CVT: Lecture 7

Novitoll

2018-03-09

1 Machine learning

- Supervised learning
- Unsupervised learning
- Reinforcement learning

2 Linear regression

Simple linear function $y = mx+b$ in Supervised Learning has the hypothesis $h_\theta(x) = \theta_0 + \theta_1 x$ to predict with given X N-dimensional matrix of features, θ is coefficient (something to be found during model training)

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + .. + \theta_i x_i = \sum_{i=0}^{m} \theta_i x_i = \theta^T x$$

, where m - is number of features.

Now we want to minimize the difference between hypothesis $h_\theta(x)$ and ground truth y for high accuracy, the difference is called Cost function - $J(\theta)$.

$$J(\theta) = \frac{1}{2} \sum_{i=0}^{m} (h_\theta(x^i) - y^i)^2$$

This is the generalized cost function, in practice you will find its implementations as:

- mean squared error (MSE) = $\frac{1}{2m} \sum_{i=0}^{m} ..$
- hinge loss

- log loss
- ..

In order to $\min_\theta J(\theta)$ let's differentiate it (keeping in mind that X - is a matrix of independent variables):

$$\frac{\partial}{\partial\theta_i}J(\theta) = \frac{\partial}{\partial\theta_i}\ \frac{1}{2}(h_\theta(x)-y)^2 =$$
$$2\ \frac{1}{2}\frac{\partial}{\partial\theta_i}\ (h_\theta(x)-y)\ \frac{\partial}{\partial\theta_i}\ (h_\theta(x)-y) =$$
$$(h_\theta(x)-y)\ \frac{\partial}{\partial\theta_i}(\theta_0+\theta_1x_1+..+\theta_n-y) =$$
$$(h_\theta(x)-y)\ \frac{\partial}{\partial\theta_i}(h_\theta(x)-y) =$$
$$(h_\theta(x)-y)\ x_i$$

Update of our cost function (Batch gradient descent) is:

$$\theta_i := \theta_i - \alpha\sum_{i=0}^{m}(h_\theta(x^i)-y^i)\ x^i \tag{1}$$

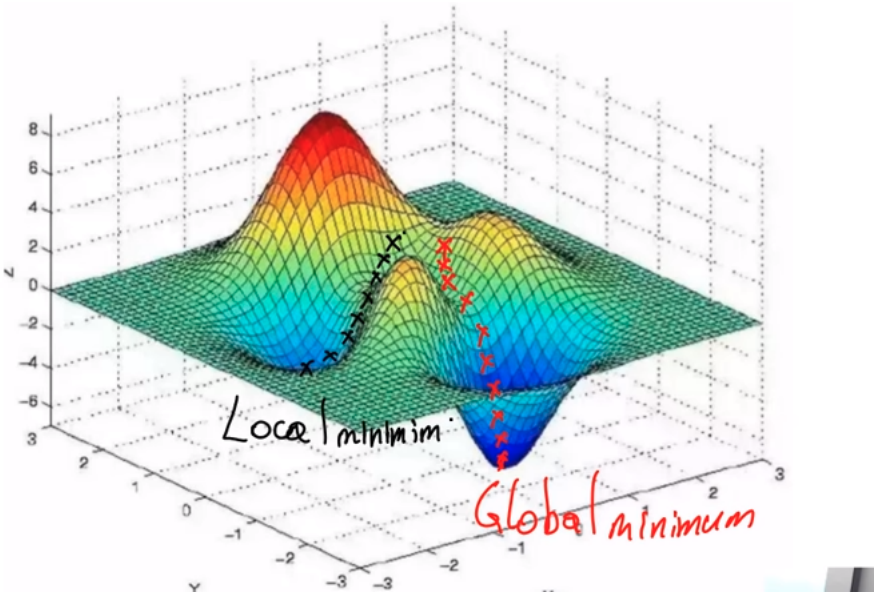
, where α - is learning rate. BGD (1) is good for small dataset and always finds the local minima, however with the real world dataset it's slow. For that, usually Stochastic gradient descent (SGD) is used:

```
Repeat 1..10 {
    for j=1 .. to m {
```

$$\theta_i := \theta_i - \alpha(h_\theta(x^j)-y^j)\ x_i^j \tag{2}$$

```
    }
}
```

SGD works well with big dataset, however wont converge to the 0 (minima), but though will be close as much as possible. The idea is to find global minima in order not to **overfit** with finding local minima.


Local minimum
Global minimum