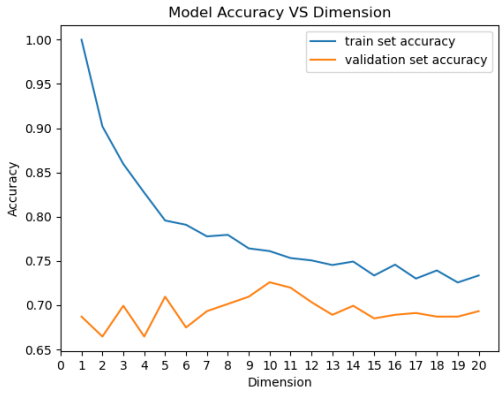
Qien Song CSC311H HW 1

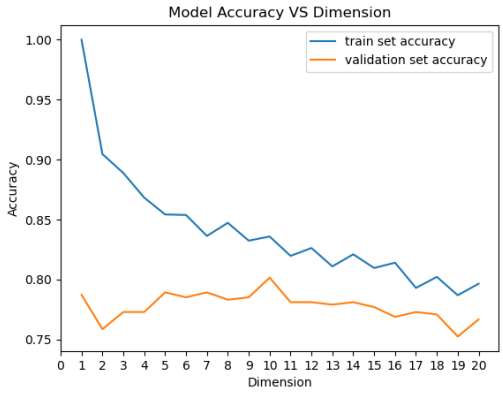
1. b)



Optimal Dimension: 10

Test accuracy: 0.666

c)



Optimal Dimension: 10

Test accuracy: 0.743

Consider x, y as two vectors

cosine similarity

$$sim(x,y) = \frac{x^T y}{|x||y|} = \cos\theta$$

θ is the angle between the vectors

Many high dimension data have intrinsically lower dimensions. In this dataset, most elements in the design matrix are zero. Cosine similarity always yields a value between -1 and 1, which solves the pitfall of the "curse of higher dimensionality", Using Euclidean distance as in b) means that points are further apart and lead to lower accuracy.

2.

a) $w_j = w_j - \alpha \frac{\partial (J+R)}{\partial w_j}$ (where α is the given learning rate)

$= w_j - \alpha \frac{\partial J}{\partial w_j} - \alpha \frac{\partial R}{\partial w_j}$

$= w_j - \alpha \beta_j w_j - \alpha \frac{\partial J}{\partial w_j}$

$= (1 - \alpha \beta_j) w_j - \alpha \frac{\partial J}{\partial w_j}$

$b = b - \alpha \frac{\partial (J+R)}{\partial b}$

$= b - \alpha \frac{\partial J}{\partial b} - \alpha \frac{\partial R}{\partial b}$

$= b - \alpha \frac{\partial J}{\partial b}$

The reason that the weight decays is because in every iteration we multiply w_j by $1 - \alpha \beta_j$, which makes the weight factor smaller each iteration (comparing with non-regularized version), achieving the objective of penalizing large values of w.

Note that the bias term is not decayed because it only acts as an offset.

b) $\frac{\partial J_\beta^{reg}}{\partial w_j} = \frac{\partial (J+R)}{\partial w_j}$

$= \frac{1}{N} \sum_{i=1}^{N} x_j^{(i)} \left(\sum_{j'=1}^{D} x_{j'}^{(i)} w_{j'} - t^{(i)} \right)' + \beta_j w_j$

$= \frac{1}{N} \sum_{i=1}^{N} x_j^{(i)} \sum_{j'=1}^{D} x_{j'}^{(i)} w_{j'} - \frac{1}{N} \sum_{i=1}^{N} x_j^{(i)} t^{(i)} + \beta_j w_j$

$= \sum_{j'=1}^{D} \left(\frac{1}{N} \sum_{i=1}^{N} x_j^{(i)} x_{j'}^{(i)} \right) w_{j'} + \beta_j w_j - \frac{1}{N} \sum_{i=1}^{N} x_j^{(i)} t^{(i)}$

$= \sum_{j'=1}^{D} \left(\frac{1}{N} \sum_{i=1}^{N} x_j^{(i)} x_{j'}^{(i)} + \frac{\beta_j w_j}{D w_{j'}} \right) w_{j'} - \frac{1}{N} \sum_{i=1}^{N} x_j^{(i)} t^{(i)}$

Instead of evenly distribute $\beta_j w_j$ across all dimensions, we can define a term Γ s.t.

$\Gamma = \beta_j$ iff $j = j'$

and $\Gamma = 0$ otherwise

I can briefly demonstrate that the summation by using Γ is the same as using $\frac{\beta_j w_j}{D w_{j'}}$

For $\frac{\beta_j w_j}{D w_{j'}}$, we know that taking this term out of the $\sum_{j'=1}^{D}$ summation sequence would equate the regularization factor to $\beta_j w_j$

For Γ, we know that $j = j'$ only once throughout

$\sum_{j'=1}^{D}$ iteration, so the term is equal to $\beta_j w_{j'}$ when $j=j'$, thus I know the transformation I used is equal

Therefore

$$A_{jj'} = \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)} x_{j'}^{(i)} + \Gamma \qquad \text{where } \Gamma = \beta_j \text{ if } j=j'$$
$$\Gamma = 0 \quad \text{otherwise}$$

$$c = \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)} t^{(i)}$$

c)

$$\frac{\partial J_\beta^{reg}}{\partial w_j} = \sum_{j'=1}^{D}\left(\frac{1}{N}\sum_{i=1}^{N} x_j^{(i)} x_{j'}^{(i)} + \Gamma\right) w_{j'} - \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)} t^{(i)} = 0$$

We can vectorize c as $\frac{1}{N} X^T t$

And for A we can vectorize $\frac{1}{N}\sum_{i=1}^{N} x_j^{(i)} x_{j'}^{(i)}$ as $\frac{1}{N} X^T X$ and Γ as $diag(\vec{\beta})$ where $\vec{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_D \end{bmatrix}$ is the regularization vector and $diag(\beta)$

$$= \begin{bmatrix} \beta_1 & 0 & \cdots & 0 \\ 0 & \beta_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \beta_D \end{bmatrix}$$

Therefore $A = \frac{1}{N} X^T X + \text{diag}(\vec{\beta})$

$C = \frac{1}{N} X^T t$

Therefore the closed form solution is

$$\left[\frac{1}{N} X^T X + \text{diag}(\vec{\beta})\right] \vec{w} - \frac{1}{N} X^T t = 0$$

$$\left[X^T X + N\,\text{diag}(\vec{\beta})\right] \vec{w} = X^T t$$

$$\vec{w} = \left(X^T X + N\,\text{diag}(\vec{\beta})\right)^{-1} X^T t$$

3.

① $y = Xw + b\mathbf{1}$, $\mathbf{1}$ is a vector of all ones

② $J = \frac{1}{N}\sum_{i=1}^{N} 1 - \cos(y^{(i)} - t^{(i)})$

$$\frac{\partial J}{\partial y_i} = \frac{1}{N} \cdot (\sin(y^{(i)} - t^{(i)})$$

$$\therefore \frac{\partial J}{\partial y} = \frac{1}{N}\sin(y - t)$$ (a $N \times 1$ vector, where N is data size)

③ $J = \frac{1}{N}\sum_{i=1}^{N} 1 - \cos(y^{(i)} - t^{(i)}) = \frac{1}{N}\sum_{i=1}^{N} 1 - \cos(w^T x^{(i)} + b - t^{(i)})$

$$\frac{\partial J}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N} \underbrace{x_{ij}^T}_{1\times D} \underbrace{\sin(w^T x^{(i)} + b - t^{(i)})}_{D \times 1}.$$

x_{ij} denotes j^{th} column

$x^{(i)}$ denotes i^{th} row

$$\therefore \frac{\partial J}{\partial w} = \frac{1}{N} X^T \sin(Xw + b - t)$$

④ $\frac{\partial J}{\partial b} = \frac{\partial J}{\partial y} \cdot \frac{\partial y}{\partial b}$ since $\frac{\partial y}{\partial b} = 1$

$$= \frac{\partial J}{\partial y} = \frac{1}{N}\sin(y - t) = \frac{1}{N}\sin(Xw + b - t)$$

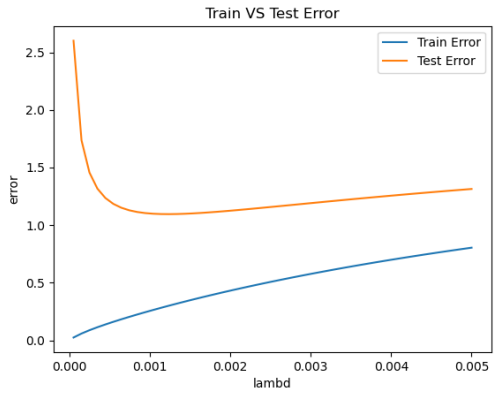
4.

b) Arguments for cross validation: train data, sequence of lambda values and number of folds.

Procedure:

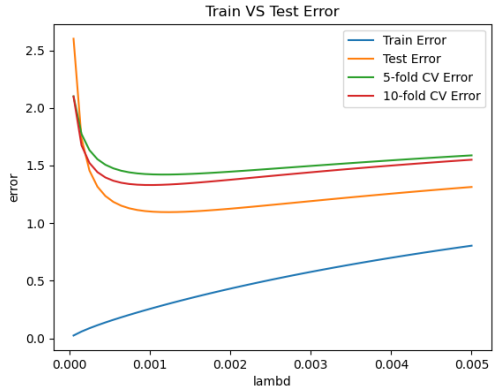
1. Shuffle the data
2. Select a lambda value from the sequence
3. Split the data into k folds
4. Iterate through each fold (use each fold as validation)
   ① Train the model based on non-validation data
   ② Apply the weights on validation fold to get error
   ③ Record the CV error
5. Record and save the mean of CV errors as the CV error of the lambda value.
6. Repeat 2-5 until all lambda values have been selected
7. We obtain a sequence of CV error corresponding to each lambda value.

c)



The training and test error is shown in the plot.

d)



Optimal λ for 5-fold:

0.001139

Optimal λ for 10-fold:

0.00104

We significantly overfit on train sample because the number of features ($D=4000$) exceed the number of samples and this means that without regularization, the training error will be 0. (By basic linear algebra, we are guaranteed a solution $Xw = t$ when X is an $N \times D$ matrix and $N < D$)

With regularization penalty, we see that training error increases over lambd.

We also see that test error is extremely large with small lambda, showing the problem of overfitting as performance on test set is extremely poor. Similarly, for lambda = 0, CV error is extremely high.

We see that CV error and test error reach its minimum when lambda is around 0.001 and gradually increases afterwards. This is because for larger values of lambda it penalizes too much on the weight parameter.

So for small values of lambda, we haven't penalized enough on weight parameters and for large values we penalize too much.

We also see that the error of 10-fold is slightly less than 5-fold. That is because in each iteration, we get to train more samples (for 10-fold, we train $\frac{9}{10}$ of the training set each iteration). This leads to lower CV error as the model inherently has trained on more data.