NOTES

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Machine Learning: Linear Regression

Author:
W (CID: your college-id number)

Date: January 6, 2020

1 Introduction

Machine Learning can be categorized into two main categories:

1. Supervised Learning: Given input x and output y, the algorithm learns the mapping function $f(x) = y$.

2. Unsupervised Learning: Given only input x, the algorithm learns the underlying structure.

Supervised Learning can be categorized into two main categories:

1. Classification: Predicting a category or a label

2. Regression: Predicting a real value number or a vector

Linear Regression is a machine learning algorithm for regression problems.

2 1D Linear Regression

Given a set of inputs X and a set of output Y. We want to find the line of best fit that goes through all the points $(x_1, y_1), (x_2, y_2), ..., (x_i, y_i)$

We can use a linear line to make a prediction $\hat{y}_i = ax_i + b$. The goal is to minimize the the difference between the prediction $\hat{y}_i$ and the actual output y_i.

To minimize the difference, we want to minimize the **sum of squared errors**

$$E = \sum_{i=0}^{N}(y_i - \hat{y}_i)^2 = \sum_{i=0}^{N}(y_i - (ax_i + b))^2$$

We want to minimize E with respect to a and b, so we need to use partial derivatives

$$\frac{\partial E}{\partial a} = \sum_{i=0}^{N} 2(y_i - (ax_i + b))(-x_i)$$
$$\frac{\partial E}{\partial b} = \sum_{i=0}^{N} 2(y_i - (ax_i + b))(-1)$$

To find the minimum, we want to find when the derivatives are set to 0

$$\frac{\partial E}{\partial a} = \sum_{i=0}^{N} 2(y_i - (ax_i + b))(-x_i) = 0 \Rightarrow a\sum_{i=0}^{N} x_i^2 + b\sum_{i=0}^{N} x_i = \sum_{i=0}^{N} x_i y_i \tag{1}$$
$$\frac{\partial E}{\partial b} = \sum_{i=0}^{N} 2(y_i - (ax_i + b))(-1) = 0 \Rightarrow a\sum_{i=0}^{N} x_i + bN = \sum_{i=0}^{N} y_i \tag{2}$$

Since we know the values of x_i and y_i, we can solve for a and b by solving the simultaneous equations.

2.1 R-squared

R-squared is a metric to test how good a model is.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{Y})^2}$$

When the sum of squared error is 0 R^2 will be 1, indicating a perfect model.

If $R^2 \leq 0$, it means $\sum(y_i - \hat{y}_i)^2 \geq \sum(y_i - \bar{Y})^2$. The model has the same (or worse) performance as predicting the average of Y, which is not very good.

2.2 Proving Moore's Law

Moore's Law is an observation that the number of transistors doubles every two years. Although the data is not linear, we can still apply linear regression.

Let x be the number of years and y be the number of transistors. Moore's Law can be model as

$$\log(y) = ax + b$$

To find how long it takes for the number of transistors to double, we assume there are y_1 transistors in year x_1 and y_2 transistors in year x_2.

$$\log(y_1) = ax_1 + b$$
$$\log(y_2) = ax_2 + b$$

If $y_2 = 2y_1$,

$$\begin{aligned} \log(y_2) &= \log(2y_1) \\ ax_2 + b &= \log(2) + ax_1 + b \\ x_2 &= \frac{\log(2)}{a} + x_1 \end{aligned}$$

Using linear regression to solve for a, we can conclude that the number of transistors double every $\frac{\log(2)}{a}$ years.

3 Multiple Linear Regression

In multiple linear regression, multiple inputs contribute to a single output.

In this case, each input x is a column vector of dimension $D \times 1$. The model for prediction one sample is $\hat{y} = w^T x + b$. The bias term b can be absorbed into w by appending 1 to each vector x so that $\hat{y} = w^T x$

When predicting N samples, we can write the model as

$$\hat{Y} = X_{N \times D} w_{D \times 1}$$

Notice that each row in data matrix X is an input and each column represents a different feature.

Similar to 1D Linear Regression, we want to minimize the **sum of squared errors**

$$E = \sum_{i=0}^{N} (y_i - \hat{y}_i)^2 = \sum_{i=0}^{N} (y_i - w^T x_i)^2$$

We want to minimize E with respect to each component of w, w_j for $j = 1...D$

$$\frac{\partial E}{\partial w_j} = \sum_{i=0}^{N} 2(y_i - w^T x_i)\left(-\frac{\partial w^T x_i}{\partial w_j}\right) = \sum_{i=0}^{N} 2(y_i - w^T x_i)(-x_{ij})$$

To find the minimum, we want to find when the derivatives are set to 0. Since $j = 1...D$, there will be D equations

$$\begin{aligned} \frac{\partial E}{\partial w_j} = \sum_{i=0}^{N} 2(y_i - w^T x_i)(-x_{ij}) = 0 \Rightarrow \sum_{i=0}^{N} w^T x_i x_{ij} &= \sum_{i=0}^{N} y_i x_{ij} \\ w^T \sum_{i=0}^{N} x_i x_{ij} &= \sum_{i=0}^{N} y_i x_{ij} \end{aligned}$$

Notice that $\sum y_i x_{ij}$ is a matrix $M_{1\times D}$ where m_j is the dot product of Y and the j column in data matrix X. Therefore $\sum y_i x_{ij}$ can be represented as $Y^T X$.

Notice that $\sum x_i x_{ij}$ is a matrix $M_{D\times D}$ where m_{ij} is the dot product of column i and column j in data matrix X. Therefore $\sum x_i x_{ij}$ can be represented as $X^T X$.

Hence

$$\begin{aligned} w^T(X^T X) &= Y^T X \\ (X^T X)^T w &= X^T Y && \text{Take Transpose for both sides} \\ (X^T X) w &= X^T Y \\ w &= (X^T X)^{-1} X^T Y \end{aligned}$$

Solving for w gives us the model for multiple linear regression model.

3.1 Interpreting the Weight

Each w_i is the weight of a feature for input x. The interpretation of w_i is the amount y will increase if the i feature of x is increased by 1 and other features remain constant.

4 Polynomial Regression

We can still apply linear regression to polynomial functions. Linear regression means that the weight parameters are linear.

For example given a polynomial $y = ax^2 + bx + c$. We can treat it as a multiple linear regression problem where each input x is a column vector with features $[x^2 \quad x \quad 1]$.

We can then solve for $w = [a \quad b \quad c]$ as described in *Multiple Linear Regression*

5 Gradient Descent

Gradient descent is a optimization method for finding local minimum or minimum in a function.

Instead of solving for w directly, we set w to some random value and take step to update w in the negative of the gradient in each iteration. (α is the learning rate)

$$w \leftarrow w - \alpha \frac{\partial E}{\partial w}$$

In *Multiple Linear Regression* we saw that

$$\frac{\partial}{\partial w_j} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 = \sum_{i=0}^{N} 2(y_i - w^T x_i)(-x_{ij}) = -2\left(\sum_{i=1}^{N} y_i x_{ij} - w^T \sum_{i=1}^{N} x_i x_{ij}\right)$$

$$\begin{aligned}\frac{\partial}{\partial w} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 &= -2(Y^T X - X^T X w) \\ &= -2(X^T Y - X^T \hat{Y}) \\ &= 2X^T(\hat{Y} - Y)\end{aligned}$$

Therefore the weight for linear regression can be updated iteratively by

$$w \leftarrow w - \alpha X^T(\hat{Y} - Y)$$

Notice that the 2 is absorbed into the learning rate.

6 Regularization

In machine learning, regularization is the process of adding information to solve ill-posed problems and prevent over-fitting.

6.1 L1 Regularization

Sometimes we only want to select a few useful feature to predict the trend. L1 Regularization achieves **sparsity**, meaning the final weight will contains mainly zeros with a few non-zeros.

We simply add a penalty term to the original cost. So E becomes

$$E = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \lambda|w|$$

In *Multiple Linear Regression* we saw that

$$\frac{\partial}{\partial w} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 = 2X^T(\hat{Y} - Y)$$

$$\therefore \frac{\partial E}{\partial w} = 2X^T(\hat{Y} - Y) + \lambda sign(w)$$

$$sign(w) = \begin{cases} -1, w < 0 \\ 0, w = 0 \\ 1, w > 0 \end{cases}$$

Since L1 norm is non-differentiable, we have to use gradient descent to solve for w, described in *Gradient Descent*

$$w \leftarrow w - \alpha\left(X^T(\hat{Y} - Y) + \lambda sign(w)\right)$$

6.2 L2 Regularization

Data contains outliers, and this will pull the linear regression line away from the main trend to minimized the sum of squared errors.

One solution is to add the squared magnitude of the weights multiplied by a constant to the error function to punish large weights. So E becomes

$$E = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \lambda|w|^2$$

The squared magnitude $|w|^2$ is $\sum w_i^2$, which is the dot product $w^T w$.

To solve for w, we minimize E with respect to each component of w, w_j for $j = 1..D$ then set the derivatives to 0.

In *Multiple Linear Regression* we saw that

$$\frac{\partial}{\partial w} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 = -2(y^T X - X^T X w)$$

$$\begin{aligned} \therefore \frac{\partial E}{\partial w} &= -2(Y^T X - X^T X w) + 2\lambda w \\ &= -2(Y^T X - X^T X w - \lambda w) = 0 \\ &\Rightarrow (\lambda I + X^T X)w = Y^T X \\ &\Rightarrow w = (\lambda I + X^T X)^{-1} Y^T X \end{aligned}$$

6.3 L1 Regularization vs L2 Regularization

L1 Regularization encourages sparse weights whereas L2 Regularization encourages small weights.

Both methods prevents over-fitting. L1 Regularization accomplishes this by only choosing important features; L2 Regularization accomplishes this by preventing one of the weights to go extremely large.

7 A Deeper Look into Squared Error

In linear regression the aim is to minimize the sum of squared errors

$$E = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

Minimizing the sum of squared errors is equivalent of maximizing the likelihood.

In other words, linear regression is the **maximum likelihood** solution to the line of best fit.

7.1 Maximum Likelihood Estimator (MLE)

Maximum likelihood estimation is a method of estimating the value of parameters of a model. The parameter values are found such that the actual data observed is likely to come from the model.

Suppose we want to find the true mean μ of a set of observed data coming from a Gaussian distribution $X \sim N(\mu, \sigma^2)$.

Since all the data x_i comes from the same distribution, we can calculate the probability of a single data point using the PDF of Gaussian

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Since all the data are independent and identically distributed, the joint probability is the product of all individual probability given distribution X has mean μ.

$$L = p(x_1, x_2, ..., x_N | \mu) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

This is also known as the **likelihood**. We want to find a μ that maximizes this joint probability. Meaning we want to find the best μ so that the data we observed is likely to come from this distribution.

An easier way is to find a μ that maximizes the **log of likelihood**. This is because log is a monotonically increasing function.

$$\begin{aligned}
\underset{\mu}{\operatorname{argmax}} L &= \underset{\mu}{\operatorname{argmax}}(\log L) \\
\underset{\mu}{\operatorname{argmax}}(\log L) &= \underset{\mu}{\operatorname{argmax}}\left(\log \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}\right) \\
&= \underset{\mu}{\operatorname{argmax}}\left(\sum_{i=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}\right)\right) \\
&= \underset{\mu}{\operatorname{argmax}}\left(\sum_{i=1}^{N}\left(\log(1) - \frac{1}{2}\log(2\pi\sigma^2) - \frac{(x_i-\mu)^2}{2\sigma^2}\right)\right) \\
&= \underset{\mu}{\operatorname{argmax}}\left(-\frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2\right) \\
&= \underset{\mu}{\operatorname{argmax}}\left(-\sum_{i=1}^{N}(x_i-\mu)^2\right)
\end{aligned}$$

Maximizing an error function $-E$ is equivalent of minimizing E.

$$\underset{\mu}{\operatorname{argmax}}\left(-\sum_{i=1}^{N}(x_i-\mu)^2\right) = \underset{\mu}{\operatorname{argmin}}\sum_{i=1}^{N}(x_i-\mu)^2$$

Therefore in linear regression, minimizing the sum of squared errors is equivalent to maximizing the log of likelihood.

In other words, linear regression makes the assumption that the data normally distributed, $Y \sim N(Xw, \sigma^2)$

8 A Deeper Look into Regularization

In **Bayes Theorem**, given θ is the parameters we are estimating, and y is the observed data.

$$P(\theta|y) = \frac{P(y|\theta)P(\theta)}{P(y)}$$

$P(\theta|y)$ is the **posterior**. $P(y|\theta)$ is the **likelihood**.

$P(\theta)$ is the **prior**. $P(y)$ is the **evident**.

Classical statistics focuses on maximizing likelihood function. However in a probabilistic point of view, we would want to **maximize a posterior probability (MAP)**.

We can regularize the parameters θ by adding prior knowledge about the expected distribution of θ. This prior is something we explicitly choose that is not based on the data.

In linear regression, the goal is to find the w that maximizes the posterior. We can ignore $P(y)$ because it is constant with respect to maximization. We can take log of the inner function since log is monotonically increasing.

$$\begin{aligned}
\operatorname*{argmax}_w P(w|y) &= \operatorname*{argmax}_w \left(\frac{P(y|w)P(w)}{P(y)}\right) \\
&= \operatorname*{argmax}_w \left(P(y|w)P(w)\right) \\
&= \operatorname*{argmax}_w \left(\log\left(P(y|w)P(w)\right)\right) \\
&= \operatorname*{argmax}_w \left(\log P(y|w) + \log P(w)\right)
\end{aligned}$$

8.1 L1 Regularization

L1 Regularization is equivalent to having a Laplacean Prior of 0 mean. The PDF of the Laplace distribution is

$$Laplace(x) = \frac{1}{2b}e^{-\frac{|x-\mu|}{b}}$$

We know how to find the likelihood $P(y|w)$ from *Maximum Likelihood Estimator*, and the prior $P(w)$ is the joint probability of each weight component from $Laplace(0, b)$

$$\begin{aligned}
\operatorname*{argmax}_w \left(\log P(y|w) + \log P(w)\right) &= \operatorname*{argmax}_w \left(\log \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - w^T x_i)^2}{2\sigma^2}} + \log \prod_{j=1}^{D} \frac{1}{2b} e^{-\frac{|w_j|}{b}}\right) \\
&= \operatorname*{argmax}_w \left(\sum_{i=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - w^T x_i)^2}{2\sigma^2}}\right) + \sum_{j=1}^{D} \log\left(\frac{1}{2b} e^{-\frac{|w_j|}{b}}\right)\right) \\
&= \operatorname*{argmax}_w \left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - w^T x_i)^2 - \frac{1}{b}\sum_{j=1}^{D}|w_j|\right) \\
&= \operatorname*{argmax}_w \left(-\frac{1}{2\sigma^2}\left(\sum_{i=1}^{N}(y_i - w^T x_i)^2 + \frac{2\sigma^2}{b}\sum_{j=1}^{D}|w_j|\right)\right) \\
&= \operatorname*{argmax}_w \left(-\left(\sum_{i=1}^{N}(y_i - w^T x_i)^2 + \lambda\sum_{j=1}^{D}|w_j|\right)\right)
\end{aligned}$$

Maximizing an error function $-E$ is equivalent of minimizing E.

$$\operatorname*{argmax}_w \left(-\left(\sum_{i=1}^{N}(y_i - w^T x_i)^2 + \lambda\sum_{j=1}^{D}|w_j|\right)\right) = \operatorname*{argmin}_w \left(\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \lambda|w|\right)$$

This gives us the formula for L1 Regularization.

8.2 L2 Regularization

L2 Regularization is equivalent to having a Gaussian Prior. We add a prior knowledge that each component in w is normally distributed $N(0,\tau^2)$.

We know how to find the likelihood $P(y|w)$ from *Maximum Likelihood Estimator*, and the prior $P(w)$ is the joint probability of each weight component from $N(0,\tau^2)$

$$\begin{aligned}
\underset{w}{\operatorname{argmax}}(\log P(y|w)+\log P(w)) &= \underset{w}{\operatorname{argmax}}\left(\log\prod_{i=1}^{N}\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(y_i-w^Tx_i)^2}{2\sigma^2}}+\log\prod_{j=1}^{D}\frac{1}{\sqrt{2\pi\tau^2}}e^{-\frac{w_j^2}{2\tau^2}}\right)\\
&= \underset{w}{\operatorname{argmax}}\left(\sum_{i=1}^{N}\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(y_i-w^Tx_i)^2}{2\sigma^2}}\right)+\sum_{j=1}^{D}\log\left(\frac{1}{\sqrt{2\pi\tau^2}}e^{-\frac{w_j^2}{2\tau^2}}\right)\right)\\
&= \underset{w}{\operatorname{argmax}}\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i-w^Tx_i)^2-\frac{1}{2\tau^2}\sum_{j=1}^{D}w_j^2\right)\\
&= \underset{w}{\operatorname{argmax}}\left(-\frac{1}{2\sigma^2}\left(\sum_{i=1}^{N}(y_i-w^Tx_i)^2+\frac{\sigma^2}{\tau^2}\sum_{j=1}^{D}w_j^2\right)\right)\\
&= \underset{w}{\operatorname{argmax}}\left(-\left(\sum_{i=1}^{N}(y_i-w^Tx_i)^2+\lambda\sum_{j=1}^{D}w_j^2\right)\right)
\end{aligned}$$

Maximizing an error function $-E$ is equivalent of minimizing E.

$$\underset{w}{\operatorname{argmax}}\left(-\left(\sum_{i=1}^{N}(y_i-w^Tx_i)^2+\lambda\sum_{j=1}^{D}w_j^2\right)\right)=\underset{w}{\operatorname{argmin}}\left(\sum_{i=1}^{N}(y_i-\hat{y}_i)^2+\lambda|w|^2\right)$$

This gives us the formula for L2 Regularization.

9 Dataset

This link includes some datasets that can be used for linear regression.

`http://college.cengage.com/mathematics/brase/understandable_statistics/7e/students/datasets/mlr/frames/frame.html`