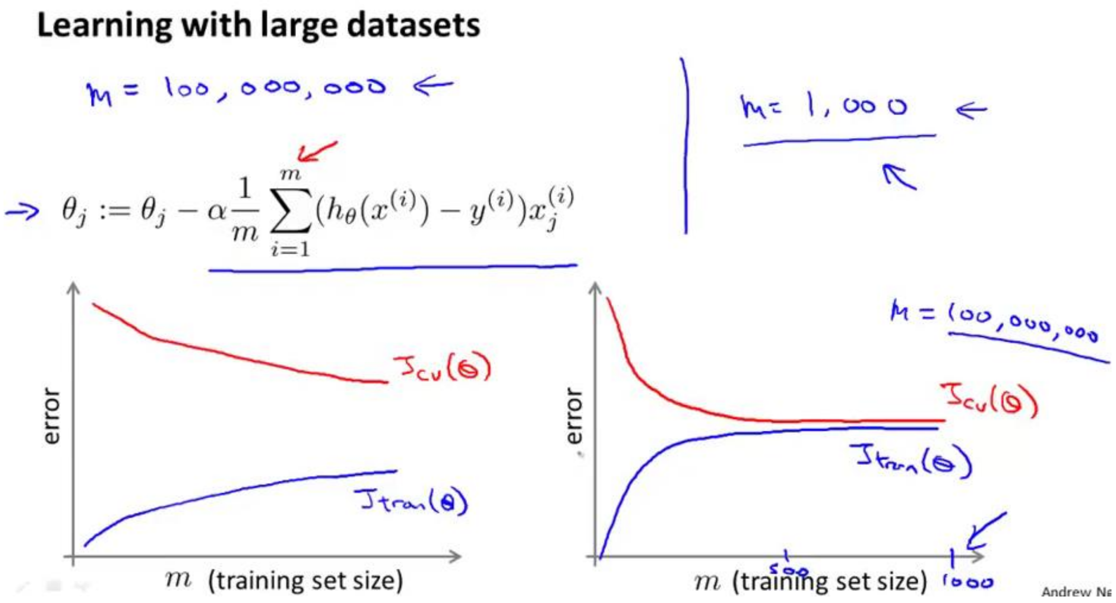


Learning with large datasets

$m = 100,000,000 \leftarrow$

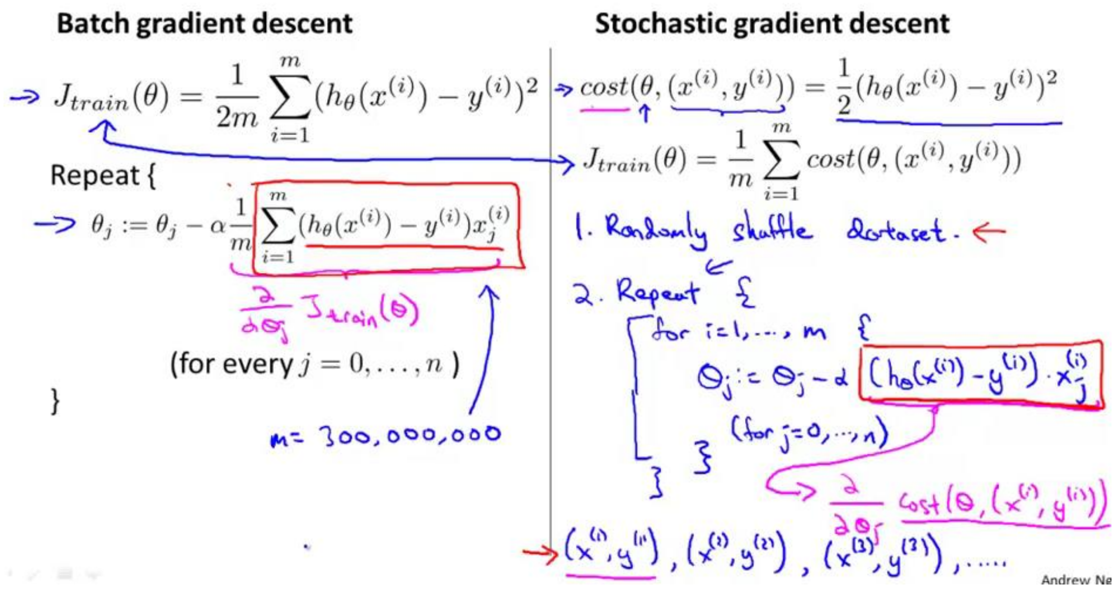
$$\rightarrow \theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$m = 1,000 \leftarrow$

高偏差时，增加数据量有用；　高方差时，增加数据量无用；

高偏差时，也可以 神经网络增加隐藏单元

随机梯度下降(stochastic gradient descent)



Batch gradient descent

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

Repeat {

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

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

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

}

$m = 300,000,000$

Stochastic gradient descent

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

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

1. Randomly shuffle dataset. $\leftarrow$

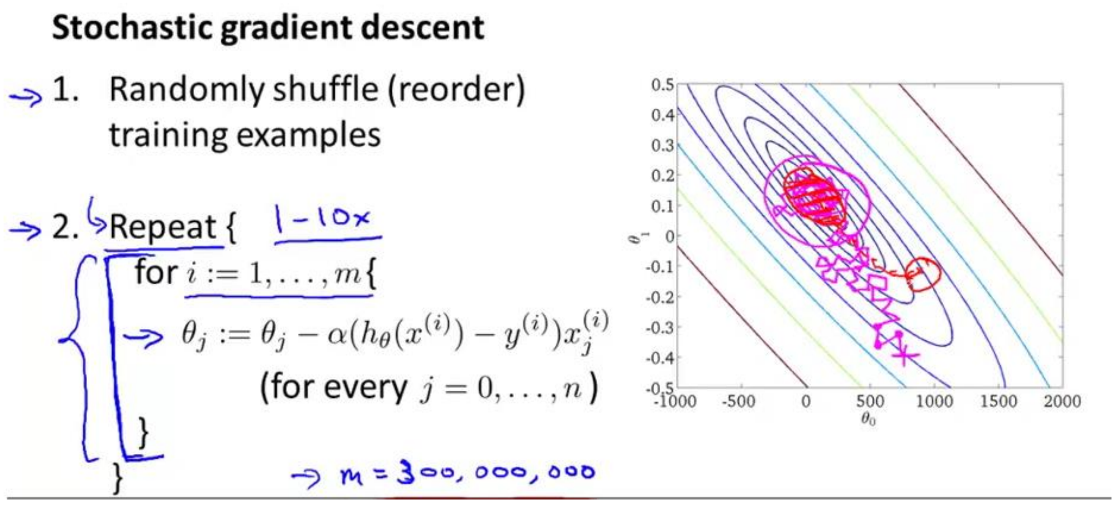
2. Repeat {

```
    for i=1,...,m {
        $\theta_j := \theta_j - \alpha \boxed{(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}}$
        (for j=0,...,n)
    }
}
```

$\frac{\partial}{\partial \theta_j} cost(\theta, (x^{(i)}, y^{(i)}))$

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

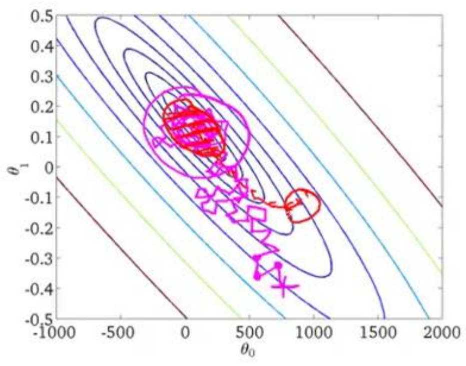
Andrew Ng



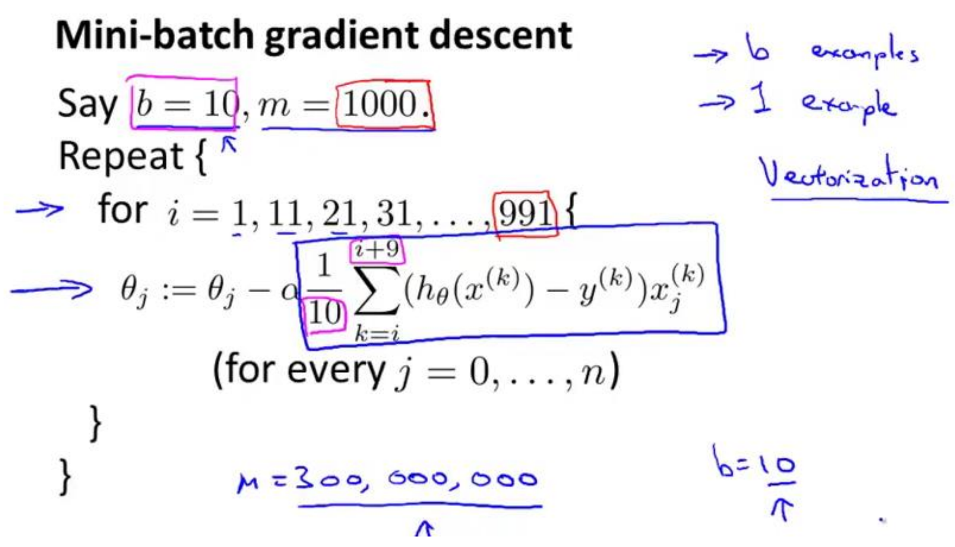
Stochastic gradient descent

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

→ $m = 300{,}000{,}000$



小批量梯度下降



Mini-batch gradient descent

→ b examples
→ 1 example
Vectorization

```
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 $j = 0, \dots, n$)
    }
}
```

$\underline{m = 300{,}000{,}000}$

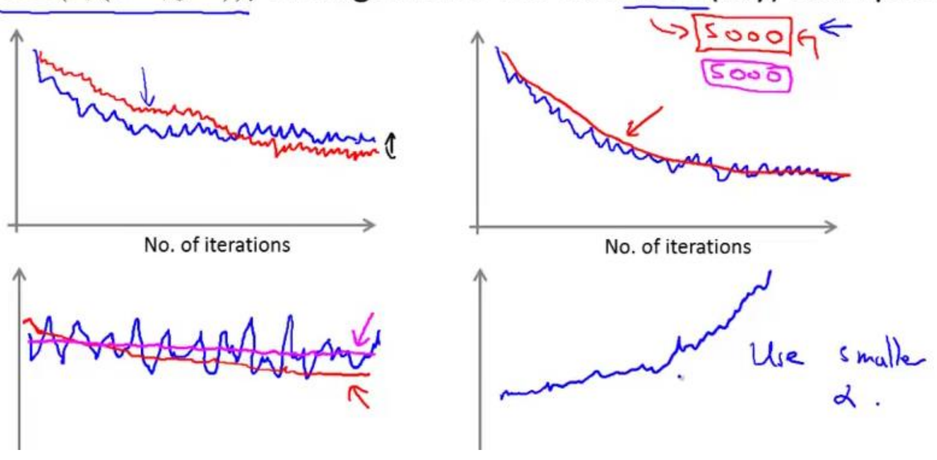
$b = \underline{10}$

Checking for convergence

- Batch gradient descent:
  - Plot $J_{train}(\theta)$ as a function of the number of iterations of gradient descent.
  - $\boxed{J_{train}(\theta)} = \frac{1}{2m}\boxed{\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2}$ $\quad \underline{M = 300,000,000}$
- Stochastic gradient descent: $\quad \underline{(x^{(i)}, y^{(i)})}, (x^{(i+1)}, y^{(i+1)}), \ldots$
  - $cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$
  - 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.

Checking for convergence

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

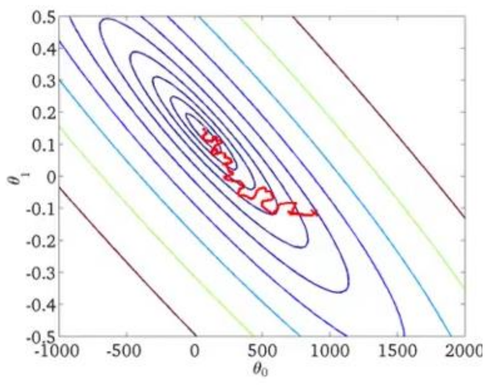


Stochastic gradient descent

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

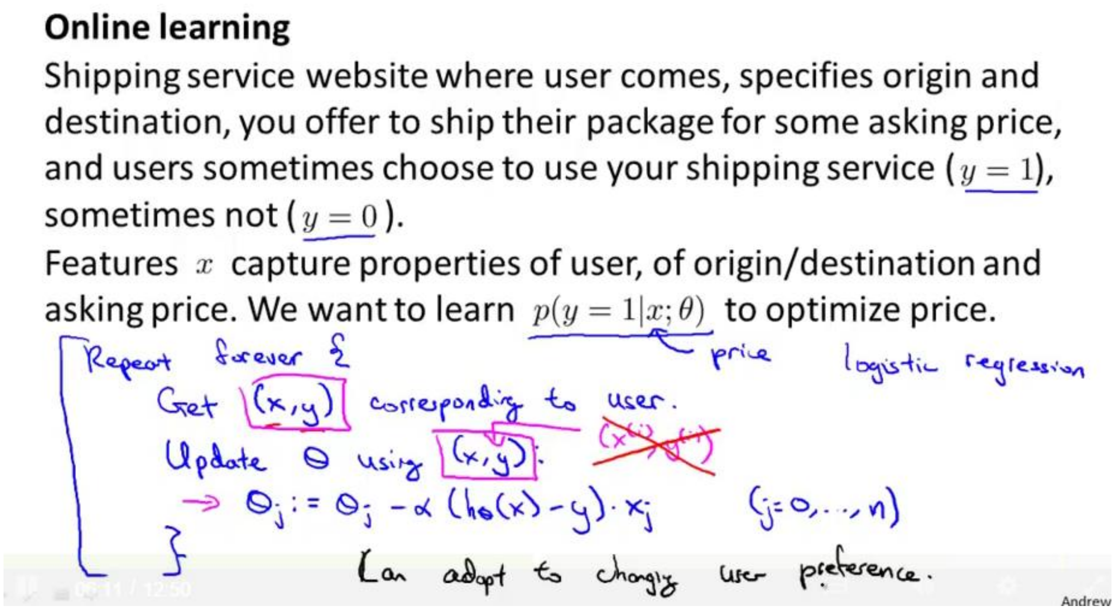
$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m} cost(\theta, (x^{(i)}, y^{(i)}))$$

```
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. $\alpha = \frac{\texttt{const1}}{\texttt{iterationNumber + const2}}$)

在线学习机制



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 ($y = 0$).

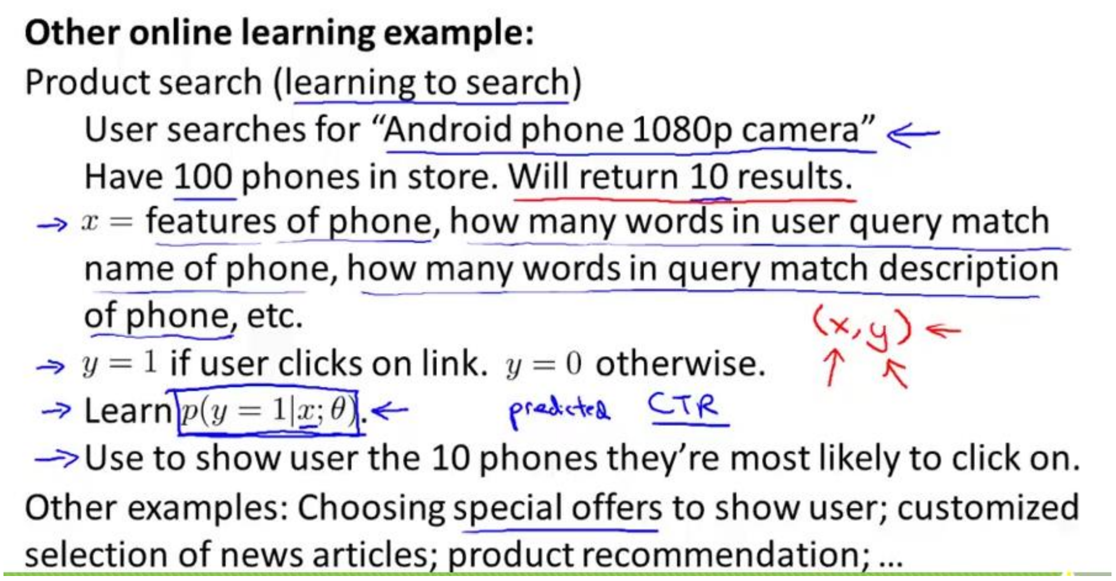
Features x capture properties of user, of origin/destination and asking price. We want to learn $p(y = 1|x;\theta)$ to optimize price.

price   logistic regression

```
Repeat forever {
    Get $\boxed{(x,y)}$ corresponding to user.
    Update $\theta$ using $\boxed{(x,y)}$. ~~$(x^{(i)}, y^{(i)})$~~
    → $\theta_j := \theta_j - \alpha(h_\theta(x) - y) \cdot x_j$    $(j = 0, \ldots, n)$
}
```

Can adopt to changing user preference.

Andrew



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. (x,y) ←

→ Learn $\boxed{p(y = 1|x;\theta)}$. ←   predicted 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; ...

映射约减 (map reduce) 方法

$\boxed{m = 400}$ $m = 400{,}000{,}000$

Map-reduce

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

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

Machine 1: Use $(x^{(1)}, y^{(1)}), \ldots, (x^{(100)}, y^{(100)})$.

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

Machine 2: Use $(x^{(101)}, y^{(101)}), \ldots, (x^{(200)}, y^{(200)})$.

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

Machine 3: Use $(x^{(201)}, y^{(201)}), \ldots, (x^{(300)}, y^{(300)})$.

$temp_j^{(3)} = \sum_{i=201}^{300} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$

Machine 4: Use $(x^{(301)}, y^{(301)}), \ldots, (x^{(400)}, y^{(400)})$.

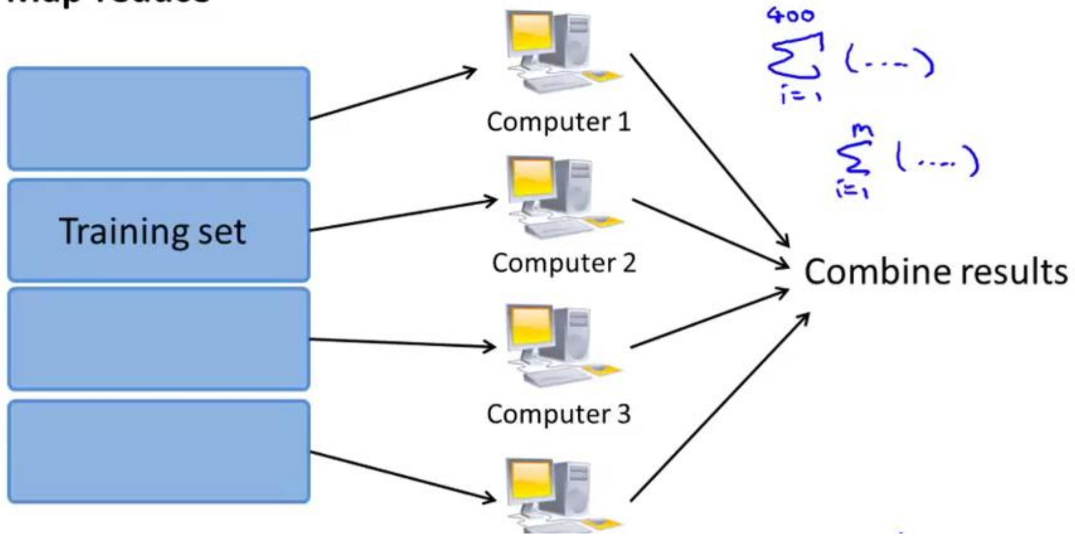
$temp_j^{(4)} = \sum_{i=301}^{400} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$

Combine:

$$\boxed{\theta_j := \theta_j - \alpha \frac{1}{400} \left(temp_j^{(1)} + temp_j^{(2)} + temp_j^{(3)} + temp_j^{(4)} \right)}$$

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

Map-reduce



$\sum_{i=1}^{400} (\ldots)$

$\sum_{i=1}^{m} (\ldots)$

Map-reduce and summation over the training set

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

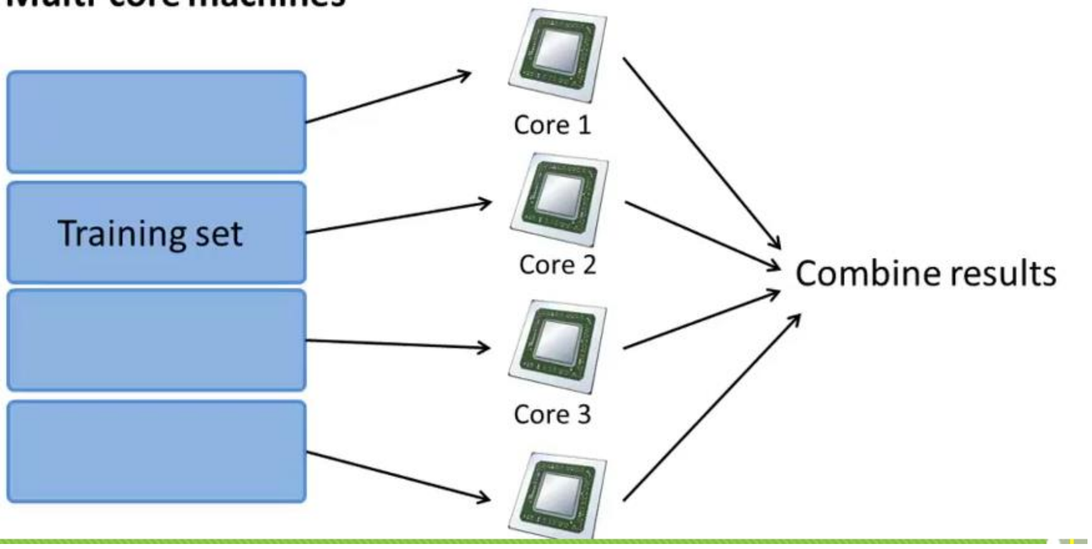
E.g. for advanced optimization, with logistic regression, need:

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

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

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

Multi-core machines


Training set
Core 1
Core 2
Core 3
Combine results