deeplearning.ai

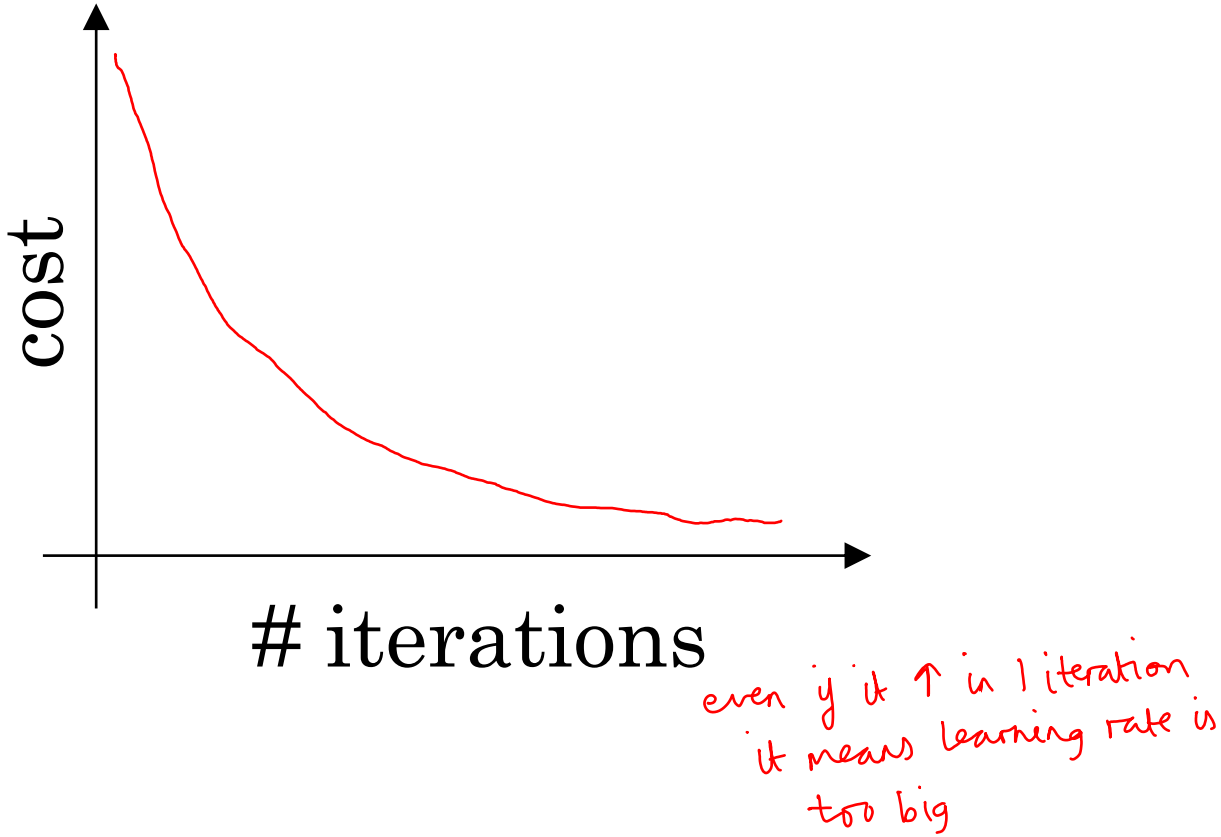
Optimization Algorithms

Understanding mini-batch gradient descent

Training with mini batch gradient descent

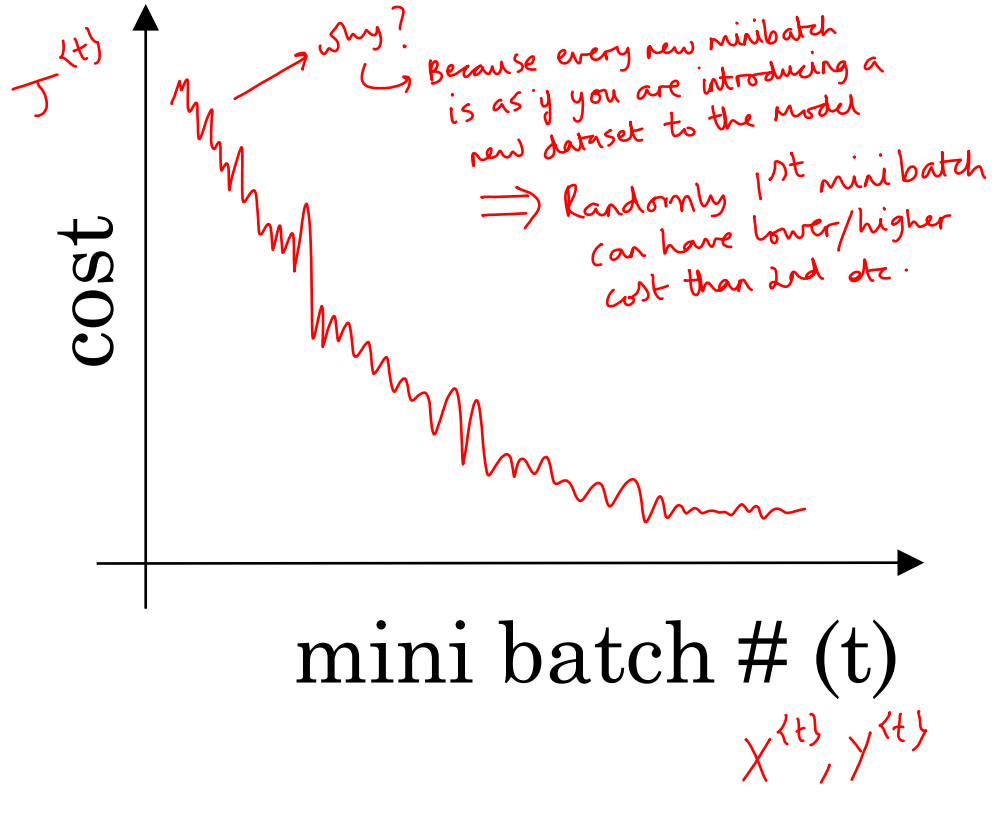
for every iteration of full data set, you expect cost to ↓

Batch gradient descent



even if it ↑ in 1 iteration it means learning rate is too big

Mini-batch gradient descent



why?
↳ Because every new minibatch is as if you are introducing a new dataset to the model
⇒ Randomly 1st mini batch can have lower/higher cost than 2nd etc.

$X^{\{t\}}, Y^{\{t\}}$

Andrew Ng

Choosing your mini-batch size

If minibatch size = m → Batch gradient descent

If minibatch size = 1 → Stochastic gradient descent

$(X^{\{1\}}, Y^{\{1\}}) = (X^{(1)}, Y^{(1)})$

$(X^{\{2\}}, Y^{\{2\}}) = (X^{(2)}, Y^{(2)})$

etc.

In practice → Between 1 & m

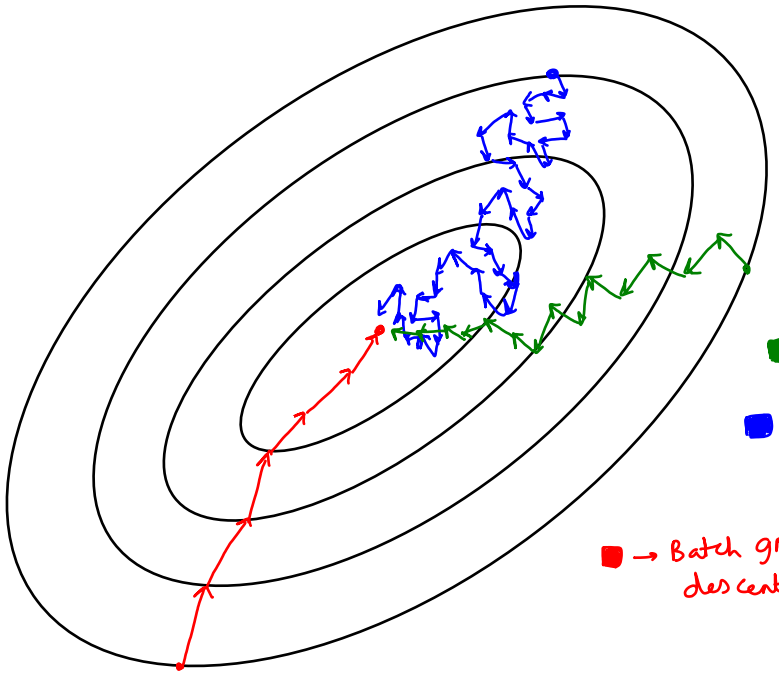
↳ Batch gradient descent → slow (size is large)

↳ Stochastic → make fast progress (per example)

↳ noise can be regulated with ↓ α

CON Loose all your speed because now you're going sample by sample Instead of using a vector!

↳ which is inherently faster (optimized for GPU/CPU)



■ → Minibatch

■ → Stochastic

(In stochastic, convergence never happens, But it will reach the bottom Area & keep oscillating there)

■ → Batch gradient descent

Choosing your mini-batch size

- If training set small, $m \leq 2000$
  use batch
- minibatch size,
  64, 128, 256, 512, ...] → uses computer memory byte size
- Make sure
  $X^{\{t\}}, Y^{\{t\}}$ fits in CPU/GPU mem
- Make it a hyper param & check which one works best