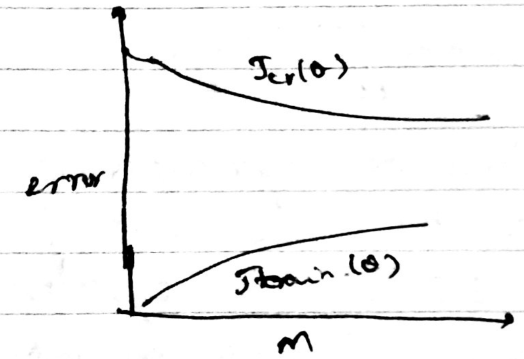
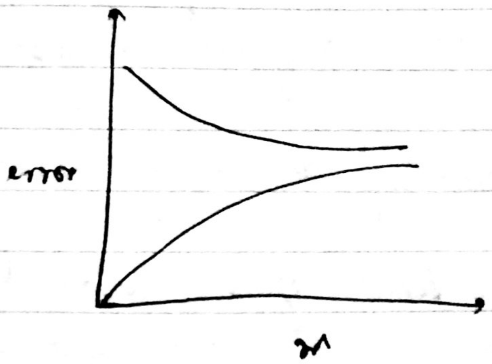
ML with large datasets.

for checking if we should use 100 million examples or not. We plots a learning curve



(High variance)
Adding examples improve efficiency



(High bias)
Adding examples wont do anything much.

Stochastic Gradient Descent

for linear regression with gradient descent,

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

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

steps computationally expensive,

Repeat {

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

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 datasets.
2) Repeat {

for $i = 1, \dots m$

{

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

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

}

}

Mini batch gradient descent

We use b examples in each iteration.

b = mini-batch size

if $b = 10$

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

$i = 1 + 10$

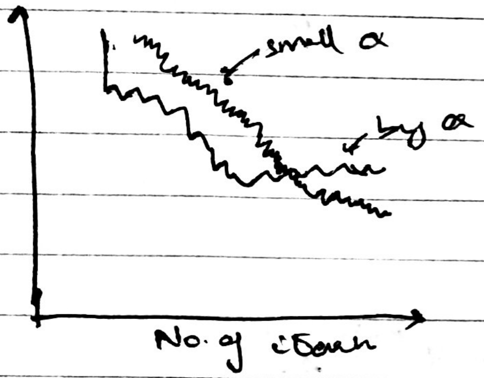
Stochastic gradient descent convergence

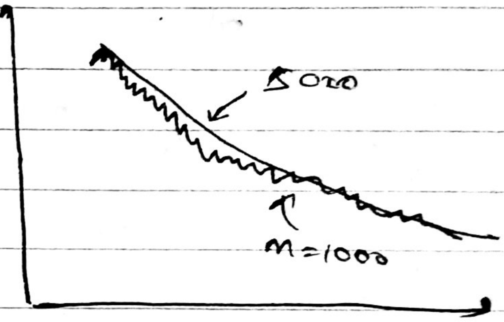
$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$

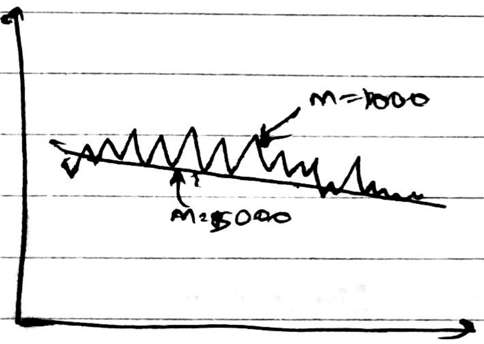
During learning, compute cost $(\theta, (x^{(i)}, y^{(i)}))$ before updating θ using $(x^{(i)}, y^{(i)})$.

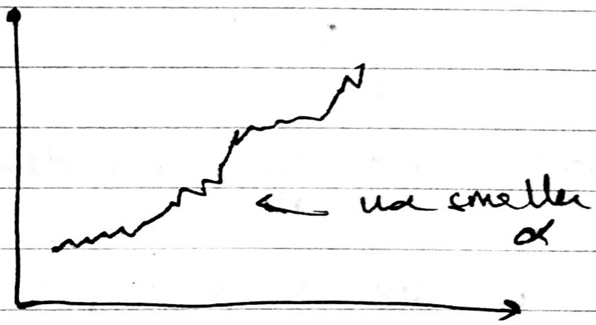
→ Every 1000 iterations plot $cost(\theta, (x^{(i)}, y^{(i)}))$ averaged over the last 1000 examples processed by algorithm.

You might see different trends









Stochastic gradient descent, if you want it to converge to a global minimum, then you could slowly decrease α over time.

$$\text{eg } \alpha = \frac{\text{const1}}{\text{iterationNo} + \text{const2}}$$

Online Learning

We may use logistic or neural.
for this, we use ~~logistic~~ ~~[illegible]~~ regression.

we want to learn $p(y=1 \mid x; \theta)$
price

Repeat forever {
get (x, y) corresponding to user.
update θ using (x, y)
$\rightarrow \theta_j := \theta_j - \alpha\,(h_\theta(x) - y).\,x_j \quad \ldots (j = 0 \ldots n)$
y

~~Map Reduce~~

PHOTO OCR
(Photo optical character recognition).

Photo OCR pipeline

1) Text detection.
2) character segmentation.
3) character classification
4). check validity $\longrightarrow$ (Clearing ≈ clearing)
However, it is ignored in this video.

Img $\rightarrow$ Text detection $\rightarrow$ Character segmentation $\rightarrow$ Character recognition

For pedestrian detection, we use ~~2t~~ classifier (82 x 36 pixel)
$y=1$ contains images of pedestrian
$y=0$ contains random images
we build a neural network or something like that.

$\frac{200 \times 200}{4 \times 4}$ 2500 $\frac{300 \times 300}{3 \times 3}$ $\frac{250 \times 250}{...}$ ~~...~~ 1 sec — 10 sec $\frac{10 \times 10,000}{60 \times 60 \times 24}$ $\frac{1000}{36 \times 8}$

Getting lots of Data

→ ~~You can take a low bias~~ For getting a high performance algo, take a low bias algo and train it on a massive training set.

Artificial data synthesis - Creating new data from scratch.

2 ways

1) use new set.

2). Synthesizing by introducing distortions to existing.

→ Make sure that you have a low bias. – Plot learning curve for it!

→ Increase the no. of hidden units until you have a low-bias

To get More data

- Artificial data synthesis
- collect / label it yourself
- "crowd source" (Eg Amazon Mechanical Turk)

CEILING ANALYSIS.

- Estimating errors due to each component.

m/4 → 1 msec
1 → 1/4 × 1000   120   1000×1000/32×2 × 2   250/62 × 10   1250/8   419   1,000,000/2   500.

Face Recognition