Pattern Recognition and Machine Learning

Assignment Report

submitted by

Karthikeya P (CS22B026)

Under the supervision

of

Dr.Arun Rajkumar

Assistant Professor

Computer Science and Engineering

Indian Institute of Technology Madras

Jan - May 2024

Contents

List of Figures

1 Question 1

1.1 Part i

Data consists of 400 data points $x_i \in \{0,1\}^{50}$, so one of the natural ways of modelling the generator of this data is having Bernoulli random variable for each mixture. Each mixture has a probability vector $\mu \in \mathbb{R}^{50}$ and the probability of point x coming from that mixture is given by $P(x|\mu) = \prod_{i=1}^{50} \mu_i^{x_i}.(1-\mu_i)^{1-x_i}$. The Likelihood function with the above model will be as follows

$$
\begin{aligned}
L\begin{pmatrix} \pi_1,\, \pi_2,\, \ldots,\, \pi_k \\ \mu_1,\, \mu_2,\, \ldots,\, \mu_k \\ x_1,\, x_2,\, \ldots,\, x_n \end{pmatrix} &= \prod_{i=1}^{n} \left\{ f\begin{pmatrix} \pi_1,\, \pi_2,\, \ldots,\, \pi_k \\ \mu_1,\, \mu_2,\, \ldots,\, \mu_k \end{pmatrix}, x_i \right\} \right\} \\
&= \prod_{i=1}^{n} \left\{ \sum_{r=1}^{k} \pi_r.P(x_i|\mu_r) \right\} \\
&= \prod_{i=1}^{n} \left\{ \sum_{r=1}^{k} \pi_r \left(\prod_{d=1}^{50} \mu_{rd}^{x_{id}}.(1-\mu_{rd})^{1-x_{id}} \right) \right\}
\end{aligned}
$$

Applying log on both sides

$$
\log L = \sum_{i=1}^{n} \log \left\{ \sum_{r=1}^{k} \pi_r \left(\prod_{d=1}^{50} \mu_{rd}^{x_{id}}.(1-\mu_{rd})^{1-x_{id}} \right) \right\}
$$

Now add k new parameters $\lambda_r^i, r \in \{1,2,\ldots,k\}, i \in \{1,2,\ldots,n\}$ for each of the data points x_i such that $\sum_{r=1}^{k} \lambda_r^i = 1 \;\forall\; i$

$$
\begin{aligned}
\log L &= \sum_{i=1}^{n} \log \left\{ \lambda_r^i \frac{\sum_{r=1}^{k} \pi_r \left(\prod_{d=1}^{50} \mu_{rd}^{x_{id}}.(1-\mu_{rd})^{1-x_{id}} \right)}{\lambda_r^i} \right\} \\
&\geq \sum_{i=1}^{n} \sum_{r=1}^{k} \lambda_r^i \log \left\{ \frac{\sum_{r=1}^{k} \pi_r \left(\prod_{d=1}^{50} \mu_{rd}^{x_{id}}.(1-\mu_{rd})^{1-x_{id}} \right)}{\lambda_r^i} \right\}
\end{aligned}
$$

- By fixing $\lambda_r^i \forall\; i,r$ one can find closed form optimal $\pi_r, \mu_r \forall\; r$
- Similarly by fixing $\pi_r, \mu_r \forall\; r$ one can find closed form optimal $\lambda_r^i \forall\; i,r$.
- The below is the derivation for finding optimal μ_{rd}
  - $\frac{\partial L(\pi,\mu,\lambda,x)}{\partial \mu_{rd}} = 0$
  - $\frac{\partial L(\pi,\mu,\lambda,x)}{\partial \mu_{rd}} = \sum_{i=1}^{n} \frac{\lambda_r^i x_{id}}{\mu_{rd}} - \frac{\lambda_r^i (1-x_{id})}{1-\mu_{rd}} = 0$
  - $\mu_{rd} = \frac{\sum_{i=1}^{n} \lambda_r^i.x_{id}}{\sum_{i=1}^{n} \lambda_r^i} \Rightarrow \mu_r = \frac{\sum_{i=1}^{n} \lambda_r^i.x_i}{\sum_{i=1}^{n} \lambda_r^i}$

- The closed form optimal solutions after fixing the necessary parameters are as follows:
  - The optimal λ_r^i for given π, μ is given by $\boxed{\lambda_r^i = \dfrac{\pi_r.P(x_i|\mu_r)}{\sum_{j=1}^{k} \pi_j.P(x_i|\mu_j)}}$
  - The optimal π_r for given λ is given by $\boxed{\pi_r = \dfrac{\sum_{i=1}^{n} \lambda_r^i}{n}}$
  - The optimal μ_r for given λ is given by $\boxed{\mu_r = \dfrac{\sum_{i=1}^{n} \lambda_r^i.x_i}{\sum_{i=1}^{n} \lambda_r^i}}$

Now maximizing the parameters alternatively by fixing the necessary parameters gives the optimal generation (sometimes may be sub-optimal if converges to local optimum). By performing this algorithm for 100 random initialisations of π, μ, λ for 20 iterations the plot of loglikelihood vs iterations averaged over the initialisations looks as follows.

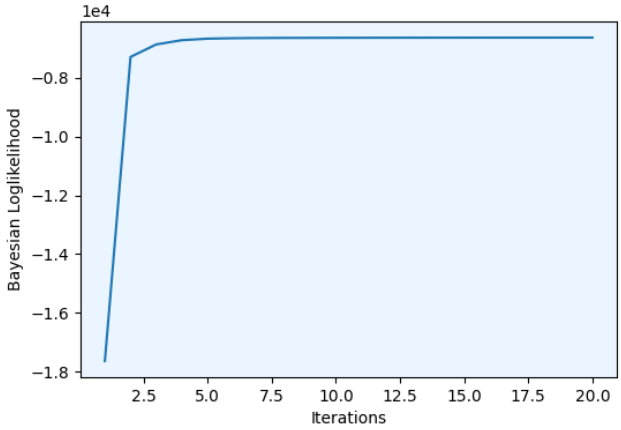

Figure 1: Loglikelihood vs iterations of the Bayesian EM algorithm over 100 initialisations

1.2 Part ii

Now modelling the mixtures as multi-dimensional gaussian distribution with mean $\mu \in \mathbb{R}^{50}$ and covariance matrix as $\sigma \in \mathbb{R}^{50\times50}$. The pdf of point x coming from the mixture with given μ, σ is given by

$$p(x, \mu, \sigma) = \frac{1}{(2\pi)^{25}|\sigma|^{0.5}} e^{-\frac{(x-\mu)\sigma^{-1}(x-\mu)^T}{2}}$$

The likelihood function with the above model will be as follows

$$L\begin{pmatrix} \pi_1,\ \pi_2,\ \ldots,\ \pi_k \\ \mu_1,\ \mu_2,\ \ldots,\ \mu_k \\ \sigma_1,\ \sigma_2,\ \ldots,\ \sigma_k \\ x_1,\ x_2,\ \ldots,\ x_n \end{pmatrix} = \prod_{i=1}^{n}\left\{ f\begin{pmatrix} \pi_1,\ \pi_2,\ \ldots,\ \pi_k \\ \mu_1,\ \mu_2,\ \ldots,\ \mu_k \\ \sigma_1,\ \sigma_2,\ \ldots,\ \sigma_k \end{pmatrix}, x_i \right\}$$

$$= \prod_{i=1}^{n}\left\{ \sum_{r=1}^{k} \pi_r . p(x_i, \mu_r, \sigma_r) \right\}$$

$$= \prod_{i=1}^{n}\left\{ \sum_{r=1}^{k} \pi_r . \frac{1}{(2\pi)^{25}|\sigma_r|^{0.5}} e^{-\frac{(x-\mu_r)\sigma_r^{-1}(x-\mu_r)^T}{2}} \right\}$$

Applying log on both sides

$$\log L = \sum_{i=1}^{n} \log \left\{ \sum_{r=1}^{k} \pi_r \left(\frac{1}{(2\pi)^{25}|\sigma_r|^{0.5}} e^{-\frac{(x-\mu_r)\sigma_r^{-1}(x-\mu_r)^T}{2}} \right) \right\}$$

Now add k new parameters $\lambda_r^i, r \in \{1, 2, \ldots, k\}, i \in \{1, 2, \ldots, n\}$ for each of the data points x_i such that $\sum_{r=1}^{k} \lambda_r^i = 1 \ \forall \ i$

$$\log L = \sum_{i=1}^{n} \log \left\{ \lambda_r^i \frac{\sum_{r=1}^{k} \pi_r \left(\frac{1}{(2\pi)^{25}|\sigma_r|^{0.5}} e^{-\frac{(x-\mu_r)\sigma_r^{-1}(x-\mu_r)^T}{2}} \right)}{\lambda_r^i} \right\}$$

$$\geq \sum_{i=1}^{n}\sum_{r=1}^{k} \lambda_r^i \log \left\{ \frac{\sum_{r=1}^{k} \pi_r \left(\frac{1}{(2\pi)^{25}|\sigma_r|^{0.5}} e^{-\frac{(x-\mu_r)\sigma_r^{-1}(x-\mu_r)^T}{2}} \right)}{\lambda_r^i} \right\}$$

- By fixing $\lambda_r^i \ \forall \ i, r$ one can find closed form optimal $\pi_r, \mu_r, \sigma_r \forall \ r$
- Similarly by fixing $\pi_r, \mu_r, \sigma_r \forall \ r$ one can find closed form optimal $\lambda_r^i \forall \ i, r$.
- The closed form optimal solutions after fixing the necessary parameters are as follows:
  - The optimal λ_r^i for given π, μ, σ is given by $\boxed{\lambda_r^i = \frac{\pi_r . p(x_i, \mu_r, \sigma_r)}{\sum_{j=1}^{k} \pi_j . P(x_i, \mu_j, \sigma_j)}}$
  - The optimal π_r for given λ is given by $\boxed{\pi_r = \frac{\sum_{i=1}^{n} \lambda_r^i}{n}}$
  - The optimal μ_r, σ_r for given λ is given by $\boxed{\mu_r = \frac{\sum_{i=1}^{n} \lambda_r^i . x_n}{\sum_{i=1}^{n} \lambda_r^i}, \sigma_r = \frac{\sum_{i=1}^{n} \lambda_r^i . (x_n - \mu_r)^2}{\sum_{i=1}^{n} \lambda_r^i}}$

Now maximizing the parameters alternatively by fixing the necessary parameters gives the optimal generation (sometimes may be sub-optimal if converges to local optimum). By performing this algorithm for 100 random initialisations of $\pi, \mu, \sigma, \lambda$ for 20 iterations the plot of loglikelihood vs iterations averaged over the initialisations looks as follows.

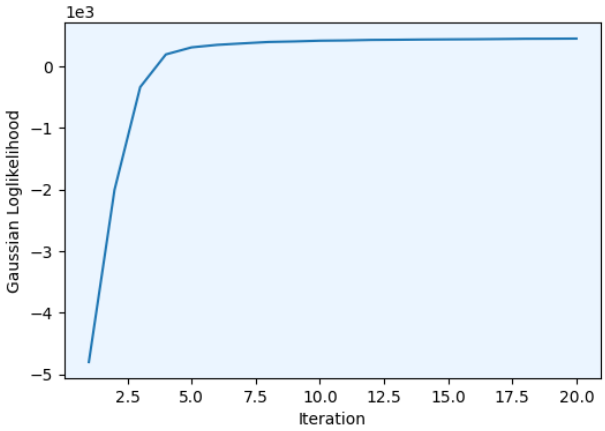

Figure 2: Loglikelihood vs iterations of the Gaussian EM algorithm over 100 initialisations

1.3 Part iii

One can also use the K-means algorithm to do the clustering of the data. After performing K-means for the given data with 100 random initialisations. The plot of averaged K-means error over the initialisations wrt iterations is as follows.

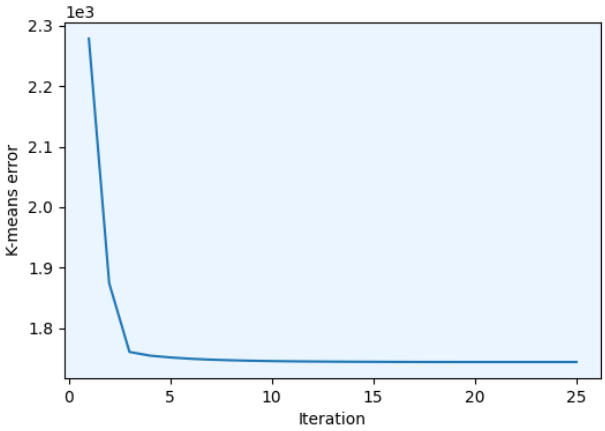

Figure 3: K-means error vs Iterations averaged over 100 initialisations

1.4 Part iv

Among the three implemented algorithms I feel the first algorithm works better as the data given is binary and if we try to model the given data using gaussian it might not give good results as the gaussian model assumes that the data is gaussian distributed which is not true in our case. So the better should be Bernoulli EM or K-means algorithm which works bettter will depend on the distribution of the data in the n-dimensional space. In this case both bernoulli and K-means are both good as if we observe the K-means error in both the cases they are around 1750 where as the error for gaussian distribution is vary large. So Bernoulli is the one that best suits the given data set.

2 Question 2

2.1 Part i

The best fit linear approximation for the given data is $w_{ML} = (XX^T)^{-1}XY$ where $X \in \mathbb{R}^{d \times n}$, $Y \in \mathbb{R}^n$ and $w \in \mathbb{R}^d$. X is the data matrix, where every column represents a data point, Y is the label matrix containing all labels for the data points. One can add the bias and perform the ridge regression best fit analytically for a given λ using the following $w_R = (XX^T + \lambda I)^{-1}XY$. Following plots shows the train and test error variation with λ

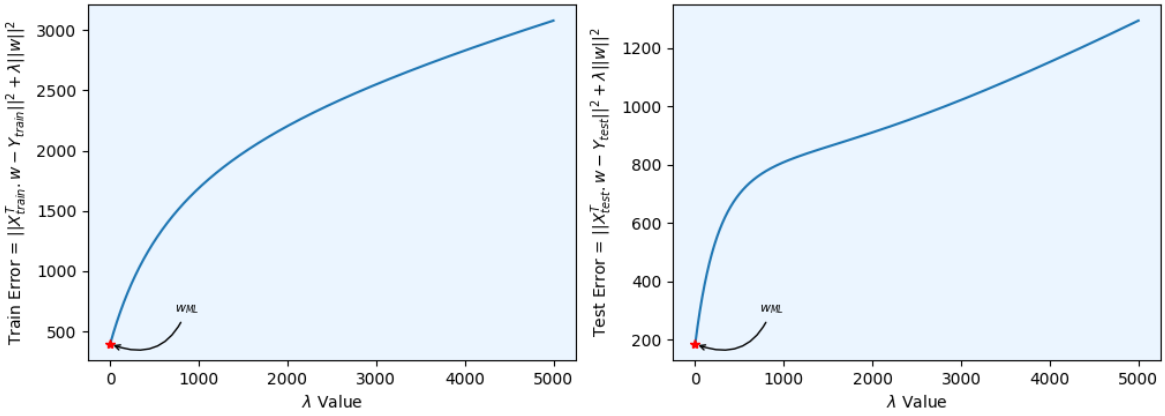

Figure 4: Train and Test error variation with λ

2.2 Part ii

The calculation of the inverse of a matrix is computationally costly so we want to avoid it. Since we know the objective function and it is uncontrolled optimisation we can perform gradient descent.

$$f(w) = ||X^T w - Y||^2$$
$$\nabla f(w) = 2X(X^T w - Y)$$
$$w^{t+1} = w^t - \eta^t . \nabla f(w)$$

After performing the gradient descent for 1000 iterations with step size of 10^{-6} the following plots are observed for separation between w, w_{ML}, test error and train error with respect to iterations.

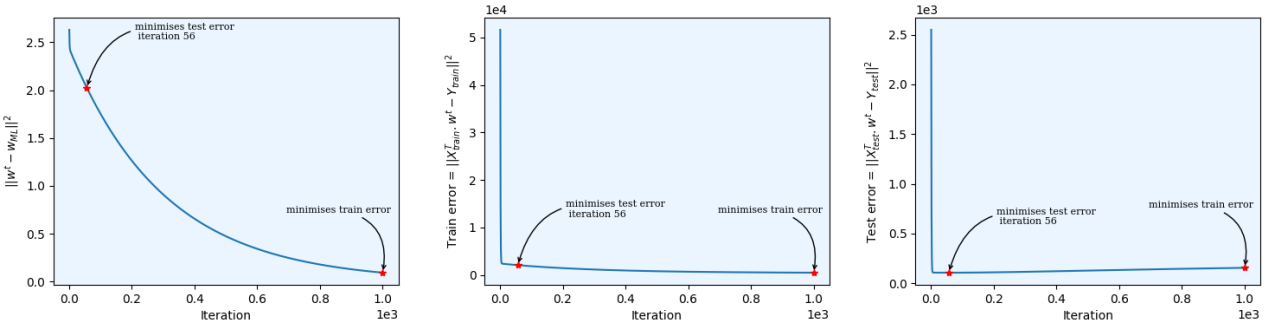

Figure 5: Train and Test errors vs iterations of Gradient Descent

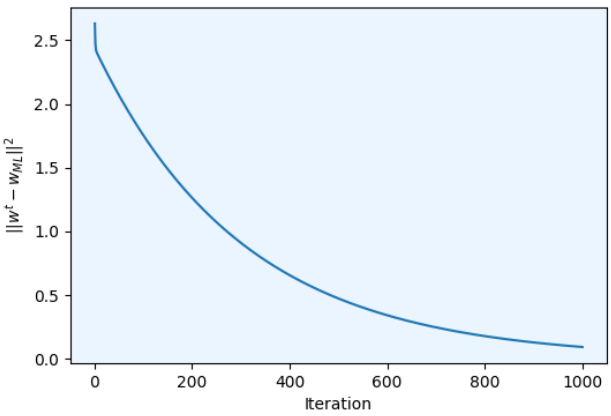

Figure 6: Separation between w, w_{ML} vs Iterations with step size 10^{-6}

2.3 Part iii

We can also perform the stochastic gradient descent to get the optimal w as the dataset is very large with 10000 data points. The idea is at each iteration of the gradient descent we take a random sample of points from entire data set and pretend that this is the new total dataset and perform gradient descent and this converges to the optimum with very high probability. Here we use a batch size of 100. The below plot shows the seperation between w, w_ML, train error, test error for different step sizes.

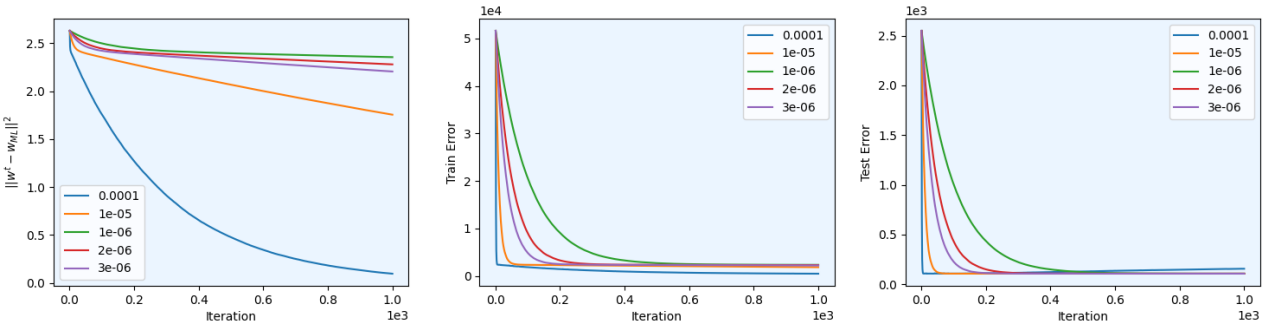

Figure 7: Train and Test errors vs iterations of Stochastic Grad-Desc for diff step sizes

Few observations that are made from the above plots are

- The first plot shows that despite the random choice of sample data w is moving towards w_{ML}. This shows that the individual small sample sets are nicely representing the overall dataset and the dataset is most likely aligned towards w_{ML}
- If we observe the second plot this represents the first plot approaching w_{ML} in terms of decreasing train error. For different step values the rate of convergence might be different but all of them converged close to w_{ML}
- The third plot shows the variation of test error with iterations is intially decreasing but when it is closely approaching to the w_ML there is a increase in the test error
- This increase indicates that the direction which best represents the train data need not be the best for the test data.
- Observing both the second and the third plots the step size 10^{-6} will work best for both test and the train data. where as 10^{-4} best works for the train data which can also be observed how close is that to w_{ML} from first plot.

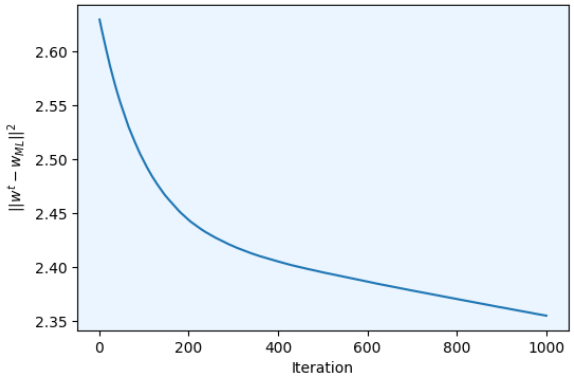

Figure 8: Separation between w, w_{ML} vs Iterations with step size 10^{-6}

2.4 Part iv

The best w that fits the train data need not be the best for test data and the total data at the same time. So to take that into account we whether there is any best fit for some λ such that $(XX^T + \lambda I)XY$ such that it minimizes the test data. The error function is given by $||X^T w - Y||^2 + \lambda||w||^2$. For calculating this we use the gradient descent algorithm on this function to find the best ridge w_R.

$$f(w) = ||X^T w - Y||^2 + \lambda||w||^2$$
$$\nabla f(w) = 2X(X^T w - Y) + 2\lambda w$$
$$w^{t+1} = w^t - \eta^t.\nabla f(w)$$

Performing the above algorithm by taking 80% of the train data for training and 20% for cross validation of λ. We get the following plots for the train, validation and test errrors.

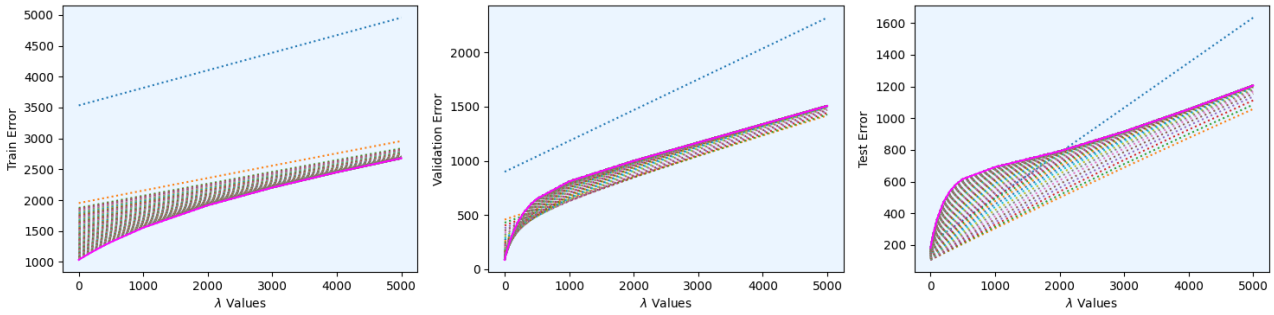

Figure 9: Train, Validation, Test errors vs λ as iterations progress

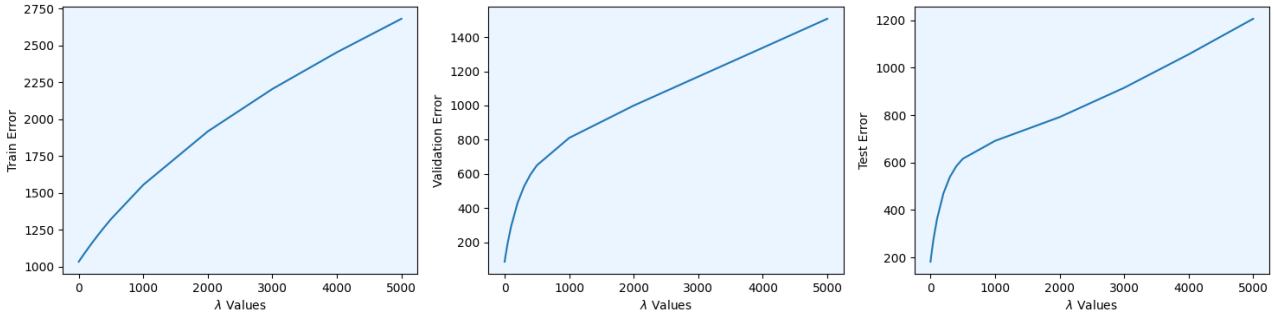

Figure 10: Train, Validation, Test errors vs λ after 1000 steps

The dotted lines are plotted wrt iterations to indicate the movement of errors.

A few observations that can be made from the above plots are summarised below:

- The first plot shows that the test error is always decreasing indicating the purpose of the gradient descent.
- We can observe the increase of train error with λ which should happen as the train data gets the minimal error at w_{ML}
- The second plot indicates the cross validation error with iteration and we can see that there are few better w that estimates the validation data after certain values of λ.
- Even then for this data the w_ML is having the best fit for the validation data as well.
- The 3rd plot shows that there are some better estimate for test data which is not obtained from w_{ML} or w_R. But this is not a good estimator because it won't be a good estimate for the train data.
- So the best estimator is something which works well on the train data as well as the test data. Here in this case that is w_{ML}
- In the 3 plots there is a sudden jump in the dotted lines indicating the first step as this will be the first time moving from completely random initial value irrespective of data towards the minimum. Which indicate the high value of gradient.

So for this data the best fit is w_{ML} and none of the w_R works better than that. This can be observed from both the plots Fig:4 and Fig:10 where the w_{ML}, w_R are calculated analytically and using gradient descent as well. This might be happening because there is a very good fit between the test and the train data giving the same estimator of train data w_{ML} to be the best fit for the test data as well.