Machine Learning Notes

杨小龙

March 17, 2020\ (work in progress)

Contents

2. About this document

TODO: Write about this document.

3. Conventions

- The function predicting the dependent variable from the input of independent variables is often called hypothesis funtion and is denoted by the name h.
- Weights in linear regression are called θ (lower case ”theta”).
- Inputs or features are called x.
- Outputs or predictions are called y.
- The number of samples or data points is N.
- The number of features is n.
- The variable i is used as index for samples (data points).
- The index for features of data points is written in brackets. For example the third feature of the second data point would be written as follows: $x_2[3]$
- An optimization operation is written as follows:

  $$\min_{\theta}$$

  where min is the function name and θ takes the place of things, which are changed during optimization. For this specific case in natual language we would say *minimize over* θ, which means look for the values of θ for which the argument is minimal.

4. Linear regression

4.1. General advantages

- Linear regression is a rather simple model, whose predictions can be easily understood by a human.
- It is fast to learn this model from data, compared to some other models.
- When a systems output is truly based on a linear combination of its inputs, linear regression is probably the perfect model to make predictions for the system.

4.2. General disadvantages

- Most systems do not have output, which is based on a linear combination of their inputs, so the model is not applicable in many cases.

4.3. Simple linear regression (univariate linear regression)

Simple linear regression is a useful base case for understanding linear regression in general. Instead of working with multiple features, there is only one feature to work with. It tries to make a line fit the data using the following formula:

$$y = \theta_0 + \theta_1 * x_i[0]$$

Where θ_0 is the intercept (it intercepts the y-axis) or bias and θ_1 is the slope of the line, which is fit.

4.4. Multiple linear regression

(a.k.a.: multivariate linear regression)

4.4.1. Characteristics

- Multiple linear regression can theoretically work with an arbitrary number of features.

- In multiple linear regression we model the data with the following formula:

$$y = \theta_0 \cdot x_i\left[0\right] + \theta_1 \cdot x_i\left[1\right] + \theta_2 \cdot x_i\left[2\right] + ... = \theta^T \cdot x_i$$

  where $x_i\left[0\right] = 1$ for all i ...

$$x_i = \begin{pmatrix} x_i\left[0\right] \\ x_i\left[1\right] \\ \vdots \\ x_i\left[n\right] \end{pmatrix} = \begin{pmatrix} 1 \\ x_i\left[1\right] \\ \vdots \\ x_i\left[n\right] \end{pmatrix} \in \mathbb{R}^{n+1}$$

(x_i is one data point for each i, a vector of features, where the first element is 1 to represent the intercept.) ... so that θ_0 is unchanged and represents the intercept. The hypothesis function thus is:

$$h_\theta(x) = \theta^T \cdot x_i$$

4.5. Methods for calculating the error

4.5.1. Ordinary Least Squares (OLS)

(a.k.a.: ”least squared errors regression” or ”least squares”)

Sum the squares (to make it positive) of differences between y_i and what the model predicts for x_i for all data points in the training dataset.

$$Error_\theta = \frac{1}{N} \sum_{i=1}^{N} \left(\left(y_i - \left(\theta^T \cdot x_i \right) \right)^2 \right)$$

or, if we replace the part of the hypothesis function:

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

Inside the sum we calculate the difference between a single y_i given in the training data and the value for y_i as it was predicted using the weights w and $x_i[k]$, where k is the index for the features and x_i is a vector of features. Since the sum is calculated on all training data points and the error would only increase with more data points instead of giving a result independent from the number of data points, we take the mean of the calculated sum instead.

4.6. Learning phase

To get the best fitting model, we will need to create the model, which has the lowest error. That means we are dealing with a minimization (of the error) problem and more generally speaking with an optimization problem.

There are multiple ways of minimizing the error of a linear regression model. For example one could generate random coefficients for the features and calculate the error. If the error is still too high according to some criteria we have, we could generate new random coefficients and calculate the error again. This process we could repeat over and over until we get a satisfactory low error for our coefficients. Another idea is to use some kind of genetic algorithm to get to a sufficiently low error. A common way, which we are going to explore in detail, is to use gradient descend to minimize the error.

4.6.1. Gradient descend for linear regression

The algorithm described herein is sometimes also called ”batch gradient descent”, because it uses all training data points to update coefficients. There are other versions, which do not use all training data points in each iteration, for example *stochastic gradient descent*.

To get to the lowest error we will calculate the slope of the error function (gradient) and walk (change parameters of our model, namely θ) towards where the highest negative slope is, decreasing the error. The slope of a function is its derivative. We are going to calculate the derivative of the error function by partially differentiating the error function.

We have the following error function:

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

which we want to minimize. It does not matter for the optimization result, whether we multiply the whole expression by a constant factor. With regard to changing θ the factor $\frac{1}{N}$ is a constant. We can also multiply again by $\frac{1}{2}$ to simplify the derivation later:

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

which is the same as:

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

This function we will call $J(\theta)$, our *cost function*:

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

This is the function we will minimize:

$$\min_{\theta} \left(J(\theta) \right)$$

The partial derivatives for θ_k are[1]:

$$\frac{\delta}{\delta\theta_k} J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left((h_\theta(x_i) - y_i) \cdot x_i\left[k\right] \right)$$

where x_0 is again 1, in order to make θ_0 the intercept.

1. Convergence of coefficients

   The algorithm to make the coefficients convergence to a local optimum is:

   Repeat until convergence[2] …

   $$\theta_k := \theta_k - \alpha \frac{\delta}{\delta\theta_k} J(\theta)$$

   … for all k.

   There are one more k than the number of features in the data set, because θ_0 is the intercept and not a coefficient for a feature and the intercept needs to be updated too.

   α is called the learning rate. It influences how far into the direction of steepest descent the value of θ_k will step in one iteration. When choosing α, one needs to keep in mind the following things:

   - If α is too high, it might be that the algorithm oversteps the local optimum at every coefficient update. This would cause the algorithm to not be able to converge to the optimum and might even diverge. Divergence is possible, because when the algorithm oversteps the optimum, it might arrive at a point in the error function landscape, where the slope is higher than before, so the learning rate might be multiplied with the higher slope from that new point, which would mean a higher update for the θ_k. This could happen an arbitrary number of times, depending on the shape of the error function landscape, and thus cause divergence.
   - If α is too small, gradient descent will need a long time to converge.
   - α is always a positive number to enable any learning.

   $\frac{\delta}{\delta\theta_k} J(\theta)$ is the derivative of the error function or cost function.

[1] To see how the derivative of $J(\theta)$ is calculated see Appendix: Derivative of the squared error function.

[2] Converging in this case means, that the values for θ do not change more than a predefined amount.

- When the slope (derivative) is negative (values become lower towards higher θ_k), we substract a negative number from the respective θ_k, so we increase θ_k.
- When the slope is positive (values become higher towards higher θ_k), we substract a positive number from θ_k, so we increase θ_k.

This means that no matter what kind of slope we have, gradent descent will move towards the correct direction.

When θ_k is already at a local optimum, no matter what the value of α is, it will not change θ_k, because the slope will be 0 and anything multiplied by 0 will be 0.

Since the learning rate α is multiplied by the slope, α does not necessarily need to change, in order for gradient descent to converge. The absolute value of the slope can descrease towards the local optimum and thereby reduce the update amount for any θ_k [3].

There is another way of getting to the local optimum, which is called *normal equation method*.

2. Implementation

When implementing gradient descent, one needs to be careful to update all coefficients based on the same (previous) state of coefficients. To update one coefficient and then use the updated coefficient when calculating the updated value for another would be a mistake. This means the algorithm needs to hold the list of previous coefficients for calculation of the updated coefficients and the updated coefficients themselves in each iteration of the update algorithm.

This is expressed as follows:

$$\theta_{k_{next}} := \theta_{k_{prev}} - \alpha \frac{\delta}{\delta \theta_{k_{prev}}} J(\theta_{prev})$$

This is called updating the θ_k ”simultaneously”. If we write the actual derivative in this formula, we get:

$$\theta_{k_{next}} := \theta_{k_{prev}} - \alpha \frac{1}{N} \sum_{i=1}^{N} ((h_\theta(x_i) - y_i) \cdot x_i)$$

[3]If the error function landscape was bowl shaped for example, the slope would get smaller the closer the error moves towards the local (and in this case global) optimum. In the case of linear regression the landscape is *always* bowl shaped. This is called a ”convex function”. It only has one local optimum, which is the global optimum.

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \left(\left(y_i - \left(\theta^T \cdot x_i \right) \right)^2 \right)$$

$$\min_{\theta} \frac{1}{2N} \sum_{i=1}^{N} \left(\left(y_i - \left(\theta^T \cdot x_i \right) \right)^2 \right)$$

For simple linear regression the derivations look as follows:

$$\frac{\delta}{\delta m} = \frac{2}{N} \sum_{i=1}^{N} \left(-x_i \left(y_i - \left(m x_i + b \right) \right) \right)$$

$$\frac{\delta}{\delta b} = \frac{2}{N} \sum_{i=1}^{N} \left(- \left(y_i - \left(m x_i + b \right) \right) \right)$$

Here is how we get to this result:

4.6.2. Disadvantages

- since the mean is susceptible to outliers, the error metric is susceptible as well

4.6.3. Advantages

- relatively simple to understand

4.6.4. Least median or squares

This method works the same as OLS, except that it takes the median instead of the mean as error.

4.6.5. Least absolute errors

$$Error_{\theta} = \frac{1}{N} \sum_{i=1}^{N} \left| y_i - \left(w^T \cdot x_i \right) \right|$$

4.6.6. Other methods

Todo

5. Polynomial Regression

6. Appendix

6.1. Derivative of the squared error function

TODO