

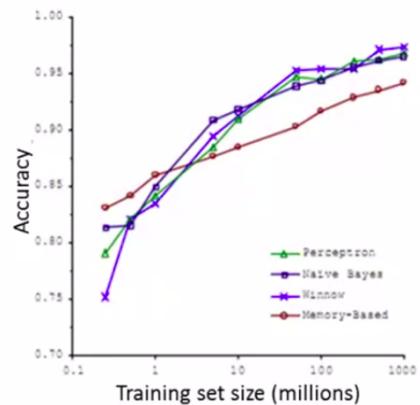
Week 10 - Lecture 1

Learning with Large Datasets

Machine learning and data

Classify between confusable words.
E.g., {to, two, too}, {then, than}.

For breakfast I ate two eggs.

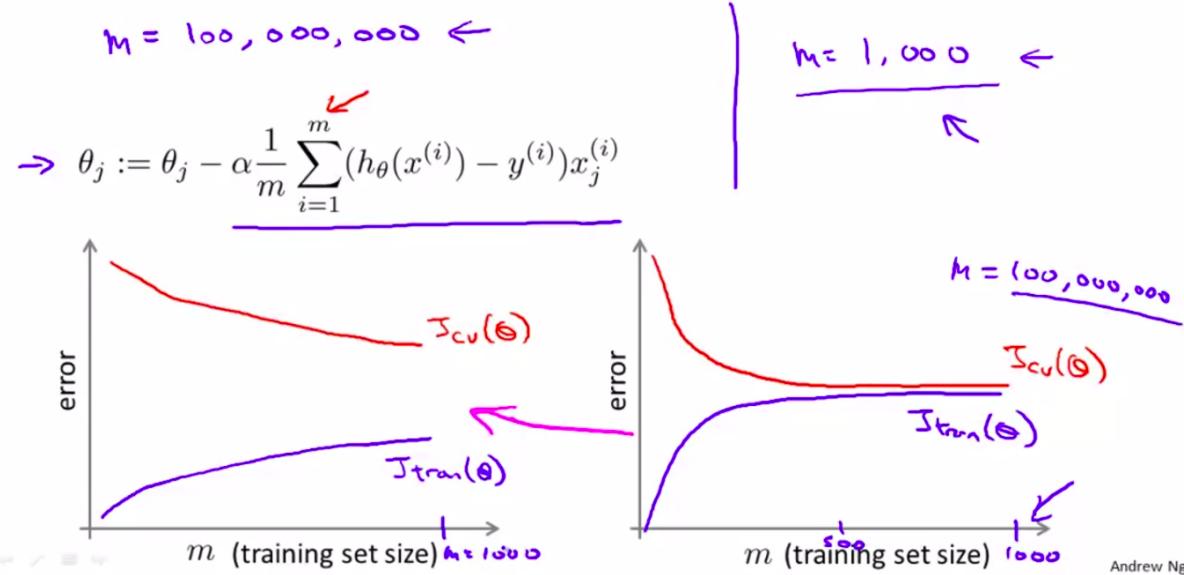


"It's not who has the best algorithm that wins.
It's who has the most data."

[Figure from Banko and Brill, 2001]

Andrew Ng

Learning with large datasets



Stochastic Gradient Descent

Remember the **Gradient Descent Algorithm**:

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 {

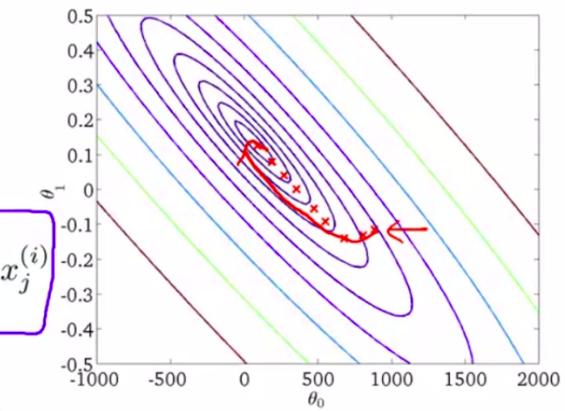
$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)} \right]$$

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

}

$$M = 300,000,000$$

Batch gradient descent



Andrew Ng

Batch gradient descent

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

Repeat {

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

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

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

}

Stochastic gradient descent

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

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

1. Randomly shuffle dataset.

2. Repeat {

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

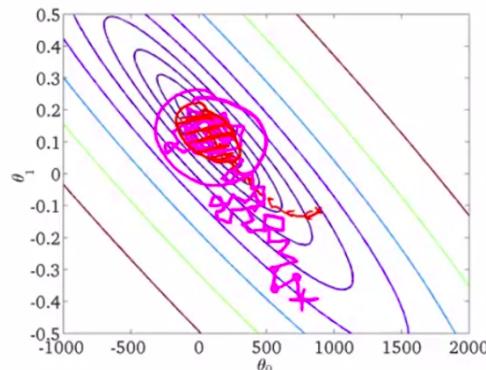
$$\theta_j := \theta_j - \frac{\partial}{\partial \theta_j} \underset{\text{(for } j=0, \dots, n\text{)}}{\uparrow} \underset{\text{cost}}{\uparrow} (\theta, (x^{(i)}, y^{(i)}))$$

$$\} \quad \rightarrow \frac{\partial}{\partial \theta_j} \underset{\text{cost}}{\uparrow} (\theta, (x^{(i)}, y^{(i)}))$$

Andrew Ng

Stochastic gradient descent

- 1. Randomly shuffle (reorder) training examples
- 2. Repeat {
 for $i := 1, \dots, m$ {
 $\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$
 (for every $j = 0, \dots, n$)
 } }
 $m = 300,000,000$



Andrew Ng

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

$$\begin{aligned} b &= \text{mini-batch size.} & b &= 10. & 2-100 \\ \text{Get } b &= 10 \text{ examples } & (x^{(i)}, y^{(i)}), \dots, (x^{(i+9)}, y^{(i+9)}) \\ & \theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{i+9} (h_\theta(x^{(k)}) - y^{(k)}) \cdot x_j^{(k)} \end{aligned}$$

Andrew Ng

Mini-batch gradient descent

Say $b = 10$, $m = 1000$.

Repeat {

→ for $i = 1, 11, 21, 31, \dots, 991$ {
 → $\theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{i+9} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)}$
 (for every $j = 0, \dots, n$)
 }
 }

Andrew Ng

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.

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

$M = 300, 500, 800$

→ Stochastic gradient descent:

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

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

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

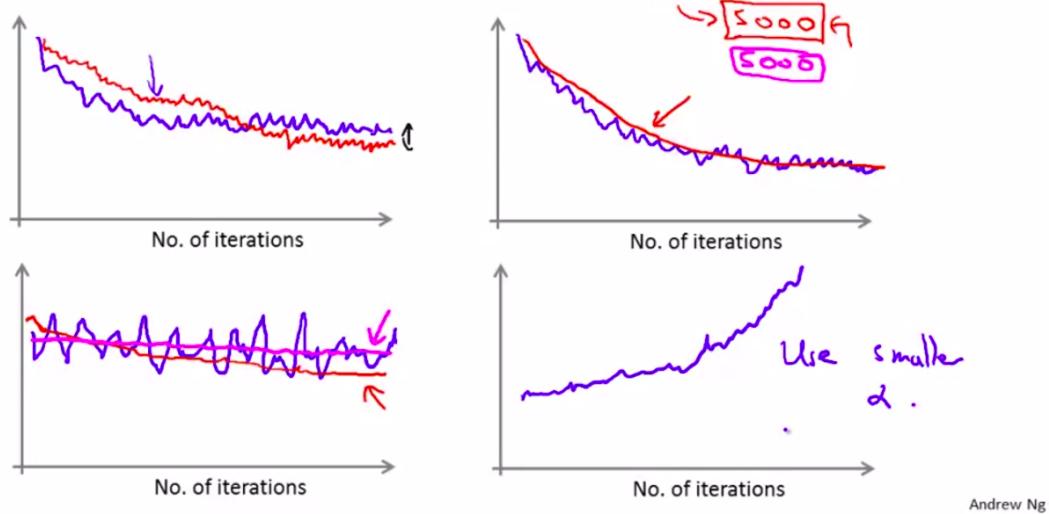
→ Every 1000 iterations (say), plot $cost(\theta, (x^{(i)}, y^{(i)}))$ averaged over the last 1000 examples processed by algorithm.

Andrew Ng

- **upper left:** Stochastic Gradient Descent convergence with different learning rates (red line has a smaller learning rate).
- **upper right:** Convergence with different number of examples to calculate the mean of the cost function (the red line represents the curve with a higher number of examples).
- **bottom left:** looks like the cost is not decreasing, i.e., the algorithm is not learning (the blue line is too noisy). The red line represents the same curve, just by looking at a higher number of examples. This might indicate that the algorithm is learning, just at a small convergence rate. If the curve remains flat, it's a stronger indicator that the algorithm isn't learning.
- **bottom right:** the algorithm is diverging. So, to try to remedy this, we should try using a smaller value of α .

Checking for convergence

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



Note: If we want stochastic gradient descent to really converge to the global minimum, we can try decreasing the value of α over time.

Stochastic gradient descent

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

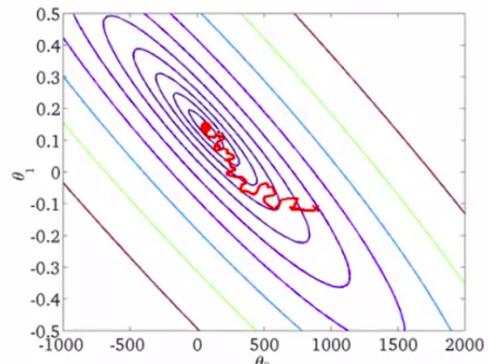
$$J_{\text{train}}(\theta) = \frac{1}{2m} \sum_{i=1}^m \text{cost}(\theta, (x^{(i)}, y^{(i)}))$$

1. Randomly shuffle dataset.

2. Repeat {

```

    for i := 1, ..., m      {
        theta_j := theta_j - alpha * (h_theta(x^{(i)}) - y^{(i)}) * x_j^{(i)}
        for j = 0, ..., n
    }
}
```



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

Andrew Ng

Advanced Topics

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.

Repeat forever {
Get (x, y) corresponding to user.
Update θ using $\nabla_{\theta} p(y = 1|x; \theta)$:
 $\rightarrow \theta_j := \theta_j - \alpha (h_{\theta}(x) - y) \cdot x_j \quad (j=0, \dots, n)$
} Can adapt to changing user preference.

Andrew Ng

Example: Predict the *Click Through Rate* (map the clicks of users accessing a website)

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 $p(y = 1|x; \theta)$. ← predicted CTR

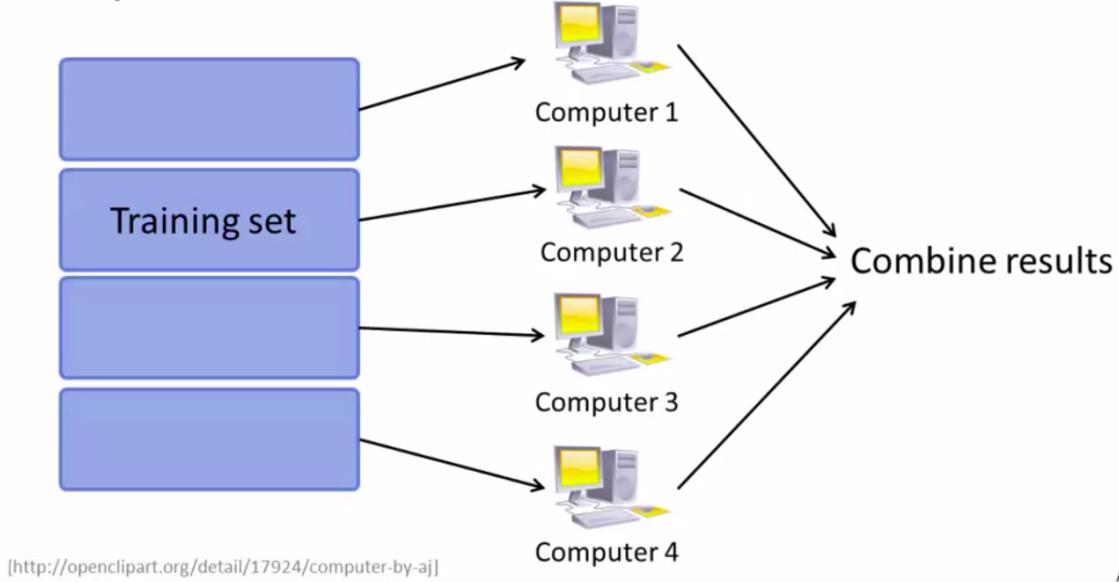
→ Use to show user the 10 phones they're most likely to click on.

Andrew Ng

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

Map Reduce and Data Parallelism

Map-reduce

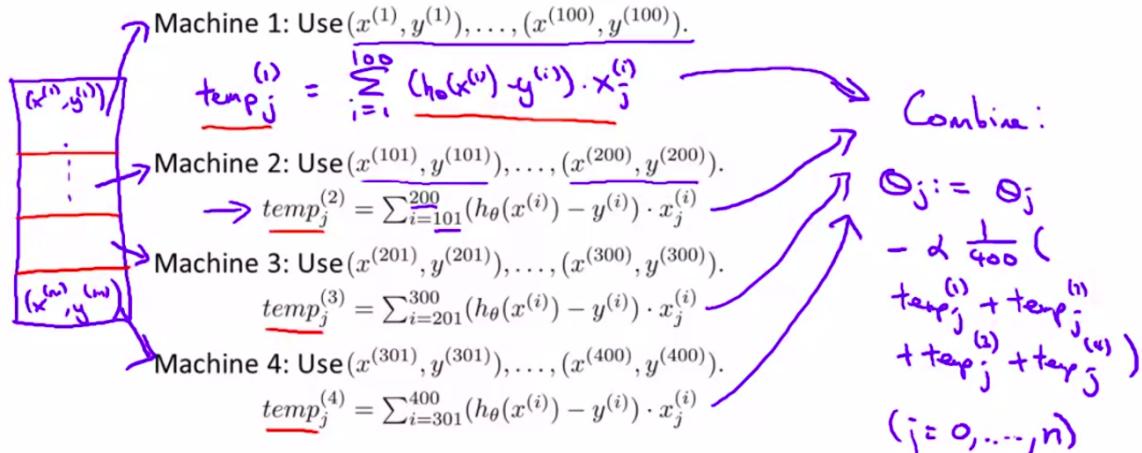


[<http://openclipart.org/detail/17924/computer-by-aj>]

Andrew Ng

Map-reduce

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



[[Jeffrey Dean and Sanjay Ghemawat](#)]

Andrew Ng

Map-reduce and summation over the training set

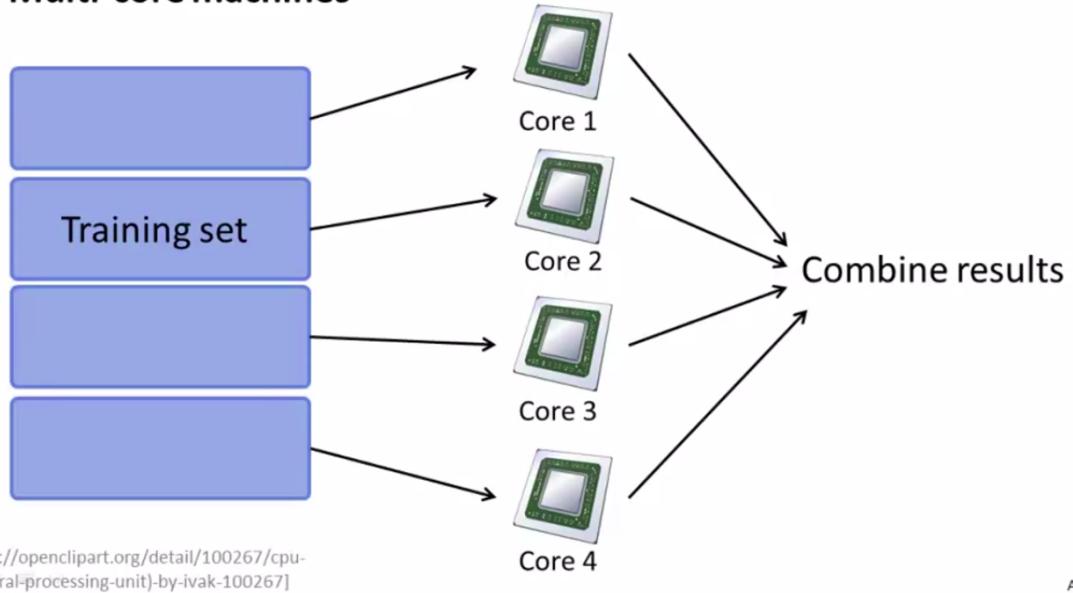
Many learning algorithms can be expressed as computing sums of functions over the training set.

E.g. for advanced optimization, with logistic regression, need:

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

$$\rightarrow \frac{\partial}{\partial \theta_j} J_{train}(\theta) = \frac{1}{m} \sum_{i=1}^m \underbrace{(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}}_{\textcircled{2}}$$

Multi-core machines



Andrew Ng

References

- [1] [Machine Learning - Stanford University](#) (<https://www.coursera.org/learn/machine-learning>)