ML Note1

Hai Wang

10/6/2020

#Note 1

# 0. What is Machine Learning?

Tom Mitchell definition: “A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P**, if its performance at tasks in **T**, as measured by **P**, improves with experience **E**.”

Example: playing checkers.

E = the experience of playing many games of checkers

T = the task of playing checkers.

P = the probability that the program will win the next game.

In general, any machine learning problem can be assigned to one of two broad classifications:

**Supervised learning** and **Unsupervised learning**.

## 0.1 Supervised Learning

In supervised learning, we are given a data set and already know what our correct output should look like, having the idea that there is a relationship between the input and the output. Supervised learning problems are categorized into “regression” and “classification” problems.

* regression: we are trying to predict results within a continuous output
* classification: we are trying to predict results in a discrete output.

## 0.2 Unsupervised Learning

Unsupervised learning allows us to approach problems with little or no idea what our results should look like.

We can derive this structure by clustering the data based on relationships among the variables in the data.

# 1. Linear Regression with One Variable

Linear regression predicts a real-valued output based on an input value.

## 1.1 Model Representation

= “input” variables

= “output” variables we are trying to predict

= a training example

= a training set

= the space of input value

= the space of output values

Supervised learning problem: given a training set, to learn a function h : X → Y so that h(x) is a “good” predictor for the corresponding value of y.



## 1.2 Cost Function

To measure the accuracy of our hypothesis function we use a **cost function**. This takes an average difference of all the results of the hypothesis with inputs from x’s and the actual output y’s.

$J(\theta\_0,\theta\_1)= \frac{1}{2m}\sum\_{i=1}^{m}(\hat{y}\_i -y\_i)^2\\=\frac{1}{2m}\sum\_{i=1}^{m}(h\_\theta(x\_i)-y\_i)^2$

This functuion is half the mean of the squares of , or the difference between the predicted value and the actual value. <- **We want to minimize this!**

This function is otherwise called the “Squared error function”, or “Mean squared error”. The mean is halved as a convenience for the computation of the gradient descent, as the derivative term of the square function will cancel out the term.



### 1.2.1 Cost Function - Intuition I

Say our training data set is scattered on the x-y plane, we want to fit a staight line (defined by )) that passes all the points, so the error is going to be zero.

Cost function or



so when , the cost function is 0, but what if ? The line no longer perfectly fit all points.



If we try a bunch of values for , we can get the following graph, which shows that, when , we are at our global minimum.



### 1.2.1 Cost Function - Intuition II

A contour plot is a graph that contains many contour lines. A contour line of a two variable function has a constant value at all points of the same line.

To minimize the cost function, the below graph with and gives us the center of the inner most circle, which is the global minimum.



## 1.3 Parameter Learning

### 1.3.1 Gradient Descent

So we have our hypothesis function (h(x)) and we have a way of measuring how well it fits into the data (J). Now we need to estimate the parameters in the hypothesis function.

We put on the x axis and n the y axis, with the cost function on the vertical z axis. The points on our graph will be the result of the cost function using our hypothesis with those specific theta parameters. The graph below depicts such a setup.



when its value is the minimum. The red arrows show the minimum points in the graph.

The way we do this is by taking the derivative (the tangential line to a function) of our cost function. The slope of the tangent is the derivative at that point and it will give us a direction to move towards. We make steps down the cost function in the direction with the steepest descent. **The size of each step is determined by the parameter , which is called the learning rate**.

The gradient descent algorithm is:

repeat until convergence:

Where

j=0,1 represents the feature index number.

At each iteration j, one should simultaneously update the parameters .

### 1.3.2 Gradient Descent Intuition

Say we only use one parameter , and plot its cost function to implement a gradient descent.

The gradient descent algorithm is:

repeat until convergence:

’s sign doest not matter. when the slope is negative, the value of increases and when it is positive, the value of decreases.



Learning Rate :

if too small, GD can be slow;

if too large, GD can overshoot the minimum, and fail to converge, or even diverge.





### 1.3.3 Gradient Descent For Linear Regression

gradient descent equation for linear regression:

repeat until convergence: {

}

where m is the size of the training set, a constant that will be changing simultaneously with , and are values of the given training set.

The point of all this is that if we start with a guess for our hypothesis and then repeatedly apply these gradient descent equations, our hypothesis will become more and more accurate. This method looks at every example in the entire training set on every step, and is called **batch gradient descent**.

Since the optimization problem we have posed here for linear regression has only one global, and no other local, optima; thus gradient descent always converges (assuming the learning rate α is not too large) to the global minimum.