Lecture 03 & 04: Regressions and Models

Yorick Chern

Foothill Machine Learning & Data Science Club

1 Linear Regression

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n$$

1. $\hat{y}$ is the predicted value
2. n is the number of features
3. x_i is the i-th feature
4. θ_j is the j-th model parameter (including the bias term θ_0 and the feature weights $\theta_1, \theta_2, ..., \theta_n$)

Vectorized form:

$$\hat{y} = h_\theta(x) = \theta \cdot x$$

where

1. h is the hypothesis function
2. θ is the parameter vector containing all the feature weights
3. x is the instance's feature vector

To measure how well or poorly our model fits, we need a cost function. We will use the Mean Squared Error.

$$J(\theta) = MSE(x, h_\theta) = \frac{1}{m} \sum_{i=1}^{m} (\theta^T \cdot x_i - y_i)^2$$

where

$$\hat{y}_i = \theta^T \cdot x_i$$

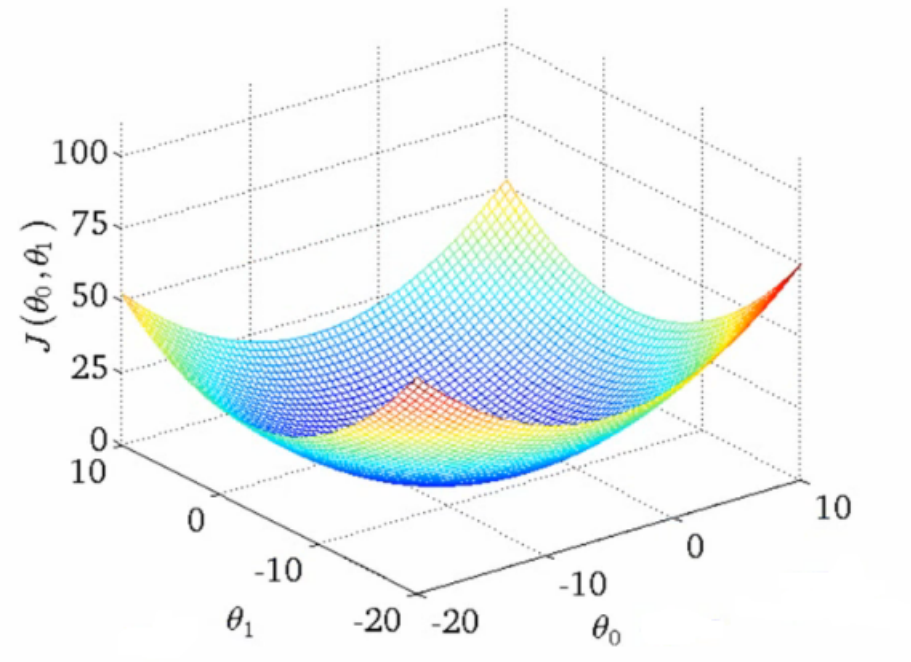

Figure 1: 3D shape of the cost function MSE

2 Gradient Descent

1. Optimization algorithm
2. It measures the local gradient of the error function with regard to the parameter vector θ
3. Goes in the direction of descending gradient
4. When the gradient is zero, we have reached the minimum!
5. Speed of convergence (when the gradient is zero) depends on the learning rate hyperparameter
6. All features should have similar scales
7. The gradient is the partial derivative of the MSE cost function:

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i) x_i$$

2.1 Batch Gradient Descent

$$\theta_{new} = \theta_{old} - \frac{\alpha}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)x_i$$

1. BGD uses the whole batch of training data at every step
2. α is the learning rate of the algorithm
3. May converge at a global minimum
4. Slow on very large training sets

2.2 Stochastic Gradient Descent

$$\theta_{new} = \theta_{old} - \alpha(\hat{y}_{random} - y_{random})x_{random}$$

1. SGD picks ONE random instance in the training set at every step and computes the gradients based only on that instance
2. α - learning schedule: the learning rate that decreases as time goes on; more explanation in the notebook
3. Much faster than BGD
4. SGD can jump out of local minima
5. Will NEVER settle at global minima (will bounce around the minima)

3 Polynomial Regressions

There are many, many ways to perform polynomial regressions. However, it is unnecessary to provide all the background math knowledge (unless you are a CS major at a well funded 4-year university hoping to complete a PhD in machine learning, then yes, you would need to understand it). Instead, I want to focus on how we could utilize the regressors already designed and how we could apply them on different datasets. Check the notebook on GitHub for more details and examples :)