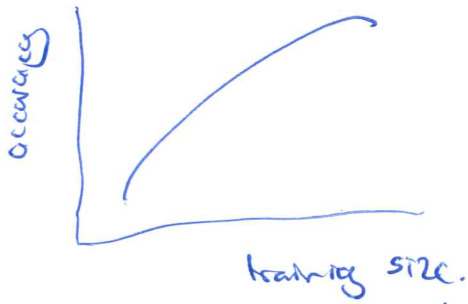
Week 10

large scale machine learning - learning with large datasets

Machine learning and data

classify between two confusable words.
e.g. (to, two, too), (then, than)



for breakfast I ate <u>two</u> eggs.

"It's not who has the best algorithm that wins. It's who has the most data"

---

learning with large datasets

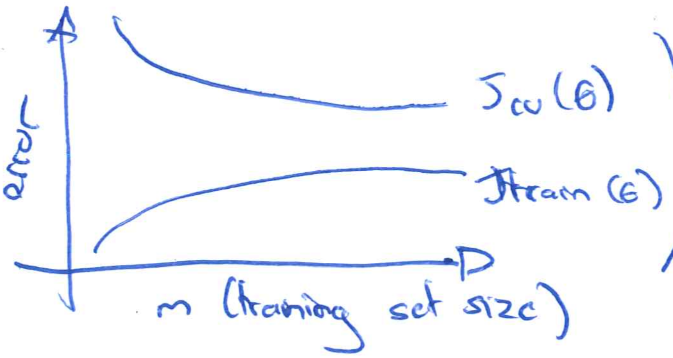
m = 100,000,000 (lets say website)

and train gradient descent

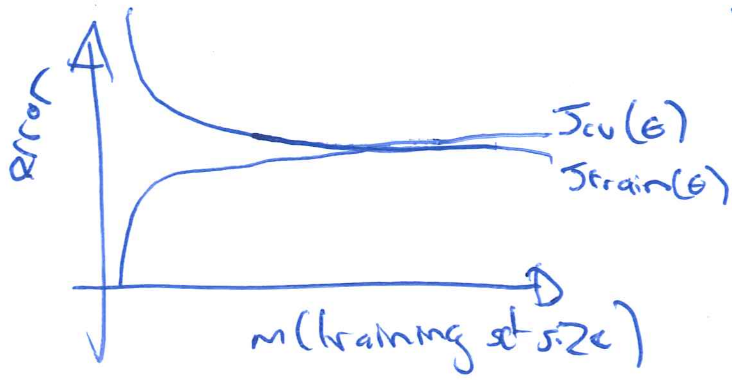
$$\rightarrow \theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}$$

100 000 000

why not use m=1000?



→ high variance algorithm → more data!

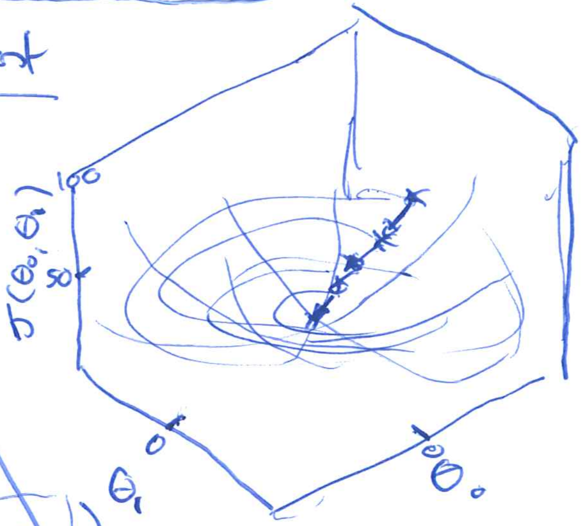


→ high bias → less data.
(add extra features, or hidden units etc

Week 10

Large scale machine learning - stochastic gradient descent

Linear regression with gradient descent



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

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

cost function

Gradient Descent: ("Batch or all)

Repeat {

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

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

}

If m is large, gradient descent is computationally expensive.

$m = 300{,}000{,}000$ ← population of US.

Batch gradient descent

"All"

Our Alternative is: stochastic gradient descent → takes a more random path to the global minimum

↳ does not converge (in the same sense as batch gradient descent

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

$$J_{train}(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{cost}(\theta, (x^{(i)}, y^{(i)}))$$

1) Randomly shuffle dataset. ← interest of safety

2) Repeat { repeat inner loop 1-10 times

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

$$\theta_j := \theta_j - \alpha \underbrace{\boxed{\left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x_j^{(i)}}}_{\frac{\partial}{\partial\theta_j} \text{Cost}(\theta, (x^{(i)}, y^{(i)}))}$$

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

}

← so rather than waiting for the sum to modify parameters, we modify with each training example

}

$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \dots, n$

↑ each looks at the next training example & takes another step in parameter space → modify's parameters to fit data

Week 10

Large scale machine learning - mini batch gradient descent

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 = 10 2 - 100

Get $\boxed{b = 10}$ examples $(x^{(i)}, y^{(i)}) \ldots (x^{(i+9)}, y^{(i+9)})$

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

$i := i+10$

Mini-batch gradient descent

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 $j = 0, \ldots, n$)
  }
}

b examples ← can parallelize the "mini-batch" computations
1 example

Mini batch will outperform stochastic gradient descent only if you have a good vectorized implementation.

Week 10

Large scale machine learning - stochastic gradient descent convergence

Checking for convergence:

Batch gradient descent:

Plot $J_{train}(\theta)$ as a function of the number of iterations of gradient descent

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

Stochastic gradient descent:

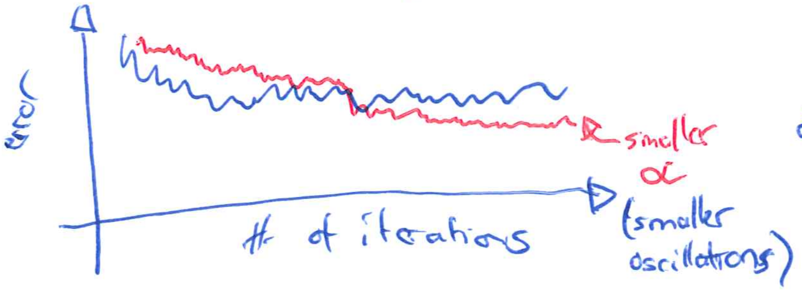
$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

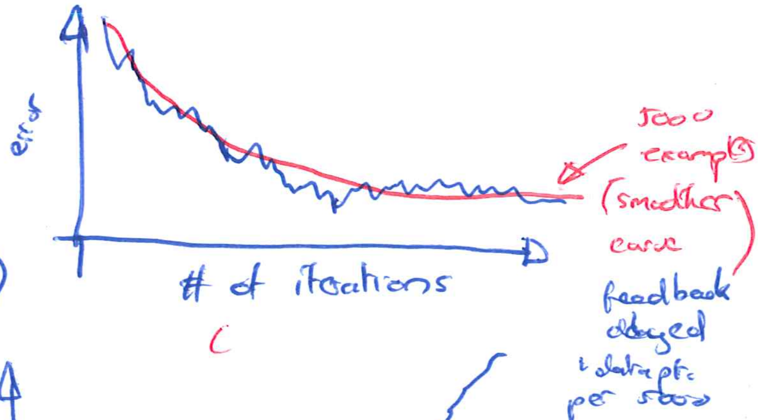
during learning, compute $cost(\theta, (x^{(i)}, y^{(i)}))$ before updating θ using $(x^{(i)}, y^{(i)})$. (right before we update our parameters θ; compute cost on current example $(x^{(i)}, y^{(i)})$)

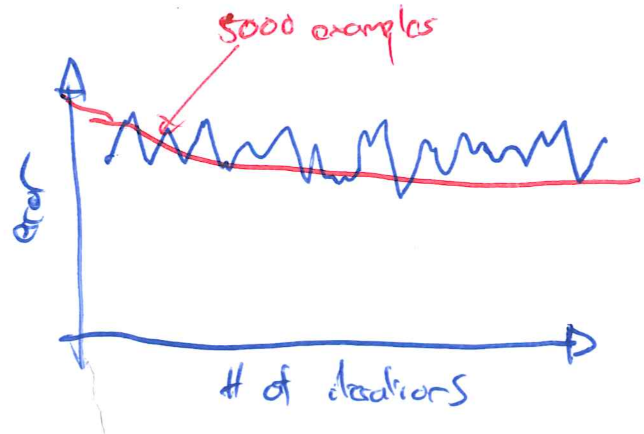
every 1000 iterations (say), plot $cost(\theta, (x^{(i)}, y^{(i)}))$ averaged over the last 1000 examples processed by the algorithm.

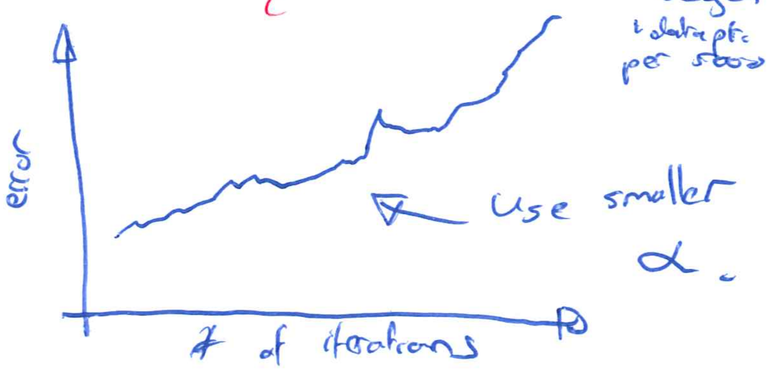
Checking for convergence

Plot $cost(\theta, (x^{(i)}, y^{(i)}))$, averaged over the last 1000 (say) examples









Week 10

Large scale machine learning - stochastic gradient descent convergence

Learning rate α is typically held constant. Can slowly decrease α over time if we want θ to converge e.g $\alpha = \frac{\text{const1}}{(\text{iteration\#}) + \text{const2}}$

$\alpha \to 0$.

Week 10

Large scale machine learning - online learning

Online learning

Shipping service website where user comes, specifies origin and destination, you offer to ship their package for some asking price, and 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.
price → we can use logistic regression

Repeat forever {
  Get (x, y) corresponding to user
  Update θ using (x, y):
    $\theta_j := \theta_j - \alpha (h_\theta(x) - y) \cdot x_j \quad (j = 0, \dots, n)$
}

~~Disc~~ We are discarding the idea that there is a training set ie only single (x, y)

Can adapt to changes in user preferences

Other online learning example

Product search (learning to search)

- 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 phone, how many words in query match description of phone etc.

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

- Learn $p(y=1 \mid x; \theta)$ "problem of learning CTR"
- Use to show user the 10 phones they're most likely to click on.

Other examples: choosing special offers to show user; customized selection of news articles; product recommendation

Week 10

Large scale machine learning - Map reduce and data parallelism

Map reduce

$m = 400\,000\,000 \rightarrow$ pretend $m = 400$

Batch gradient descent $\boxed{\theta_j := \theta_j - \alpha \frac{1}{400} \sum_{i=1}^{400} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}}$

machine 1: use $(x^{(1)}, y^{(1)}), \dots, (x^{(100)}, y^{(100)})$

= the same

$x^{(1)}, y^{(1)}$ … $x^{(m)}, y^{(m)}$

machine 1: use $(x^{(1)}, y^{(1)}) \dots (x^{(100)}, y^{(100)})$ — $\frac{1}{4}$ of training set

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

machine 2: use $(x^{(101)}, y^{(101)}), \dots, (x^{(200)}, y^{(200)})$ — $\frac{2}{4}$ of training set

$temp_j^{(2)} = \sum_{i=101}^{200} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$

etc etc

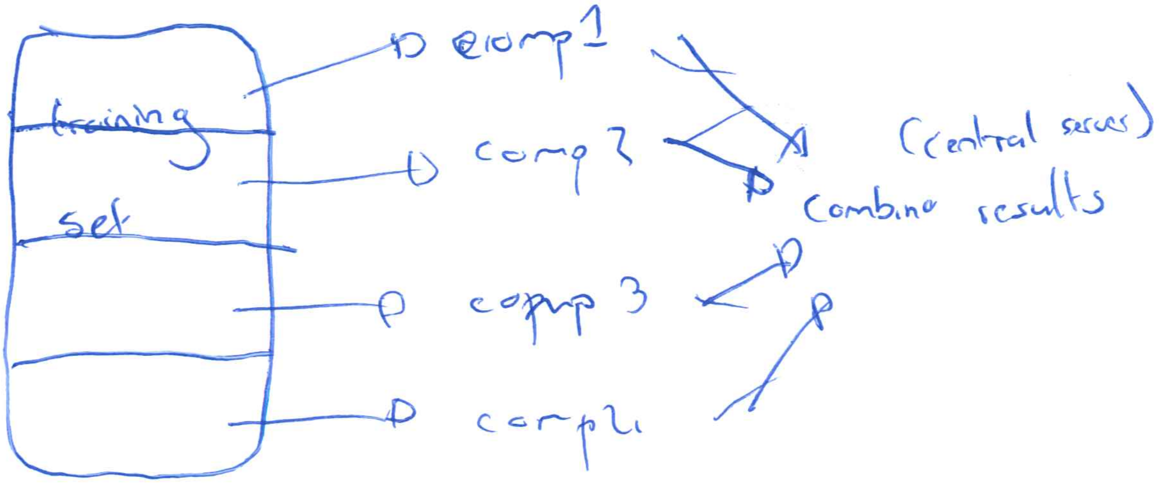
$\frac{3}{4}$

$\frac{4}{4}$

combine: $\boxed{\theta_j := \theta_j - \alpha \frac{1}{400} (temp_j^{(1)} + temp_j^{(2)} + temp_j^{(3)} + temp_j^{(4)})}$ $(j = 0, \dots, n)$

= the same

Map-reduce



Week 10

Large scale machine learning - map reduce & data parallelism

map-reduce & summation over the training set

Many learning algorithms can be expressed as computing sums of functions over the training set

e.g. for advanced optimization, with logistic regression, need:
eg train of

Multi-core machines