Stochastic Gradiant Descent

Gradiant Descent → very slow if # samples in training data is large.

Eg: #N > 3 million ? → GD will be very slow.

$$E(b_0, b_1, \dots b_d) = \frac{1}{2N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2.$$

Repeat until convergence {

$$b_j := b_j - \alpha \boxed{\frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)\, x_{ij}} \quad (\text{for } j = 0, 1, 2 \dots d)$$

}

$\frac{\partial E(b_0, b_1 \dots b_d)}{\partial b_j}$ ↓

∑ - loop for N to look at all the training samples.

N = 300 000 000

↓

Batch GD.

- looking at all training data/samples.
- read/fill these 300000000 records into memory to computer the derivative term.
- very slow if N is very large.

Can we do faster ?

- Yes : looking only at 1 training sample in each iteration.

Stochastic GD:

$$\hat{y}_i = b_0 x_0 + b_1 x_1 + \dots + b_d x_d$$

$$\text{cost}(B, (x_i, y_i)) = \frac{1}{2} (\hat{y}_i - y_i)^2$$

(single data sample.)

x	y
x_{11}	y_1
x_{21}	y_2
x_{31}	y_3
⋮	⋮
x_N	y

$$E(b_0, b_1, \dots b_d) = \frac{1}{N} \sum_{i=1}^{N} \text{cost}(B, (x_i, y_i))$$

$$= \frac{1}{N} \sum_{i=1}^{N} \left[\frac{1}{2} (\hat{y}_i - y_i)^2 \right]$$

→ Scan through the training example → modify param $(b_0, b_1, \dots b_d)$ one by one

(2)

Steps:

(1) Randomly Shuffle dataset. } → (Safer) Better to shuffle randomly to make the algo converge faster.

(2) Repeat {

- depends on Training data ⇒ (1 ... 10)
- for large N, one iteration is enough.

for $i = 1, 2, \dots N$ (training samples)

{

$b_j := b_j - \alpha \underbrace{(\hat{y}_i - y_i) \cdot x_{ij}}$ → only 1 sample at a time.

(for $j = 0, 1, 2, \dots d$)

}

}

modify $(b_0, b_1 \dots b_d) \longrightarrow \,,, \longrightarrow \,,,$

$(x_{11}, y_1), (x_{21}, y_2), (x_{31}, y_3) \dots$ — random samples

$\frac{\partial}{\partial b_j} cost\,[B, (x_i, y_i)]$.

$\frac{\partial}{\partial b_j} \left[\frac{1}{2}(\hat{y}_i - y_i)^2\right]$

$= (\hat{y}_i - y_i)\, x_{ij}$

- SGD doesn't converge as similar to GD.
- instead the parameters are closer to the global minimum. — good enough for practical applications.

B·GD	SGD
Use all N samples in one iteration.	Use only 1 sample in each iteration.

Mini Batch GD:

- in between GD and SGD.
- uses param "k" = mini-batch size. (smaller than N)
  - 10 or (2 to 100)

Example: $k = 10$ → use 10 samples in updating param in each iteration.

$\underbrace{(x_1, y_1) \dots (x_{10}, y_{10})}\ (x_{11}, y_{11}), (x_{12}, y_{12}) \dots$

used in 1 iteration.

(3) Mini Batch GD Algo:

$k = 10$; $N = 1000$ (samples)

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

$$b_j := b_j - \alpha \frac{1}{10} \sum_{m=i}^{i+9} (\hat{y}_m - y_m) x_{mj}$$

(for $j = 0, 1, 2, \ldots d$)

} }

↓ sums only 10 training data.

Why k-examples than 1-example?

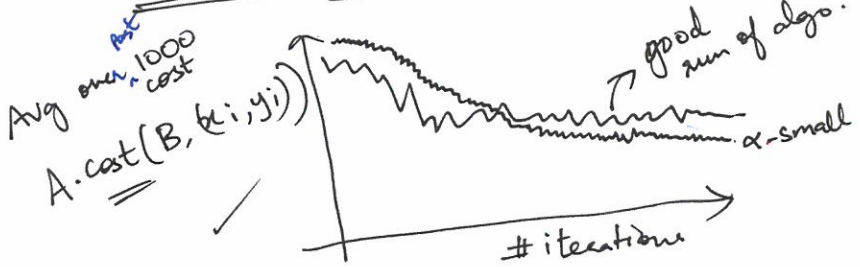
Vectorization

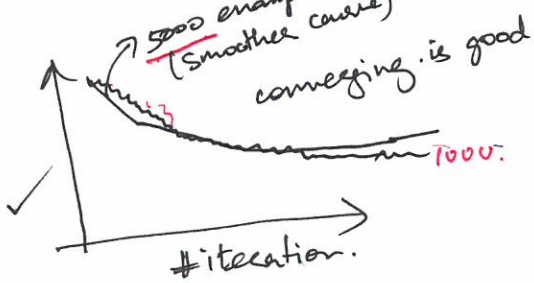
- parallelize your gradiant computation over $\underset{k}{10}$ samples.
- use linear algebra api's that uses vectorizations.

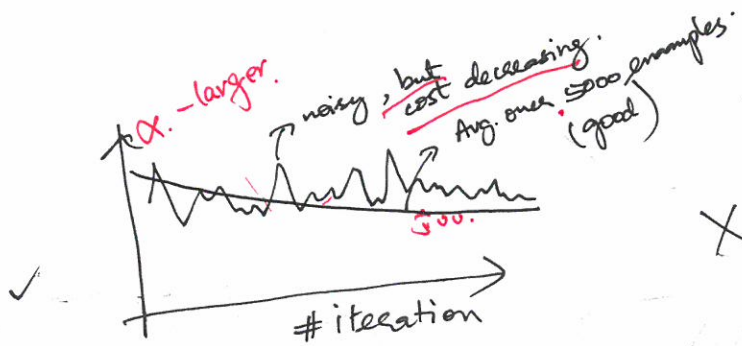
Disadv. of Mini-B GD:

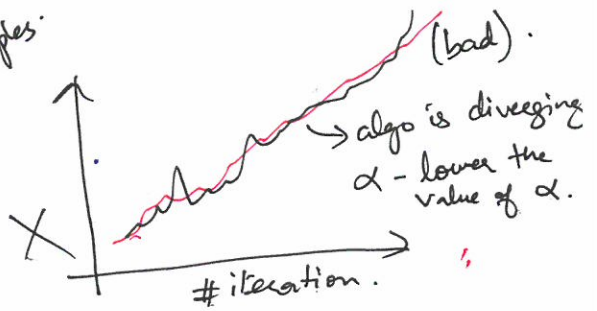
- Additional parameter to tune 'k'- minibatch size.
- choose k = 2 to 100.

Convergence of SGD: Avg. Cost $(B, (x_i, y_i))$ vs. iteration.









Note:- slowly decrease α over time → inorder for B to converge.

$$\alpha = \left(\frac{\text{constant 1}}{\text{iteration. No.} + \text{constant 2}} \right) \Rightarrow$$

const 1, const 2 — extra parameters.

fix vs. ~ 0.1.