Foundations of Machine Learning

Sewade O. Ogun

November 16, 2019, AIMS Ghana

Chapter 1

Supervised Learning

1.1 Regression Problem

Given a set of datapoints, $D = \{(X_i, y_i)\}_{i=1}^{n}$ where x_i are the features and y_i are the target, corresponding to the features, we can learn a function or hypothesis which maps the features x_i to the target y_i i.e $h : \mathbb{R}^d -> \mathbb{R}^c$

Given a supervised learning task, it is required to understand the given problem at hand to determine;

a. Data

b. Hypothesis or model or Hypothesis class e.g. Regression

c. Criteria (Loss, Cost)

d. Learning Algorithm

1.1.1 Hypothesis

The hypothesis function is

$$h_\theta(X) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n = \sum_{i=0}^{n} \theta_i x_i \qquad (x_0 = 1) \tag{1.1}$$

where θs are parameters of the model and θ_0 is referred to as the **bias/intercept term**. Since this model has parameters, and the goal is to find the best parameters for the data, this form of models are referred to as **parametric models**. The other form are called **non-parametric models**.

1.1.2 Criteria

The criteria for the linear regression model is called **Ordinary Least Squares (OLS) or Mean Squared Error** because of the form of the criteria. This is given as;

$$L(\theta) = \frac{1}{2} \sum_{i=1}^{n} \left(h_\theta(x_i) - y_i\right)^2 \tag{1.2}$$

1.1.3 Learning Algorithm

The goal of the learning algorithm is to find the parameters θ that minimizes the loss function i.e.

$$\min_{\theta} L(\theta) = \frac{1}{2}\sum_{i=1}^{n}\left(h_\theta(x_i) - y_i\right)^2$$

Two approaches (Analytical method or Numerical methods) can be used to solve this problem.

Numerical Approach

Let Θ be the vector of θs and $Theta_0$ the initial solution, we can update Θ by repeatedly subtracting the gradient of the loss function from Θ until convergence i.e.
Repeat

$$\theta_j := \theta_{j-1} - \alpha\frac{\partial}{\partial\theta_{j-1}}L(\theta_{j-1}) \tag{1.3}$$

where α is the **learning rate** and is an hyperparameter to be tuned. This form of learning in (1.3) is known as **gradient descent**. It involves taking steps in the opposite direction of steepest ascent gotten through the gradient.

Computing the Gradient of Loss Function

For one example, $h_\theta(x_i) = \Theta x_i$

$$\begin{aligned}
\frac{\partial}{\partial\theta}L(\theta) &= \frac{\partial}{\partial\theta}\frac{1}{2}\left(h_\theta(x_i) - y_i\right)^2 \\
&= \frac{\partial}{\partial\theta}\frac{1}{2}\left(\theta_i x_i - y_i\right)\frac{\partial}{\partial\theta}\left(\theta x_i - y\right) \\
&= \left(\theta_i x_i - y_i\right)x_i^j \\
&= \left(h_\theta(x_i) - y_i\right)x_i^j
\end{aligned}$$

where x_i^j is the jth feature of the ith example. Therefore the update rule can be modified as; $\theta_j = \theta_j + \alpha(y_i - h_\theta(x^{(i)}))x_i^j$. The equation indicates that for a fixed learning rate, the amount of update is dependent on the difference between predictions and target. This is called the **Widrow-Hoff update rule** or **least mean square rule**.

Batch Gradient Descent

Gradient descent in general assumes that the loss function is differentiable and convex. If the loss function is not convex, the solution might get stuck in a local optimum and not find the global optimum. For linear regression, we can show that the loss function is convex as it is a function of 2nd polynomial and positive.
The batch gradient descent uses all the examples on each iteration to compute the gradient. For a large set of examples, this can be computationally expensive and not used in practice. This also requires that the gradient is computed for every example before making one step of gradient descent.

Repeat until convergence;

$$\Theta = \Theta + \alpha \sum_{i=1}^{n} \left(y_i - h_\theta(x^{(i)}) \right) x_i^j$$

Stochastic Gradient Descent

Stochastic gradient descent selects a sample at random from the examples and uses this to update the gradient. This has the advantage of being fast and requiring less memory but does not guarantee convergence to the optimal solution (even though an approximate solution is usually good enough in practise) after the update. Also, update is not smooth as the direction of descent is dependent on only one data which may be noisy.

Repeat until convergence;
for i : 1 to n;

$$\Theta = \Theta + \alpha \left(y_i - h_\theta(x^{(i)}) \right) x_i^j$$

Mini-batch Stochastic Gradient Descent

A compromise between batch gradient descent and stochastic gradient descent is the mini-batch gradient descent. It uses a mini-batch of examples samples at random from the dataset to update the parameters. This has the advantage of reducing the variance during update and therefor smoother. It also maximixes the usage of hardware and cpu vectorization for efficient computations.

Repeat until convergence;
for k subset in n;

$$\Theta = \Theta + \alpha \sum_{i=1}^{n} \left(y_i - h_\theta(x^{(i)}) \right) x_i^j$$

Analytical Method

The analytical method performs a single computation using the data matrix. It the does not require to be performed iteratively and does not require a learning rate.

$$\begin{aligned} L(\theta) &= \frac{1}{2}\|X\Theta - y\|^2 \\ &= \frac{1}{2}(X\Theta - y)^T(X\Theta - y) \\ &= \frac{1}{2}(\Theta^T X^T - y^T)(X\Theta - y) \\ &= \frac{1}{2}(\Theta^T X^T - y^T)(X\Theta - y) \text{ as } (A-B)^T = A^T - B^T \text{and } (AB)^T = B^T A^T \\ &= \frac{1}{2}(\Theta^T X^T X\Theta - 2y^T X\Theta + y^T y) \end{aligned}$$

The gradient of $L(\Theta)$ is then computed as;

$$\begin{aligned} \nabla_\theta = \frac{1}{2}\left(2X^T X\Theta - 2X^T y\right) \\ X^T X\Theta = X^T y \\ \Theta = \left(X^T X\right)^{-1} X^T y \end{aligned} \tag{1.4}$$

(1.4) is known as the normal equation and requires the inverse of $X^T X$ to be computed. $X^T X$ may not be invertible and can be expensive to compute depending on the size of the dataset. To avoid the invertibility problem, the pseudoinverse can be computed and we can manually remove correlated features (by computing correlation matrix) if the features are linearly dependent. Another approach is to add regularization. To regularize the model, the sum of squares of the parameters is added to the loss function. This has the effect of also reducing the size of values of the parameters if the regularization parameter λ is set appropraitely.

$$L(\theta) = \frac{1}{2}\|X\Theta - y\|^2 + \lambda \|\Theta\|_2^2 \tag{1.5}$$

$$\Theta = \left(X^T X + \lambda I\right)^{-1} y^T X \tag{1.6}$$

The proof of these regularization is shown in a later chapter using the Maximum a posteriori extimation(MAP). I is the identity matrix and $\left(X^T X + \lambda I\right)^{-1}$ always exist