Assignment 2 - Machine Learning
Theoretical Questions

Mohammad Hossein Basouli

May 3, 2025

Question 1

Because it tries to fit a model from only an specific set of hypothesis functions (which is itself a subset of all polynomial functions) called linear functions. More specifically, it only finds the coefficients associated with linear terms of our features.
We could generalize linear regression by including non-linear terms as our features (e.g. Adding x^2, $sin(x)$ or even interaction between different features of our data like x_1x_2 etc.) to capture non-linear patterns in our data.

Question 2

When we have multicolinearity it gets hard for the model to identify the exact influence for each of the correlated variables, thus this would affect the interpretability and accuracy of our model. We could identify multicolinearity by **correlation tests** among different pairs of variables or calculating **Variance Inflation Factor (VIF)** for each of the variables. Also we can resolve this issue by many different approaches such as:

- Subset Selection
  - **Foward Stepwise Selection**
  - **Backward Stepwise Selection**
  - **Best Subset Selection**
- Shrinkage Methods
  1. **Ridge**

2. **Lasso**
3. **Elastic Net**

- Dimention Reduction Methods

  1. **Principal Component Analysis (PCA)**

Question 3

- **Equal Treatment of the Features**: When we input features with different scales, our model automatically gives more importances and gets influenced more by the features that have a larger scale although that feature might not be important than the others.
- **Making the Importance of the Features Comparable**: We cannot compare the importance of the features at the end of training our model, because high coefficients don't necessarily mean high importance, thus we must normalize the features in order to be able to compare them.
- **Accurate and Stable Convergence**: e.g. Gradient decent gives high importance to the features with large numbers, making them change more dramatically than the other features.

Question 4

Outliers can dramatically increase Mean Squared Error (MSE) even when we have approximately found the optimal solution to our problem. To overcome this issue, we could use **Huber Loss** which essentially uses Mean Abosolute Error (MAE) for errors larger than an specific amount and Mean Squared Error for error which don't exceed that specificed number. Since we don't penalize the error of outliers as much as MSE, this time the model would fit better.

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{if } |a| \le \delta \\ \delta\left(|a| - \frac{1}{2}\delta\right) & \text{if } |a| > \delta \end{cases}$$

Question 5

- The local-minimums should be quite flat. If the local-minimums are very deep, it's hard for **Stocastic Gradient Decent (SGD)** to get out of the local-minimum.

- Number of iterations are high enough.
- Learning Rate is controlled.

Question 6

We might consider a log-scale **transformation** for the following reasons:

- Capturing non-linearity in the data. e.g. Assume that the relationship between the depenedent variable y and x is as $y = \log(x)$; if we apply log-scaling to xs we would end up with the following relationship $y = x'$ where $\log(x) = x'$.
- Reducing heteroscedasticity.
- Reducing impact of outliers.

Also here is a brief explanation about the alternative approaches that we might want to use:

- **Square Root**:
  - Capturing relationships such as $y = \sqrt{x}$.
  - Reduces variance but less aggressive than **log-scale transformation**
- **Inverse**: capturing hyper-bolic relationships
- **Box-Cox**:
  - Automatically selecting the appropriate scaling. e.g. if we use $\lambda = 0$ it would use **log-scale transformation**. or use $\lambda = 0.5$ to use something like **Square Root**.
  - Formula:

$$x^{(\lambda)} = \begin{cases} \dfrac{x^\lambda - 1}{\lambda}, & \text{if } \lambda \neq 0 \\ \log(x), & \text{if } \lambda = 0 \end{cases}$$

Question 7

Both **Ridge & Lasso Regression** are regularization methods that prevent the model from overfitting on the training data. They try to penalize the over usage of the variables and constraint them in order to get only a few non-zero variables that are really important in the problem. **Ridge** adds a quadratic pentalty term to the RSS which helps in faster convergence of the non-important variables towards zero (**Ridge** doesn't force the variables to take the value zero exactly). Whereas **Lasso** only adds the summation of absolute values of the features as a penalty term, which actually forces non-important features to zero, but we a slower convergence speed.

$\text{Loss}_{\text{Ridge}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{p} w_j^2$
$\text{Loss}_{\text{Lasso}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{p} |w_j|$

Question 8

This state usually occurs when the model is underfitted on the data. Since the complexity of the model is pretty low, it suffers from high bias. Also underfitting usually occurs when complexity of the model is not high enough to capture the pattern in the data, thus we must decrease the regularization hyperparameter α in order to turn the variables that have taken zero values back.

Question 9

Locally Weighted Linear Regression (LWR) is just like the **Ordinary Global Linear Regression**, expect it tries to fit the parameters for each single data point separately. More specifically, it tries to have a local prediction for each point separately. It adds a weighting function $w^(i)$ in the **RSS** which tries to give more importance to the points in the training set which are close to the point that we want to predict outcome for. The formula for **weighted RSS** in **LWR** and the weighting function are as follows:

$$J(\theta) = \sum_{i=1}^{m} w^{(i)} \left(y^{(i)} - \theta^T x^{(i)}\right)^2$$

$$w^{(i)} = \exp\left(-\frac{(x^{(i)} - x)^2}{2\tau^2}\right)$$

where τ controls the bandwidth (i.e., how local the weighting is).

- Benefits: LWR works well on non-linear data and doesn't actually need to know anything about the complexity of the data in advance.
- Limitations:
  1. Very slow.
  2. Takes so much memory.
  3. Only works if we have some data around the point that we want to predict.
  4. Very complex.

Question 10

Elastic Net tries to combine the advantages of both of the methods **Ridge & Lasso Regression**, more specifically, it inherits handling of correlated features by distributing the weights among them and shrinking them all out from **Ridge** and setting some features to zero from **Lasso**. The formula is as follows:

$$\text{Cost} = \text{RSS} + \lambda \left(\alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2 \right)$$