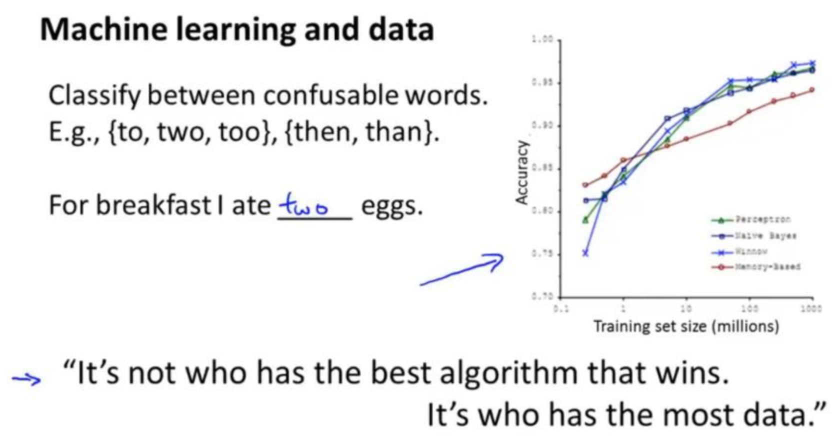
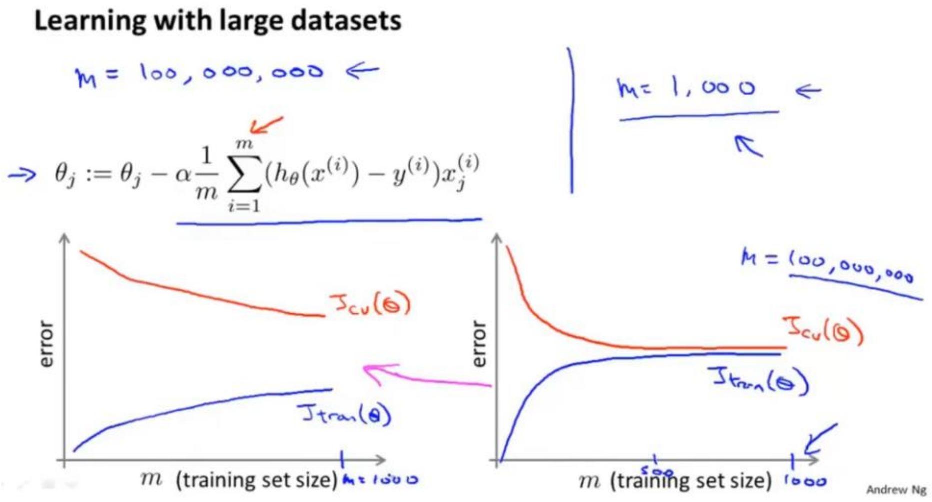
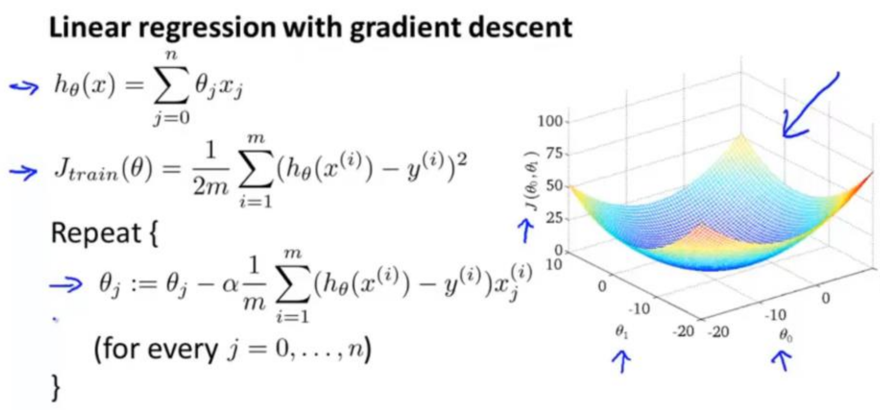
Large scale machine learning

1 Learning with large datasets





2 Stochastic gradient descent



Linear regression with gradient descent

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

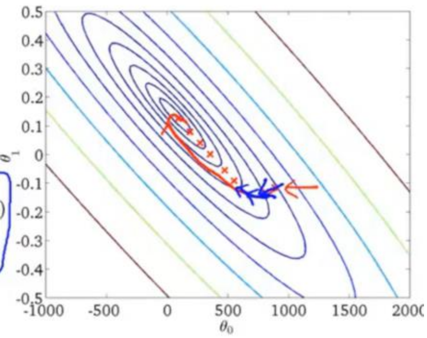
$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Repeat {

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

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

}



$M = 300,000,000$

Batch gradient descent

Batch gradient descent

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

Repeat {

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

$\frac{\partial}{\partial \theta_j} J_{train}(\theta)$

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

}

$m = 300,000,000$

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 dataset.
2. Repeat {
   for $i = 1, \ldots, m$ {
   $\Theta_j := \Theta_j - \alpha \boxed{(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}}$
   (for $j = 0, \ldots, n$)
   }
   }

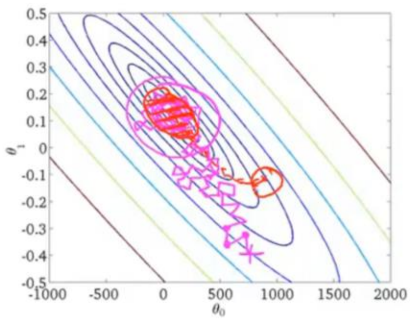
$\frac{\partial}{\partial \Theta_j} cost(\Theta, (x^{(i)}, y^{(i)}))$

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

Andrew Ng

Stochastic gradient descent

1. Randomly shuffle (reorder) training examples
2. Repeat { 1-10x
   for $i := 1, \ldots, m$ {
   $$\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$
   (for every $j = 0, \ldots, n$)
   }
   }



$m = 300,000,000$

3 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 $(x^{(i)}, y^{(i)}), \dots (x^{(i+9)}, y^{(i+9)})$

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

$i := i + 10$

Mini-batch gradient descent

Say $\boxed{b = 10}, m = \boxed{1000}$.
Repeat {
  for $i = 1, 11, 21, 31, \dots, \boxed{991}$ {

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

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

→ b examples
→ 1 example
Vectorization

$m = 300,000,000$

$b = 10$

4 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.
  - $\boxed{J_{train}(\theta)} = \frac{1}{2m} \boxed{\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2}$

$m = 300,000,000$

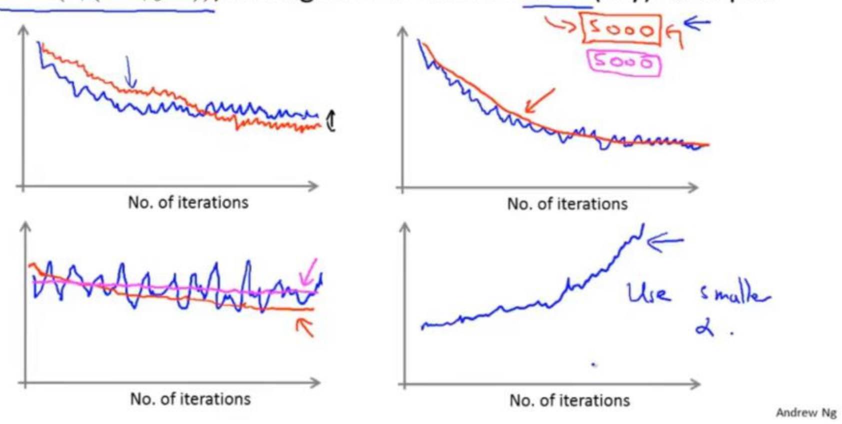
- Stochastic gradient descent:
  - $cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$
  - During learning, compute $cost(\theta, (x^{(i)}, y^{(i)}))$ before updating θ using $(x^{(i)}, y^{(i)})$.
  - Every 1000 iterations (say), plot $cost(\theta, (x^{(i)}, y^{(i)}))$ averaged over the last 1000 examples processed by algorithm.

$(x^{(i)}, y^{(i)}), (x^{(i+1)}, y^{(i+1)}), \dots$

Andrew Ng

Checking for convergence

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



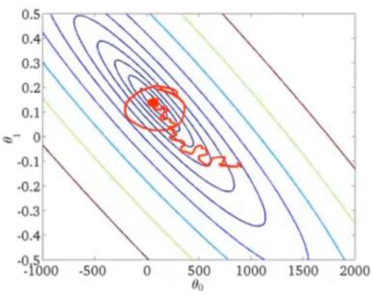
Andrew Ng

Stochastic gradient descent

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

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m} cost(\theta, (x^{(i)}, y^{(i)}))$$

1. Randomly shuffle dataset.
2. Repeat {
   for $i := 1, \ldots, m$ {
   $\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$
   (for $j = 0, \ldots, n$)
   }
   }



Learning rate α is typically held constant. Can slowly decrease α over time if we want θ to converge. (E.g. $\alpha = \frac{\text{const1}}{\boxed{\text{iterationNumber}} + \text{const2}}$) $\alpha \to 0$

Andrew Ng

5 Online learning

Online learning

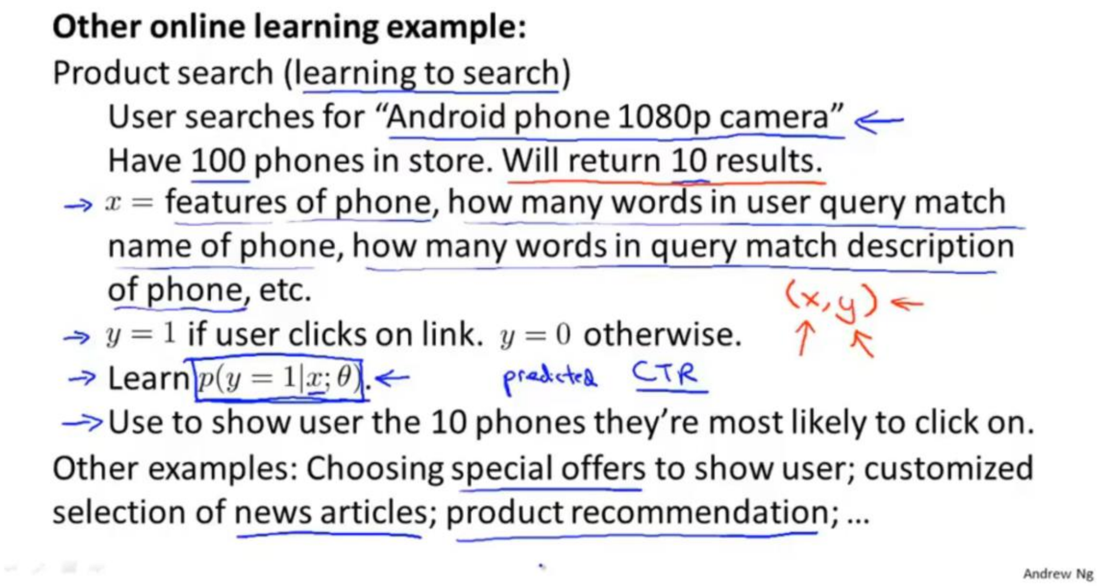
Shipping service website where user comes, specifies origin and destination, you offer to ship their package for some asking price, and users sometimes choose to use your shipping service ($y = 1$), sometimes not ($y = 0$).

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

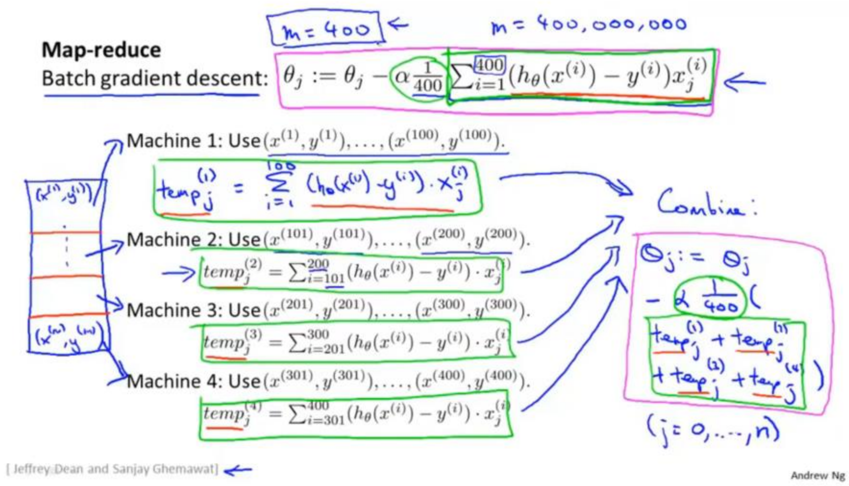
price   logistic regression

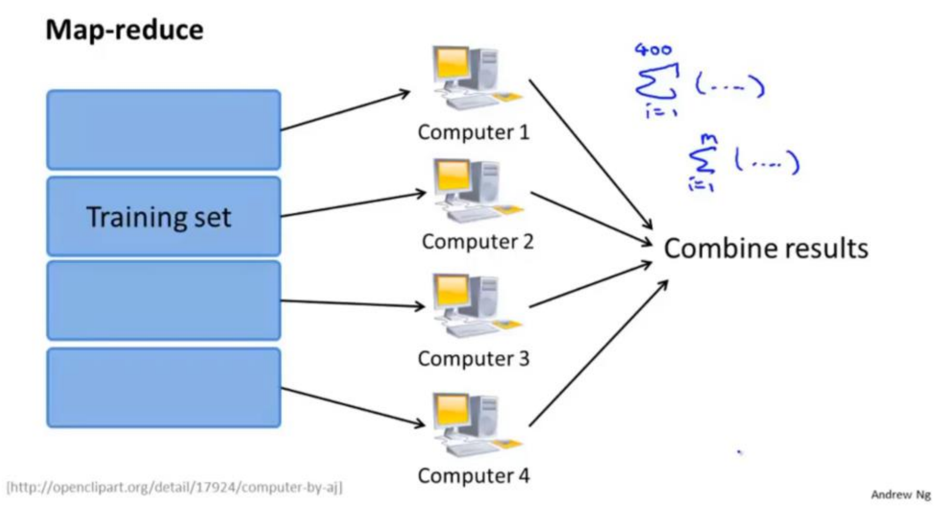
Repeat forever {
  Get $\boxed{(x,y)}$ corresponding to user. $\cancel{(x^{(i)}, y^{(i)})}$
  Update θ using $\boxed{(x,y)}$:
  $\rightarrow \theta_j := \theta_j - \alpha(h_\theta(x) - y) \cdot x_j$ $(j = 0, \ldots, n)$
}
$\rightarrow$ Can adapt to changing user preference.

Andrew Ng



6 Map-reduce and data parallelism





Map-reduce and 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:

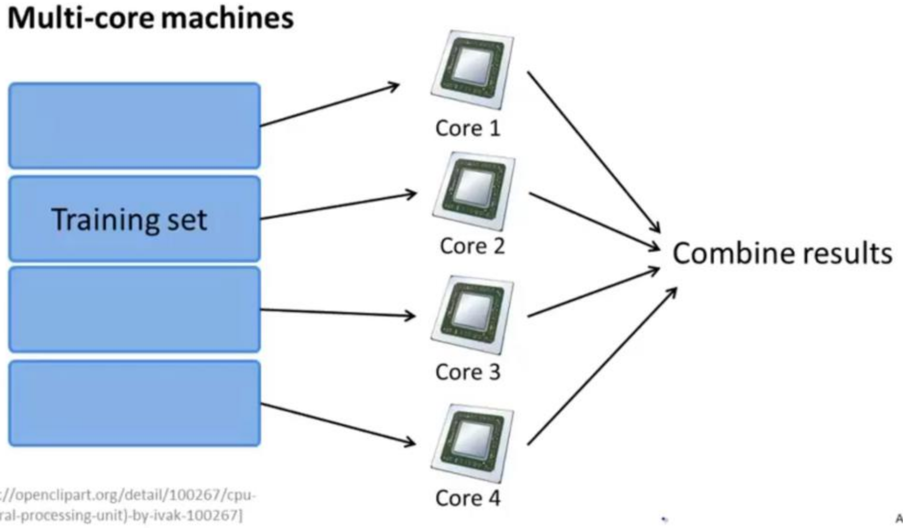
$$\underline{J_{train}(\theta)} = -\frac{1}{m}\sum_{i=1}^{m} \underline{y^{(i)}\log h_\theta(x^{(i)}) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))}$$

$$\underline{\frac{\partial}{\partial\theta_j} J_{train}(\theta)} = \frac{1}{m}\sum_{i=1}^{m} \underline{(h_\theta(x^{(i)}) - y^{(i)})\cdot x_j^{(i)}}$$

$temp^{(i)}$ $temp_j^{(i)}$

Andrew Ng

Multi-core machines



[http://openclipart.org/detail/100267/cpu-(central-processing-unit)-by-ivak-100267]

Andrew Ng