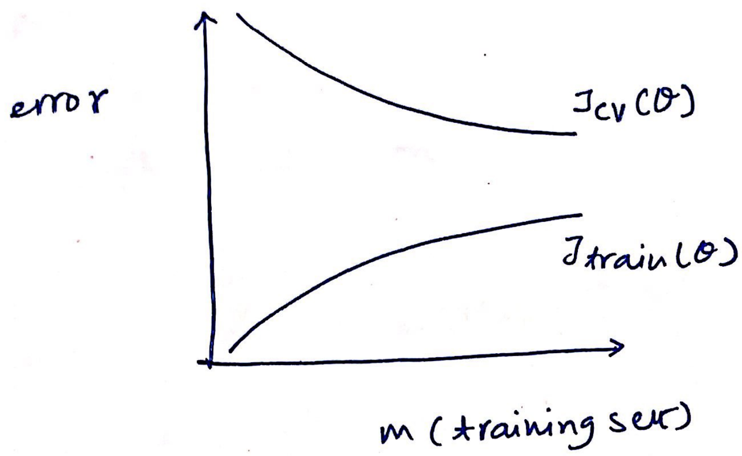
WEEK- 10

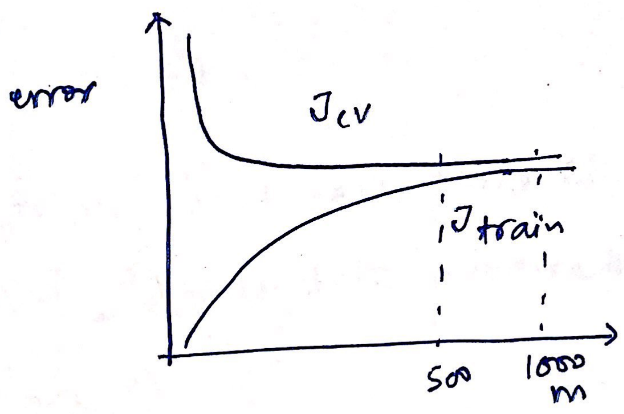
LARGE SCALE MACHINE LEARNING

$m = 100{,}000{,}000$

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



add more examples.



* add more examples is unlikely to help
* add extra features / add hidden units.

Stochastic Gradient Descent :

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

1. Randomly shuffle dataset

2. Repeat
{

for $i = 1 \dots m$

{

$$\theta_j := \theta_j - \alpha.\left(h_\theta(x^i) - y^i\right).x_j^i$$

for $j = 0 \dots n$

}

}

$\Downarrow$

→ fitting (x^1, y^1) → update parameters → fit

→ fitting (x^2, y^2) → update parameters → fit

→ ... (x^3, y^3)

⋮

→ (x^m, y^m)

Batch Gradient descent.

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

Repeat :
{

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

Mini-Batch gradient descent

→ Batch gradient descent: use all m examples in each iteration

→ Stochastic gradient descent: use 1 example in each iteration

→ Mini-Batch gradient descent: Use b examples in each iteration

b = mini-Batch size

$b \in [2, 100]$

~~# Get b examples: $(x^1, y^1) \ldots (x^{i+9}, y^{i+9})$~~

$\cancel{\theta_j := \theta_j - \frac{\alpha}{10} \sum}$

Say $b = 10$, $m = 1000$

Repeat {

for $i = 1, 11, \ldots 991$

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

for every $j = 0 \ldots n$

}

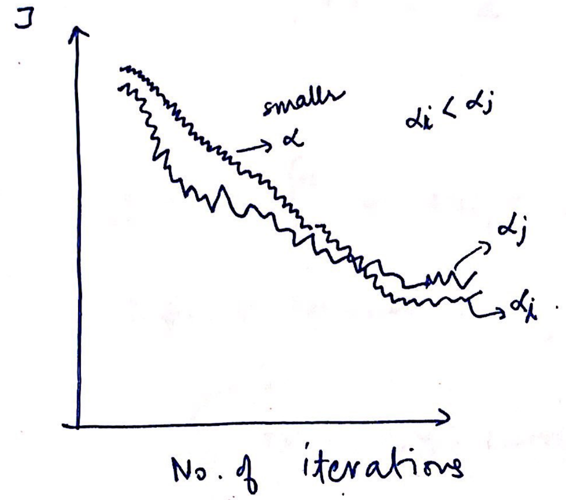
}

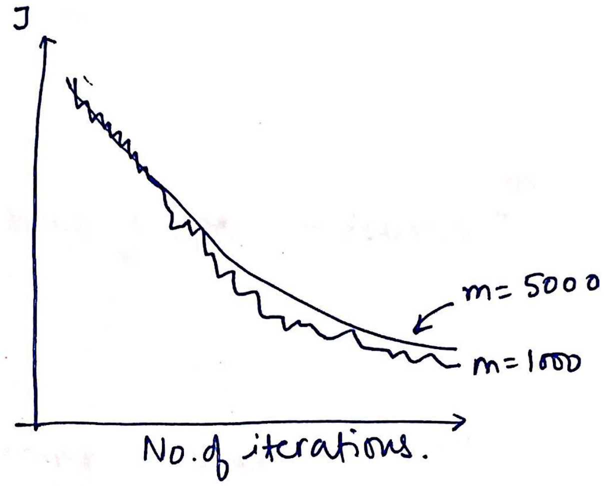
Stochastic Gradient descent Convergence

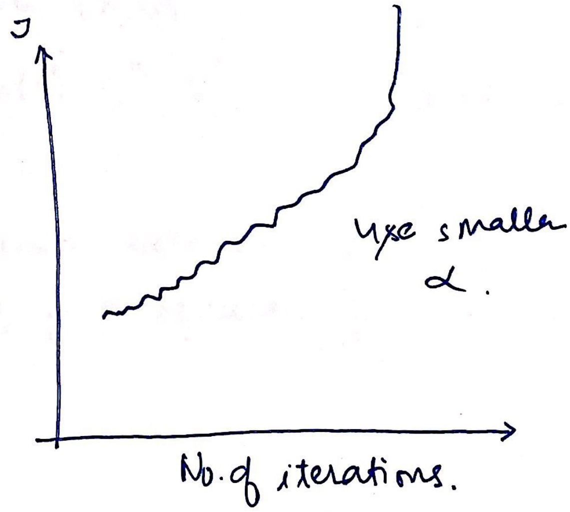
$$cost(\theta, x^i, y^i) = \frac{1}{2}(h_\theta(x^i) - y^i)^2$$

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

↳ Every 1000 iterations (say), plot $cost(\theta, x^i, y^i)$ averaged over the last 1000 examples processed.







* learning rate α, is typically held constant. Can slowly decrease α over time if we want θ to converge.

eg. $\left[\alpha = \dfrac{\cancel{\text{const}}\ \text{const.}1}{\text{iteration Number} + \text{const}2.} \right]$ $\qquad \underline{\underline{\alpha \to 0}}$

ONLINE LEARNING

→ shipping service website where user comes, specifies origin and destination, you offer to ship their package for some asking price, & users sometimes choose to use your shipping service $(y=1)$, sometimes not $(y=0)$

→ Features x capture properties of user, of origin/destination and asking price. We want to learn $p(y=1 \mid x; \theta)$ to optimize price.

→ Algorithm : {* Can adapt to changing user preference }*

Repeat forever

{

Get (x, y) corresponding to user

Update (θ) using only (x, y) ***

$$\theta_j := \theta_j - \alpha \left(h_\theta(x^i) - y^i \right) . x_j^i \qquad j \in \{0, \dots n\}$$

}

[** we don't train on a particular data set rather as users come in θ_j gets updated]

Example: Product Search | Predicted CTR.

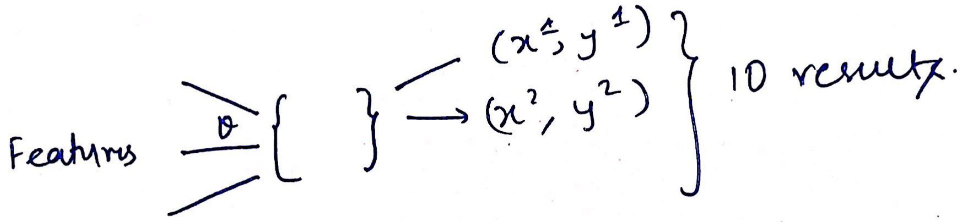
→ user searches for "Android phone 1080p camera"

→ Have 100 phones in store. will return 10 results.

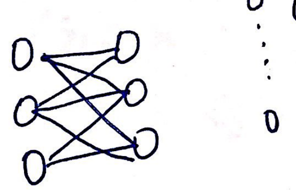
x = features of phone ‖ how many words in user query match name of the phone ‖ match description of the phone. etc

* $y=1$ if user clicks on link. $y=0$ otherwise

$$\boxed{\text{Learn } p(y=1 \mid x; \theta)}$$



or



100 units but only 10 of them will 10

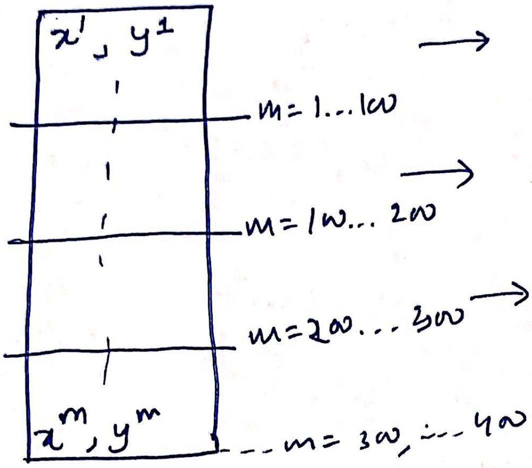
{ other examples: choosing special offers to show user customised selection of news articles. }

Map reduce :→

m $\approx$ large

Batch gradient descent: $\theta_j := \theta_j - \alpha \underbrace{\frac{1}{m} \cdot \sum_{i=1}^{m} (h_\theta(x^i) - y^i) x_j^i}_{\frac{\partial J}{\partial \theta_j}}$

Machine 1: Use $(x^1, y^1) \ldots (x^{100}, y^{100})$



$$temp_j^1 = \sum_{i=1}^{100} (h_\theta(x^i) - y^i) x_j^i$$

$$temp_j^2 = \sum_{i=100}^{200} (h_\theta(x^i) - y^i) x_j^i$$

$\vdots$

$temp_j^1$

$temp_j^2$

$\vdots$

$temp_j^n$

n = no. of machines.

Combine:

$$\theta_j = \theta_j - \alpha \frac{1}{400} (temp_j^1 + \ldots + temp_j^n)$$