Learning with large datasets

Sheer amount of data → Learning algorithm works better

Large datasets → low-bias learning algorithm

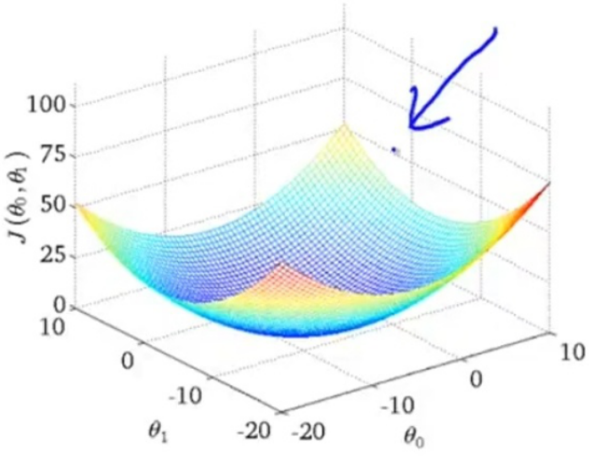
own unique problem - Computational prob.

Stochastic Gradient Descent

Linear regression with gradient descent

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j$$

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



Repeat {

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

(for every $j = 0, \dots, n$)

}

if m is large, Computing can be very expensive.

Batch gradient descent

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

Repeat {

$$\theta_j := \theta_j - \alpha \underbrace{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}}_{\frac{\partial}{\partial\theta_j} J_{train}(\theta)}$$

(for every $j = 0, \ldots, n$)

}

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
2. Repeat {

for $i = 1, \ldots, m$ {

$\theta_j := \theta_j - \alpha \underbrace{(h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}}_{\frac{\partial}{\partial\theta_j} cost(\theta, (x^{(i)}, y^{(i)}))}$

(for $j = 0, \ldots, n$)

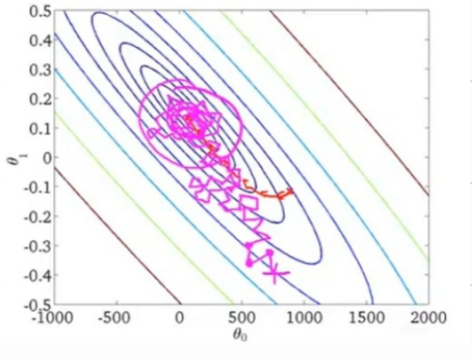
}

}

Batch gradient descent will tend to take a reasonably straight line trajectory to get to the global minimum

↕

Stochastic gradient descent: every iteration is just trying to fit single training ex. better



Mini-batch gradient descent

Batch GD : Use all m examples in each iteration

SGD : Use 1 example in each iteration

Mini-batch GD: Use b examples in each iteration

b = mini-batch size

Say $b = 10, m = 1000.$

Repeat {

→ for $i = 1, 11, 21, 31, \ldots, 991$ {

→ $\theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{i+9} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)}$

(for every $j = 0, \ldots, n$)

}

}

첫 10 examples로 cost를 α만큼 고침,

다음 10 examples로 고침 … m개 샘플까지 반복

good vectorized implementation : better than SGD

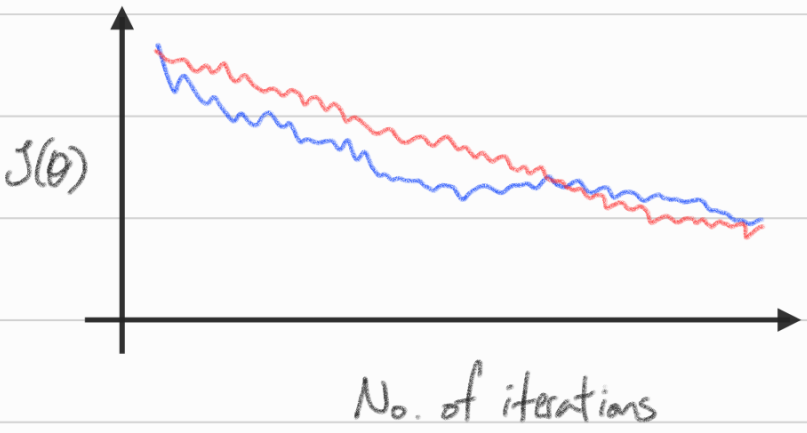
Stochastic Gradient Descent Convergence

Stochastic gradient descent:

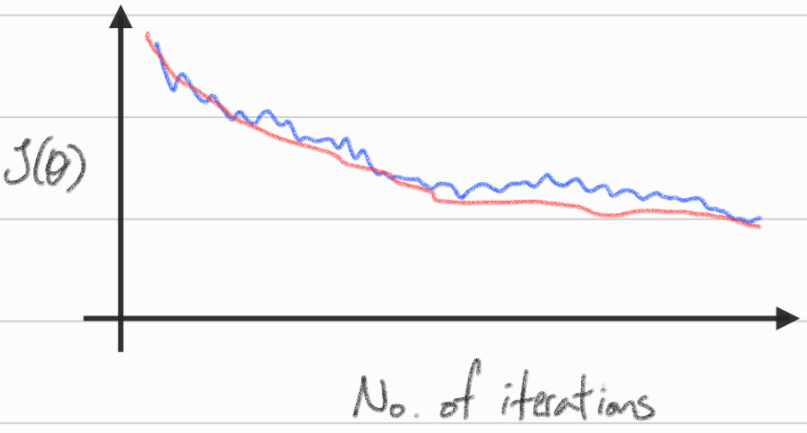
→ $cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$ ⇒ $(x^{(i)}, y^{(i)})$, $(x^{(i+1)}, y^{(i+1)})$, …

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

→ Every 1000 iterations (say), plot $cost(\theta, (x^{(i)}, y^{(i)}))$ averaged over the last 1000 examples processed by algorithm.



$\alpha > \alpha$



averaged over 1000 ex

averaged over 5000 ex (smoother)

↳ get one data point on plot every 5000 ex.