UNIVERSITÉ Concordia UNIVERSITY

COMP 6321 – Machine Learning

Assignment 1

Pablo Arevalo Escobar
pescobar24@yahoo.com
40081955

Q1. PROBABILITY AND BAYES RULE (10 Points)

Known Probabilities:

P (B_1 = G) = 0.9

P (B_1 = M) = 0.1

P (B1 = A) = 0.0

	B_2 G	**B_2 M**	**B_2 A**
B_1 G	1.0	0.0	0.0
B_1 M	0.1	0.9	0.0
B_1 A	0.2	0.3	0.5

(a)

P (B_2 = G) given the above.

Making use of marginal distribution:

P (B_2 = G) = $\sum_{y \in \text{Val}(y)} P(B_2 = G, B_1 = y)$

P (B_2 = G) = P (B_2 = G, B_1 = G) + P (B_2 = G, B_1 = M) + P (B_2 = G, B_1 = A)

P (B_2 = G, B_1 = G) = P (B_2 = G | B_1 = G) * P (B_1 = G)

= 1.0 * 0.9

= 0.9

P (B_2 = G, B_1 = M) = P (B_2 = G | B_1 = M) * P (B_1 = M)

= 0.1 * 0.1

= 0.01

P (B_2 = G) = 0.9 + 0.01 + 0

= 0.91

(b)

P (B_1 = G | B_2 = G) = $\frac{P(B_2=G \mid B_1=G)\, P(B_1=G)}{P(B_2=G)}$

= $\frac{1.0*0.9}{0.91}$

= 0.989

= 0.99

Q2. MLE AND MAP (15 points)

(a)

Derivation of the MLE for the Poisson distribution:

$$L(k\,|\lambda) = \prod_{j=1}^{n} \frac{\lambda^{k_j} * e^{-\lambda}}{k_j!}$$

$$l(k\,|\lambda) = \ln(\prod_{j=1}^{n} \frac{\lambda^{k_j} * e^{-\lambda}}{k_j!})$$

$$= \sum_{j=1}^{n} \ln(\frac{\lambda^{k_j} e^{-\lambda}}{k_j!})$$

$$= \sum_{j=1}^{n} \ln(\lambda^{k_j} e^{-\lambda}) - \ln(k_j!)$$

$$= \sum_{j=1}^{n} k_j \ln(\lambda) - \lambda \ln(e) - \ln(k_j!)$$

$$l(k\,|\lambda) = \sum_{j=1}^{n} k_j \ln(\lambda) - \lambda - \ln(k_j!)$$

Taking the derivative:

$$\frac{\partial l(k\,|\lambda)}{\partial \lambda} = \sum_{j=1}^{n} \frac{k_j}{\lambda} - 1$$

Finding Maxima:

$$0 = \sum_{j=1}^{n} \frac{k_j}{\lambda} - 1$$

$$0 = -n + \sum_{j=1}^{n} \frac{k_j}{\lambda}$$

$$n = \sum_{j=1}^{n} \frac{k_j}{\lambda}$$

$$\lambda n = \sum_{j=1}^{n} k_j$$

$$\lambda_{MLE} = \frac{1}{n} \sum_{j=1}^{n} k_j$$

Intuitive Result:

The MLE for the Poisson distribution gives us the sample mean of the distribution.

(b)

Derivation of the MAP for the Gamma distribution:

$$P(\lambda|k_{1:n}) \propto P(k_{1:n}\,|\lambda) * P(\lambda)$$

$$P(\lambda|k_{1:n}) = \frac{\lambda^{\alpha-1}\, e^{-\frac{\lambda}{\beta}}}{\Gamma(\alpha)\beta^{\alpha}} * \prod_{j=1}^{n} \frac{\lambda^{k_j} * e^{-\lambda}}{k_j!}$$

$$P(\lambda|k_{1:n}) = \prod_{j=1}^{n} \frac{\lambda^{k_j+\alpha-1} * e^{-\lambda-\frac{\lambda}{\beta}}}{k_j!\,\Gamma(\alpha)\beta^{\alpha}}$$

Taking the log:

$$\ln\left(P(\lambda|k_{1:n})\right) = \ln\left(\prod_{j=1}^{n} \frac{\lambda^{k_j+\alpha-1} * e^{-\lambda-\frac{\lambda}{\beta}}}{k_j!\,\Gamma(\alpha)\beta^{\alpha}}\right)$$

$$= \sum_{j=1}^{n} \ln\left(\frac{\lambda^{k_j+\alpha-1} * e^{-\lambda-\frac{\lambda}{\beta}}}{k_j!\,\Gamma(\alpha)\beta^{\alpha}}\right)$$

$$= \sum_{j=1}^{n} \ln\left(\lambda^{k_j+\alpha-1} * e^{-\lambda-\frac{\lambda}{\beta}}\right) - \ln\left(k_j!\,\Gamma(\alpha)\beta^{\alpha}\right)$$

$$= \sum_{j=1}^{n} \ln\left(\lambda^{k_j+\alpha-1}\right) + \ln\left(e^{-\lambda-\frac{\lambda}{\beta}}\right) - \ln\left(k_j!\,\Gamma(\alpha)\beta^{\alpha}\right)$$

$$= \sum_{j=1}^{n} (k_j+\alpha-1)\ln(\lambda) + \left(-\lambda - \frac{\lambda}{\beta}\right) - \ln\left(k_j!\,\Gamma(\alpha)\beta^{\alpha}\right)$$

Taking the derivative:

$$\frac{\partial P(\lambda|k_{1:n})}{\partial \lambda} = \sum_{j=1}^{n} \frac{k_j+\alpha-1}{\lambda} - 1 - \frac{1}{\beta}$$

$$0 = -n - \frac{n}{\beta} + \frac{1}{\lambda}\sum_{j=1}^{n} k_j + \alpha - 1$$

$$n + \frac{n}{\beta} = \frac{1}{\lambda}\sum_{j=1}^{n} k_j + \alpha - 1$$

$$\lambda\left(n + \frac{n}{\beta}\right) = \sum_{j=1}^{n} k_j + \alpha - 1$$

$$\lambda\left(\frac{\beta n + n}{\beta}\right) = \sum_{j=1}^{n} k_j + \alpha - 1$$

$$\lambda\left(\frac{n(\beta+1)}{\beta}\right) = \alpha n - n + \sum_{j=1}^{n} k_j$$

$$\lambda = (\alpha n - n + \sum_{j=1}^{n} k_j) * (\frac{\beta}{n(\beta+1)})$$

$$\lambda = \frac{\beta(\alpha n - n + \sum_{j=1}^{n} k_j)}{n(\beta+1)}$$

$$\lambda_{MAP} = \frac{\beta(\alpha - 1 + \frac{1}{n}\sum_{j=1}^{n} k_j)}{\beta+1}$$

(c)

As the number of samples, n, goes to infinity the samples no longer give any weight to λ_{MAP}. This is because the 'weight' of the samples in λ_{MAP} is determined by $\frac{1}{n}$, a value which approaches 0 as n goes to infinity.

At infinity, the value of λ_{MAP} is then a function of the hyper-parameters: α and β. More precisely, it is a function of:

$$\lambda_{MAP} = \frac{\beta(\alpha - 1)}{\beta+1}$$

It should also be noted that λ_{MLE} acts as a weight for λ_{MAP}.

consider the following:

$$\lambda_{MLE} = \frac{1}{n}\sum_{j=1}^{n} k_j$$

$$\lambda_{MAP} = \frac{\beta(\alpha - 1 + \frac{1}{n}\sum_{j=1}^{n} k_j)}{\beta+1}$$

$$\lambda_{MAP} = \frac{\beta(\alpha - 1 + \lambda_{MLE})}{\beta+1}$$

Setting:

$\alpha \approx 1$

$\beta \approx 1$

$$\lambda_{MAP} = \frac{1(1 - 1 + \lambda_{MLE})}{1+1}$$

$$\lambda_{MAP} = \frac{(\lambda_{MLE})}{2}$$

Q3. Ridge regression (5 points)

$$E(w) = (Xw - t)^T(Xw - t) + \lambda w^T w$$

$$\frac{\partial E(w)}{\partial w} = \frac{\partial}{\partial w}[(Xw - t)^T(Xw - t) + \lambda w^T w]$$

$$= \frac{\partial}{\partial w}[(Xw - t)^T(Xw - t)] + \frac{\partial}{\partial w}[\lambda w^T w]$$

Using:

$$\frac{\partial E(w)}{\partial w} = \frac{\partial}{\partial w}\frac{1}{2}[(Xw - t)^T(Xw - t)]$$

$$= \frac{\partial}{\partial w}\, 0.5 * [(Xw - t)^T(Xw - t)]$$

$$= X^T Xw - X^T t$$

$$= (2X^T Xw - 2X^T t) + \frac{\partial}{\partial w}[\lambda w^T w]$$

Using:

$$\frac{dw^T Aw}{dw} = (A + A^T)w$$

$$\lambda w^T w = \lambda w^T Iw$$

$$\lambda \frac{dw^T Iw}{dw} = \lambda(I + I^T)w$$

$$= (\lambda I + \lambda I^T)w$$

$$= \lambda Iw + \lambda I^T w$$

$$= (2X^T Xw - 2X^T t) + (\lambda Iw + \lambda I^T w)$$

$$= (2X^T Xw - 2X^T t) + (2\lambda Iw)$$

$$= 2\lambda Iw + 2X^T Xw - 2X^T t$$

$$\frac{\partial E(w)}{\partial w} = 2w(\lambda I + X^T X) - 2X^T t$$

Optimizing:

$$2w(\lambda I + X^T X) - 2X^T t = 0$$

$$2w(\lambda I + X^T X) = 2X^T t$$

$$2w = (\lambda I + X^T X)^{-1} 2X^T t$$

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

Q4. Robust Linear Regression (10 points)

$$P(t|X,w,b) = \prod_{i=1}^{N} \frac{1}{2b}\, e^{-\frac{|X_i{}^t w - t_i|}{b}}$$

Taking the log

$$\ln\left(P(t|X,w,b)\right) = \ln\left(\prod_{i=1}^{N} \frac{1}{2b}\, e^{-\frac{|x_i{}^T w - t_i|}{b}}\right)$$

$$= \sum_{i=1}^{N} \ln\left(\frac{1}{2b}\, e^{-\frac{|x_i{}^T w - t_i|}{b}}\right)$$

$$= \sum_{i=1}^{N} \ln\left(\frac{1}{2b}\right) + \ln\left(e^{-\frac{|x_i{}^T w - t_i|}{b}}\right)$$

$$= \sum_{i=1}^{N} \ln\left(\frac{1}{2b}\right) - \frac{|x_i{}^T w - t_i|}{b} \ln\left(e^1\right)$$

$$= \sum_{i=1}^{N} \ln\left(\frac{1}{2b}\right) - \frac{|x_i{}^T w - t_i|}{b}$$

$$= n \ln\left(\frac{1}{2b}\right) + \frac{1}{b} \sum_{i=1}^{N} - |x_i{}^T w - t_i|$$

Finding the derivative

$$= \frac{\partial}{\partial w}\left[n \ln\left(\frac{1}{2b}\right) + \frac{1}{b} \sum_{i=1}^{N} - |x_i{}^T w - t_i|\right]$$

$$= \frac{\partial}{\partial w}\left[\frac{1}{b} \sum_{i=1}^{N} - |x_i{}^T w - t_i|\right]$$

$$\boldsymbol{DERIV} = \frac{\partial}{\partial w} - \frac{1}{b}\left[|X^T w - t_i|\right]$$

We are looking for a loss function, $E_{Laplace}$ (**w**) whose minimization is equivalent to finding the MLE of **w** under the Laplacian model. In other words, we are looking for a loss function whose derivative is equivalent when minimalized to the derivative **DERIV** when minimalized.

Since $-\frac{1}{b}$ is a scalar value applied to the whole in **DERIV**, we can ignore it as it could be removed when the derivative is set to 0. Therefore, $E_{Laplace}$(**w**) is a function whose derivative is of the form:

$$\frac{\partial}{\partial w}\left[|X^T w - t_i|\right]$$

And therefore, we can conclude that:

$$E_{Laplace}(w) = |X^T w - t_i|$$

(b)

I believe that the Laplacian loss function provides a more robust fit to the data compared to the Gaussian because of the omission of a square. First we must note that the Laplacian loss function is 2 times the positive square root of the Gaussian loss function.

To elaborate, the Gaussian loss function avoids negative numbers by squaring the difference between the mean and x and halving the value. The Laplacian loss function takes the absolute value of the difference. This means that when we are faced with large outliers, the Gaussian loss function would output a loss which is half the square of the Laplacian. This behaviour results in the Gaussian loss function being more sensitive to outliers as they produce a larger loss.

Example:

Let $x_i^T w - t_i = -100$

Then:

$E_{Laplacian}(w) = |-100|$

$= 100$

$E_{Gaussian}(w) = \frac{1}{2}(-100)^2$

$= 5000$

In this example, for the same outlier, the Gaussian loss is 50 times the Laplacian.

UNIVERSITÉ Concordia UNIVERSITY

COMP 6321 – Machine Learning

Assignment 1: Report

Pablo Arevalo Escobar
40081955

Part 1: Unregularized Polynomial Regression

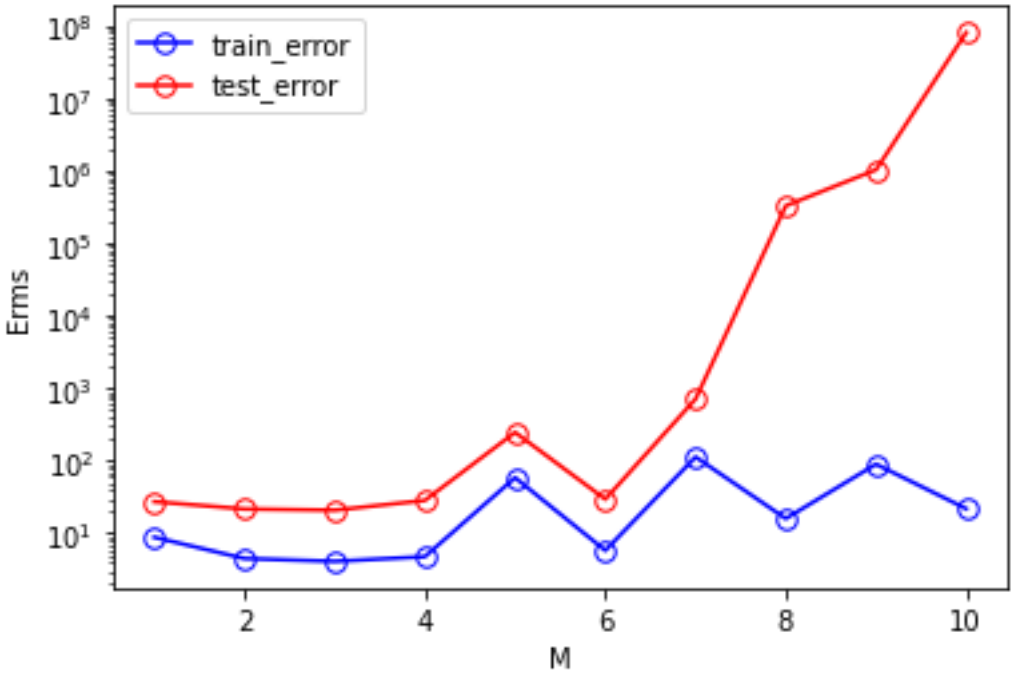

Figure 1: Plot of training and testing error against polynomial degree (M)

On the left side of the plot, we see a glimpse of the effects of underfitting in the M [5-6] region. The model at M =5 is underfitted and has a larger error, however, at M=6 we see that the error begins to trend downwards and fit the data. The question still remains on why the data from M[1-4] performs better than M = 5 if it's underfitted? This is most likely due to the limited size of the data, within the small set of data given, a polynomial with M=1 does a better job at reducing the error than all other polynomials measured apart from M=5. However, given larger data sets it is likely that the non-linear nature of the data will be more evident and therefore M=1 will produce a larger error than M[2-4].

On the right side of the plot M [7-10], we see the effects of overfitting as expected. The test error is increasing almost exponentially while the training error is stabilising. While we would expect the training error to approach 0 the data most likely has too many points and parameters for a hyperplane defined by a 10-degree polynomial to produce a perfect fit. A much larger M is most likely necessary for the training error to approach 0.

Part 2: 1D Polynomial Regression Visualization

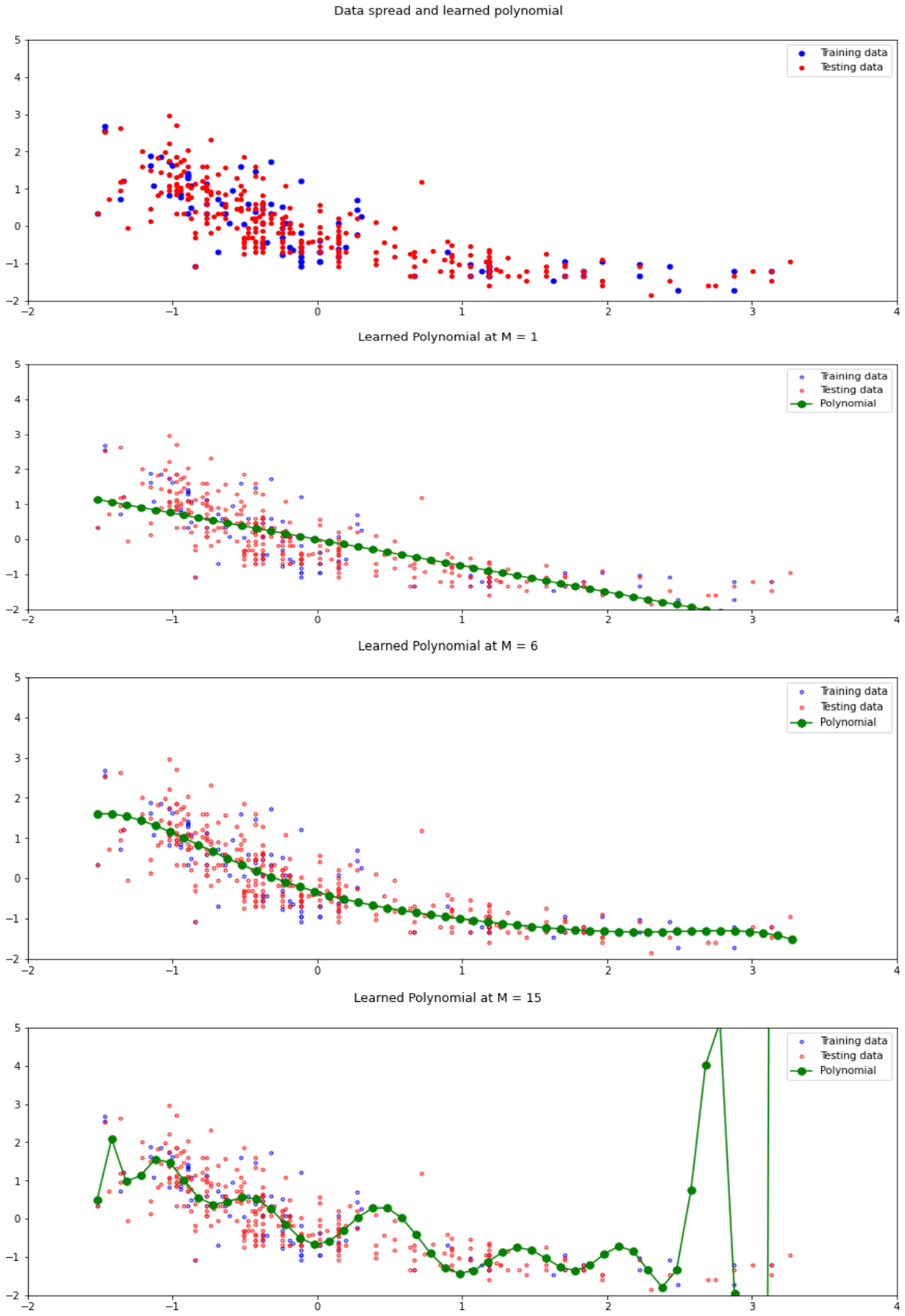

Figure 2: Plots of the learned polynomial at different values of M

M = 1:

It is easy to see that at M=1 the polynomial is underfitted and doesn't follow the general trend of the plotted data points. This is more obvious at values >= 2.5 where the learned polynomial has a large error for all points.

M = 6:

This value was chosen as it was the one which minimized the error the most in Part 1. Visually, we can see that the learned polynomial is well fitted to the general trend of the data and is not heavily influenced by specific values or outliers.

M = 15:

Here we see that the polynomial is overfitted. This is most obvious between the [2.5-3.5] interval on the graph, it is evident that the learned polynomial has overfitted to match the two vertically stacked blue points right before 3. These two points can be easily seen on the data spread graph.

Part 3: Regularized Polynomial Regression

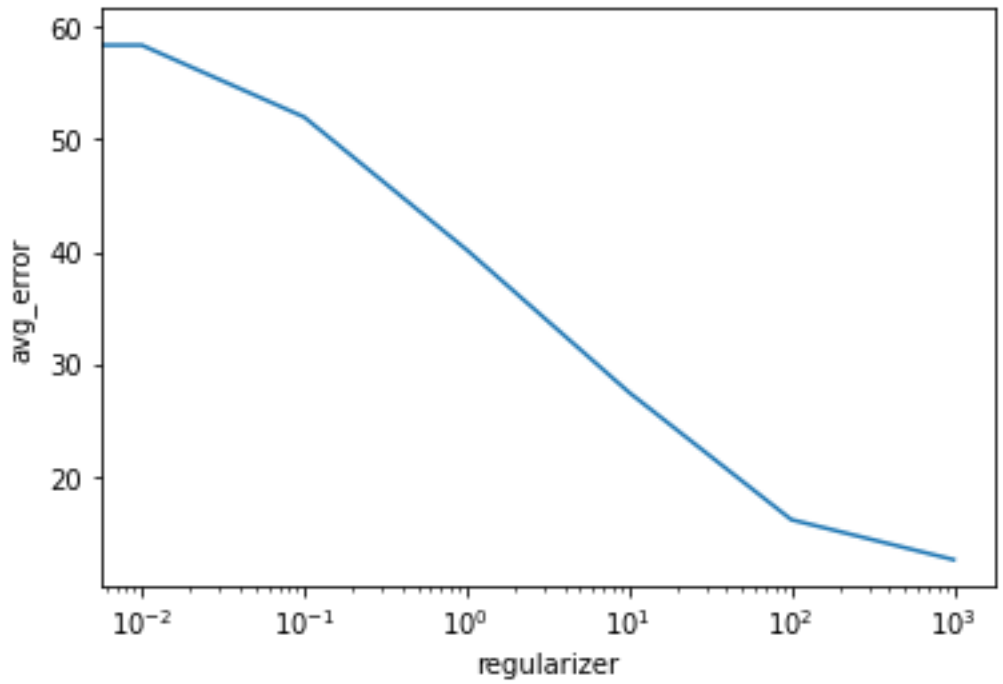

Figure 3: Average error of each regularizer value

This graph was generated using 10-fold cross validation on the first 100 values in the training set. This means that the 100 value set was split into 10 sets, and for 10 iterations one of the sets was chosen as a validation set and the rest were used to train the model. The cross-validation set was fed a regularizer value and outputted the average error, this was run for all listed values of lambda.

The average error of each regularizer value experiences a clear downward trend as the value of lambda is increased. The value of choice is 1000 (as seen in **Figure 5**). This can be seen clearly in **Figure 3** which shows us that the average error is lowest at lambda = 1000. To explain why this is the case we should consider the information from part 2. We know that there are a few key outliers in the data that could result in drastic overfitting as seen in M=15 of part 2, therefore, since larger values of lambda can be used to 'punish' outliers we can keep the line more generalized by increasing the value of M.

This 'punishment' of outliers means that larger and larger values of lambda would result in a straighter line until eventually the line produced is completely linear. The effects of this can be seen in **Figure 4**.

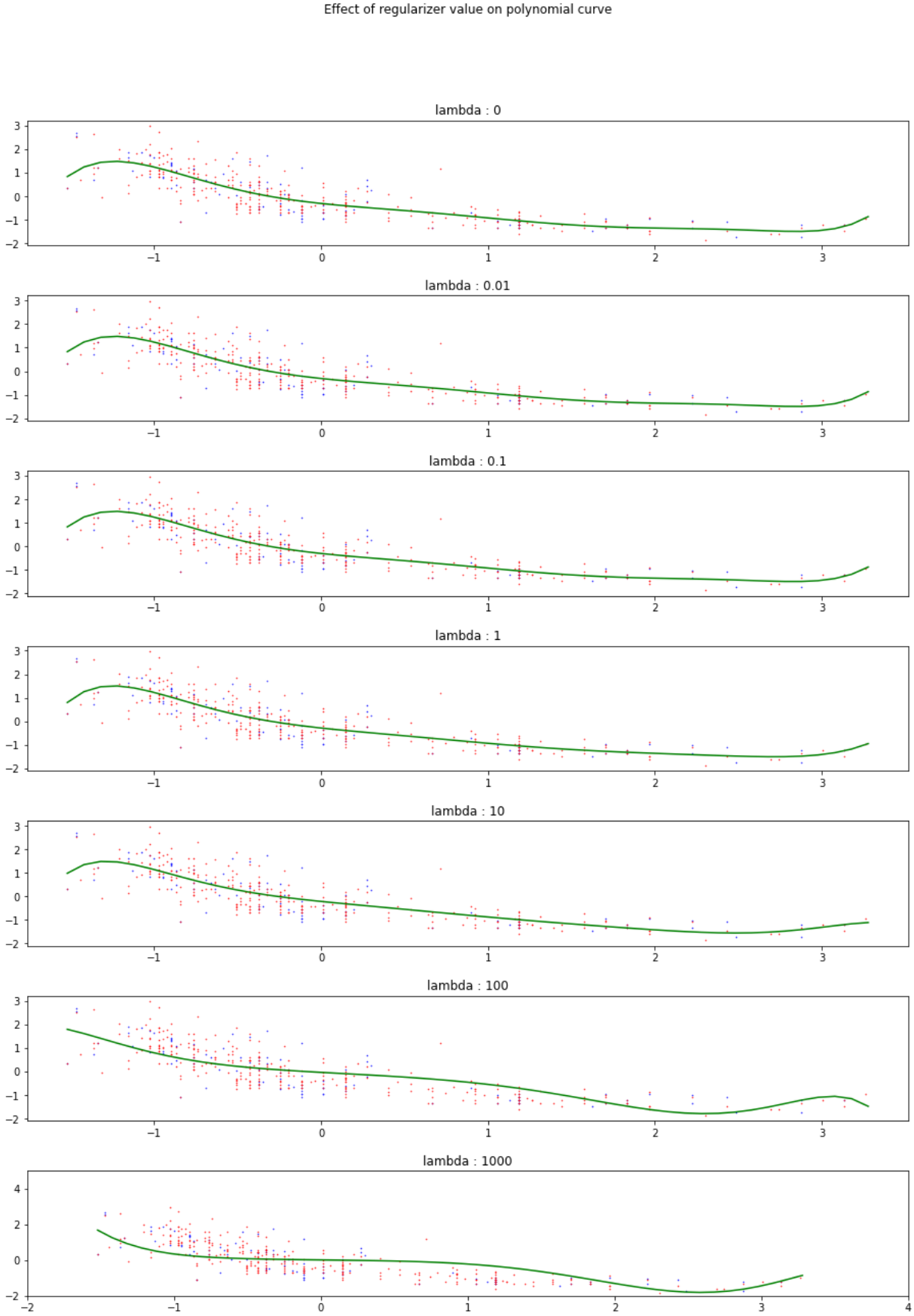

Figure 4: The effects of changing the regularize value.

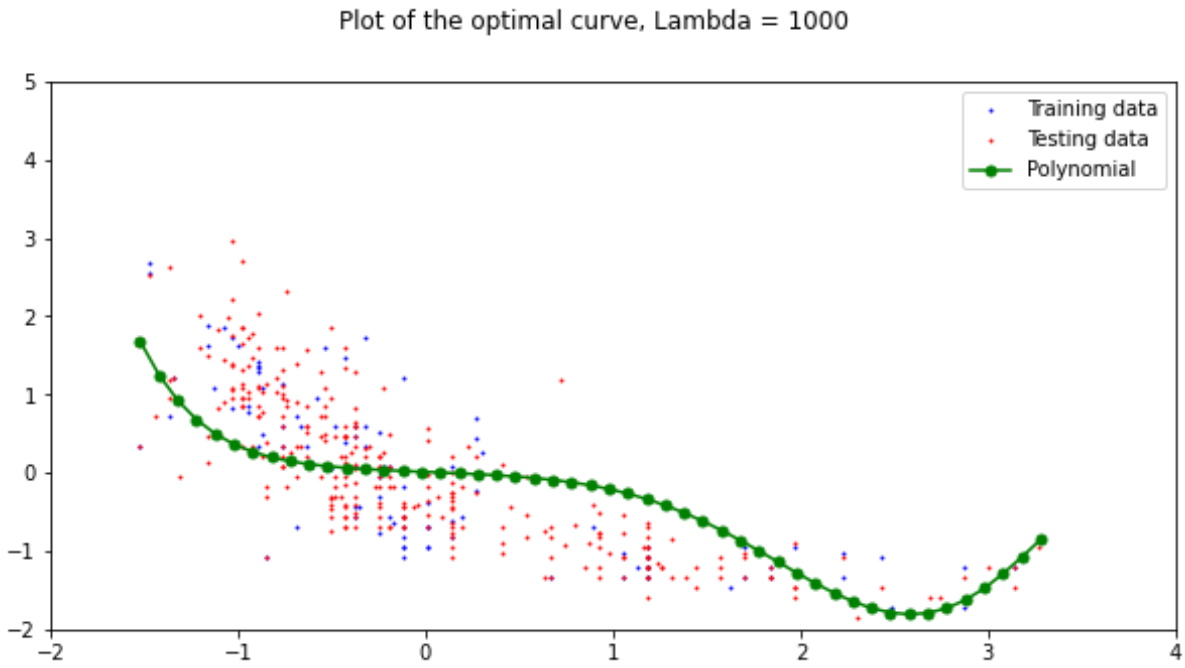

Figure 5: The curve at M=8, Lambda=1000 which produced the minimum error