57  end
58
59  %% Function for performing cross validation on the training data
60  function[best_lambda] = cross_validate(x, y, lambda_vals, k, normalization_type)
61      fprintf('Starting Cross-validation....')
62      pause(3)    % Just for dramatic effect
63      indices = crossvalind('Kfold',y,k); % generating indices for different folds
            in K-fold cross-validation
64      best_lambda = -1;
65      best_lambda_MSE = inf;
66      num_feat = size(x,2);
67      num_lambdas = size(lambda_vals,2);
68      lambda_MSE = zeros(1, num_lambdas);
69
70      for lambda_idx = 1:length(lambda_vals)
71          cur_lambda = lambda_vals(lambda_idx);
72
```

```
73          fprintf('\n==========\n');
74          fprintf('Current Lambda Value: %f', cur_lambda);
75          fprintf('\n==========\n');
76
77          cur_lambda_MSE = 0;
78
79          for i = 1:k
80              % cross-validation begins
81              test_indices = (indices == i);  % the test fold is marked with i
82              train_indices = ~test_indices;  % all the other folds are used for
                    training
83
84              phi = x(train_indices,:);
85              t = y(train_indices, :);
86
87              test_phi = x(test_indices,:);
88              test_t = y(test_indices,:);
89
90              w = (phi' * phi + cur_lambda * eye(num_feat, num_feat))^-1 * (phi' *
                    t);    % weights for ridge regression
91
92              predictions = test_phi * w;
93
94              cur_MSE = mean((predictions - test_t).^2);  % calculate MSE for
                    current fold
95              cur_lambda_MSE = cur_lambda_MSE + cur_MSE;
96
97              fprintf('MSE for fold %d: %f\n', i, cur_MSE);
98          end
```

```
99
100         cur_mean_MSE = cur_lambda_MSE/k;
101         lambda_MSE(1, lambda_idx) = cur_mean_MSE;
102
103         fprintf('Mean MSE for Lambda=%f: %f', cur_lambda, cur_mean_MSE);
104         fprintf('\n==========================================\n');
105
106         if(cur_mean_MSE < best_lambda_MSE)
107             best_lambda_MSE = cur_mean_MSE;
108             best_lambda = cur_lambda;
109         end
110     end
111
112     % Plotting MSE vs Lambda curve
113     x=log10(lambda_vals);
114     plot(x, lambda_MSE)
115     xlabel('log10(\lambda)')
116     ylabel('MSE')
117     xticklabels(x)
118     title({'Cross-validation MSE for different values of \lambda', strcat('Type
            of Normalization: ', normalization_type)})
119     saveas(gcf, strcat('MSE_Plots_',normalization_type,'.jpg'));
120 end
121
122 %% helper function to compute train and test MSE
123 function[train_MSE, test_MSE] = compute_error(x_train, y_train, x_test, y_test,
        lambda)
124     num_feat = size(x_train,2);
125
```

```
126     phi = x_train;
127     t = y_train;
128
129     test_phi = x_test;
130     test_t = y_test;
131
132     w = (phi' * phi + lambda * eye(num_feat, num_feat))^-1 * (phi' * t);
133
134     train_predictions = phi * w;
135     test_predictions = test_phi * w;
136
137     train_MSE = mean((train_predictions - t).^2);
138     test_MSE = mean((test_predictions - test_t).^2);
139 end
```