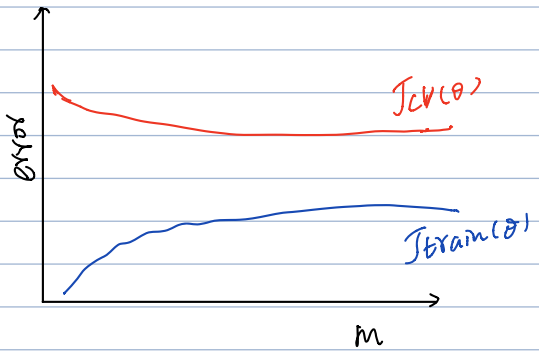
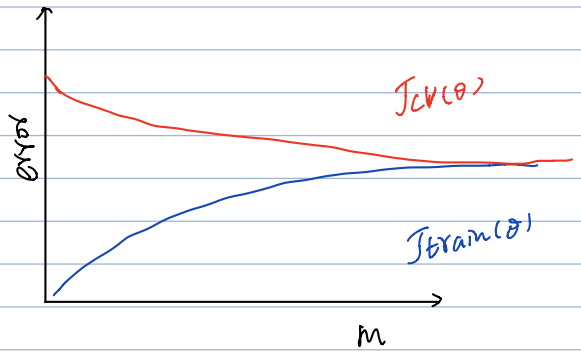
learning with large datasets.

In order to verify if the large datasets is likely to perform better than small datasets, we need to plot a learning curve for a range of value of m and verify that the algorithm has high variance problem when m is small



high variance → increasing training examples may perform better



high bias → increasing features or NN layers may work better.

Stochastic Gradient Descent

$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J_{train}(\theta) = \frac{1}{m}\sum_{i=1}^{m} cost(\theta, (x^{(i)}, y^{(i)}))$$

1. Randomly shuffle dataset. → 打亂順序.

2. Repeat {
   for i = 1 : m {
   $\theta := \theta - \alpha(h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$ → every iteration only lets the algorithm fit i^{th} traing example better.
   for j = 0 : n)
   }
   }.

$\frac{\partial}{\partial \theta_j} cost(\theta, (x^{(i)}, y^{(i)}))$

The main problem of SGD is it doesn't always point to the global minimum, and it only wonders around the global minimum in some region, it doesn't get to the global minimum and stay there.

Mini-batch Gradient Descent.

Batch GD : use all m examples in each iteration
Stochastic GD : use 1 example in each iteration.
Mini-batch GD : use b example in each iteration.
b = batch-size.

Repeat {
for i = 1 : m {
$\theta_j := \theta_j - \alpha \cdot \frac{1}{b}\sum_{k=i}^{i-b+1}(h_\theta(x^{(k)}) - y^{(k)})x_j^{(k)}$

SGD convergence.

Check for convergence:

Batch GD:

plot $J_{train}(\theta)$ as a function of the number of iterations of gradients Descent.

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left|h_\theta(x^{(i)}) - y^{(i)}\right|^2$$

SGD:

$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$

During learning, compute $cost(\theta, (x^{(i)}, y^{(i)}))$ before updating θ using $(x^{(i)}, y^{(i)})$.

Every 1000 iterations, compute $cost(\theta, (x^{(i)}, y^{(i)}))$ averaged over the last 1000 examples.

The learning rate can be slowly decreased over time:

$$\alpha = \frac{const\ 1}{iterations + const\ 2}$$

Map-reduce and Data Parallelism.

map reduce

Batch GD: $\theta_j := \theta_j - \alpha\frac{1}{400}\sum_{i=1}^{400}(h_\theta(x^{(i)}) - y^{(i)})\cdot x_j^{(i)}$