MLT Week-5

Sherry Thomas
21f3001449

Contents

Abstract

The week commences with an exploration of Supervised Learning, specifically focusing on the topic of Regression. The aim is to provide a comprehensive understanding of the underlying mechanism of this popular machine learning technique, and its various applications. Additionally, this study delves into the variants of Regression, including kernel regression, and examines the probabilistic aspects of the technique.

Introduction to Supervised Learning

Supervised learning is a machine learning technique in which the algorithm is trained on labeled data, where the target variable or the outcome is known. The goal of supervised learning is to learn a mapping between the input data and the corresponding output variable.

Given a dataset $\{x_1, \ldots, x_n\}$ where $x_i \in \mathbb{R}^d$, let $\{y_1, \ldots, y_n\}$ be the labels, where y_i could belong to the following:

- Regression: $y_i \in \mathbb{R}$ Example: Rainfall Prediction
- Binary Classification: $y_i \in \{0, 1\}$ Example: Distinguishing between cats and dogs
- Multi-class Classification: $y_i \in \{0, 1, \ldots, K\}$ Example: Digit classification

Linear Regression

Linear regression is a supervised learning algorithm used to predict a continuous output variable based on one or more input features, assuming a linear relationship between the input and output variables. The goal of linear regression is to find the line of best fit that minimizes the sum of squared errors between the predicted and actual output values.

Given a dataset $\{x_1, \dots, x_n\}$ where $x_i \in \mathbb{R}^d$, let $\{y_1, \dots, y_n\}$ be the labels, where $y_i \in \mathbb{R}$.

The goal of linear regression is to find the mapping between input and output variables. i.e.

$$h : \mathbb{R}^d \to \mathbb{R}$$

The error for the above mapping function is given by,

$$\text{error}(h) = \sum_{i=1}^{n} (h(x_i) - y_i)^2)$$

This error can be as small as zero, and this is achieved when $h(x_i) = y_i \ \forall i$. But this may not be the desired output as it may represent nothing more than memorizing the data and its outputs.

The memorization problem can be mitigated by introducing a structure to the mapping. The simplest structure being of the linear kind, we shall use it as the underlying structure for our data.

Let $\mathcal{H}_{\text{linear}}$ represent the solution space for the mapping in the linear space.

$$\mathcal{H}_{\text{linear}} = \left\{ h_w : \mathbb{R}^d \to \mathbb{R} \ \ s.t. \ \ h_w(x) = w^T x \ \ \forall w \in \mathbb{R}^d \right\}$$

Therefore, our goal is,

$$\min_{h \in \mathcal{H}_{\text{linear}}} \sum_{i=1}^{n} (h(x_i) - y_i)^2$$

Equivalently

$$\min_{w \in \mathbb{R}^d} \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

Optimizing the above is the entire point of the linear regression algorithm.

Optimizing the Error Function

The minimization equation can be rewritten in the vectorized form as,

$$\min_{w \in \mathbb{R}^d} ||X^T w - y||_2^2$$

Let this be a function of w and as follows:

$$\begin{aligned} f(w) &= \min_{w \in \mathbb{R}^d} ||X^T w - y||_2^2 \\ f(w) &= (X^T w - y)^T (X^T w - y) \\ \therefore \bigtriangledown f(w) &= 2(XX^T)w - 2(Xy) \end{aligned}$$

Setting the above equation to zero, we get

$$\begin{aligned} (XX^T)w^* &= Xy \\ \therefore w^* &= (XX^T)^+ Xy \end{aligned}$$

where $(XX^T)^+$ is the pseudo-inverse of XX^T.

On further analysis of the above equation, we can say that $X^T w$ is the projection of the labels onto the subspace spanned by the features.

Gradient Descent

The normal equation for linear regression, as seen before, needs the calculation of $(XX^T)^+$ which can computationally be of $O(d^3)$ complexity.

As we know w^* is the solution of an unconstrained optimization problem, we can solve it using gradient descent. It is given by,

$$\begin{aligned} w^{t+1} &= w^t - \eta^t \bigtriangledown f(w^t) \\ \therefore w^{t+1} &= w^t - \eta^t \left[2(XX^T)w^t - 2(Xy)\right] \end{aligned}$$

where η is a scalar used to control the step-size of the descent and t is the current iteration.

Even in the above equation, the calculation of XX^T is needed which is a computational expensive task. Is there a way to adapt this further?

Stochastic Gradient Descent

Stochastic gradient descent (SGD) is an optimization algorithm used in machine learning for finding the parameters that minimize the loss function of a model. In contrast to traditional gradient descent, which updates the model parameters based on the entire dataset, SGD updates the parameters based on a randomly selected subset of the data, known as a batch. This results in faster training times and makes SGD particularly useful for large datasets.

For every step t, rather than updating w using the entire dataset, we use a small randomly selected (k) data points to update w. Therefore, the new gradient is $2(\tilde{X}\tilde{X}^T w^t - \tilde{X}\tilde{y})$ where $\tilde{X}$ and $\tilde{y}$ is the small sample randomly selected from the dataset. This is manageable because $\tilde{X} \in \mathbb{R}^{d \times k}$ which is considerably smaller than X.

After T rounds, we use,

$$w_{SGD}^T = \frac{1}{T} \sum_{i=1}^{T} w^i$$

This has a certain guarantee to have optimal convergence partly because of the randomness involved in it.

Kernel Regression

What if the data points lie in a non-linear sub-space? Just like in the case of clustering non-linear data, we will use kernel functions here too.

Let $w^* = X\alpha^*$ for some $\alpha^* \in \mathbb{R}^n$.

$$\begin{aligned} X\alpha^* &= w^* \\ \therefore X\alpha^* &= (XX^T)^+ Xy \\ (XX^T)X\alpha^* &= (XX^T)(XX^T)^+ Xy \\ (XX^T)X\alpha^* &= Xy \\ X^T(XX^T)X\alpha^* &= X^T Xy \\ (X^T X)^2\alpha^* &= X^T Xy \\ K^2\alpha^* &= Ky \\ \therefore \alpha^* &= K^{-1}y \end{aligned}$$

where $K \in \mathbb{R}^{n \times n}$ and K can be obtained using a kernel function like the Polynomial Kernel or RBF Kernel.

How to predict using alpha and the kernel function? Let $X_{test} \in R^{d \times m}$ be the test dataset. We predict by,

$$w^* \phi(X_{test}) = \sum_{i=1}^{n} \alpha_i^* k(x_i, x_{test_i})$$

where α_i^* gives the importance of the i^{th} datapoint towards w^* and $k(x_i, x_{test_i})$ shows how similar x_{test_i} is to x_i.

Probabilistic View of Linear Regression

Given a dataset $\{x_1, \dots, x_n\}$ where $x_i \in \mathbb{R}^d$, let $\{y_1, \dots, y_n\}$ be the labels, where $y_i \in \mathbb{R}$. The probability of y_i given x_i is given by,

$$P(y|X) \sim w^T X + \epsilon$$

where $w \in R^d$ is an unknown but fixed value and ϵ is the noise corresponding to distribution $\mathcal{N}(0, \sigma^2)$.

Because of the probabilistic nature of this problem, this can be viewed as an estimation problem and the best solution can be approached using Maximum Likelihood. This is given by,

$$\begin{aligned} \mathcal{L}\begin{pmatrix} w; & x_1, x_2, \dots, x_n \\ & y_1, y_2, \dots, y_n \end{pmatrix} &= \prod_{i=1}^{n} \left[\frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(w^T x_i - y_i)^2}{2\sigma^2}} \right] \\ \log \mathcal{L}\begin{pmatrix} w; & x_1, x_2, \dots, x_n \\ & y_1, y_2, \dots, y_n \end{pmatrix} &= \sum_{i=1}^{n} \left[\frac{-(w^T x_i - y_i)^2}{2\sigma^2} + \log\left(\frac{1}{\sqrt{2\pi}\sigma} \right) \right] \end{aligned}$$

Taking gradients w.r.t. w on both sides

$$\hat{w}_{ML} = w^* = (XX^T)^+ Xy$$

Therefore, Maximum Likelihood Estimator assuming Zero mean Gaussian Noise is the same as linear regression with square error.

Credits:

Professor Arun Rajkumar: The content as well as the notations are from his slides and lecture.