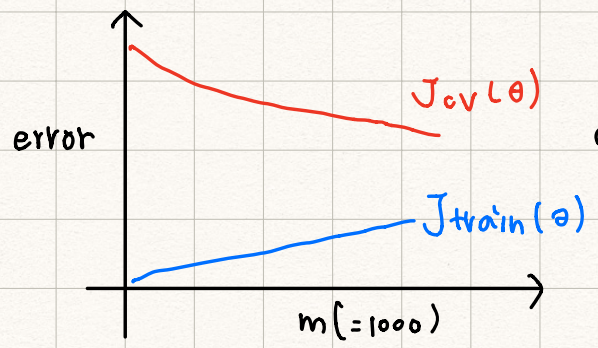
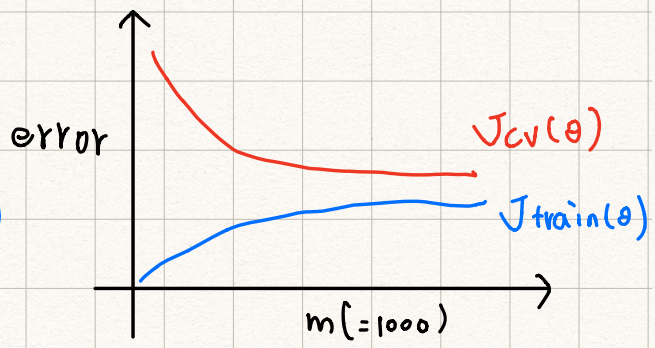
Gradient Descent with Large Datasets

Learning with Large Datasets

$m = 100000000$ ，先畫出 $m=1000$ 的 learning curve 藉此圖形觀察加大 data 是否有用？





Machine learning and data

見 week 6 最後的幾項敘述

Stochastic Gradient Descent (隨機梯度下降)

Stochastic GD

可以應用在任何以 GD 進行訓練的算法中

用來解決 GD 資料量太大所導致的計算量問題

Batch GD vs Stochastic GD

↗ = sequential mode

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

Repeat {

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

(for every $j=0\sim n$)

}

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

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

1. Randomly shuffle
2. Repeat

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

以上兩種GD的收斂方式是不同的

傾向一直線往最小值前進 | 靠近局部最小值處徘徊

Stochastic gradient descent

1. Randomly shuffle training examples
2. Repeat { for $i:=1,\ldots m$ {

$\theta_j := \theta_j - \alpha(h_\theta(x^{(i)})-y^{(i)})x_j^{(i)}$

For every $j=0,\ldots,n$

} }

1~10 次就收斂

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 通常介於 2 到 100 之間

Mini-batch gradient descent algorithm

Say b=10, m=1000

↑ 能做向量化，某些庫對向量計算有所優化故計算速度↑

Repeat {

for i = 1, 11, 21, 31, ..., 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 $j = 0, \dots n$)

}

}

Stochastic gradient descent convergence

確保此算法能收敛 and 選擇合適的學習率 α

Checking for convergence

Batch GD:

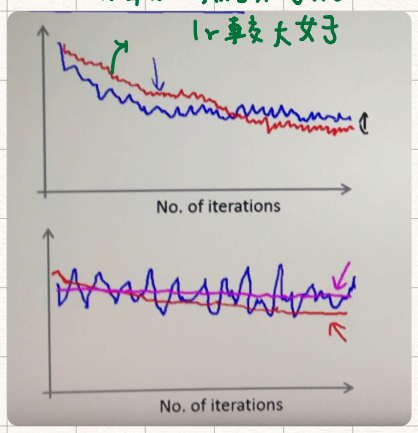
Plot $J_{train}(\theta)$ as a function of the number of iterations of gradient descent.

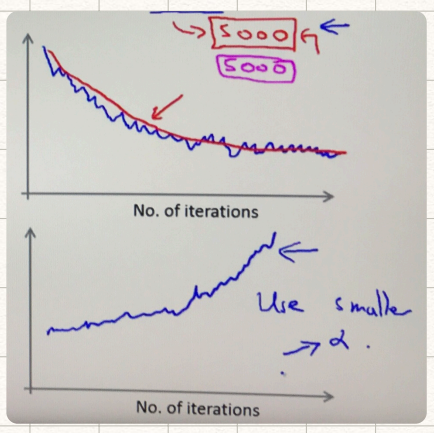
$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Stochastic GD:

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

在更新前計算 cost. every 1000 iterations 畫出 $cost(\theta, (x^{(i)}, y^{(i)}))$ 的平均值





Plot cost, averaged over the last 1000 examples

5000

Learning rate of Stochastic GD

Learning rate α is typically held constand.

將 α 設定成隨著迭代次數修改 α 值

公式： $\alpha = \frac{\text{const 1}}{\text{iterationNum} + \text{const 2}}$

but 還要設置兩個常數，工作量大

較少見人用

Advanced Topics

Online learning

對大批進入的 data 進行學習（在線學習）

- 特徵 x 是用戶的 properties
- learn $p(y=1 \mid x;\theta)$ to optimize price

  1表下單，0表不下單
- Repeat forever {

  Get (x,y) corresponding to user.

  Update θ using (x,y) ← data 訓練完便丟棄

  $\theta_j := \theta_j - \alpha(h_\theta(x)-y)\cdot x_j \quad (j=0\dots n)$

  }

此算法可以適應 user 的行為變化

Online learning example:

Product search (learning to search)

User searches for "Android phone 1080p camara"
Have 100 phones in store. Will return 10 results.

- x = features of phone. how many words in user query match .....
- $y = 1$ if user clicks on link. $y = 0$ otherwise

Learn $p(y = 1 \mid x; \theta)$. predict CTR (click through rate)

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

Other online learning examples:

- customized selection of news articles
- product recommendation
- choosing special offers to show user

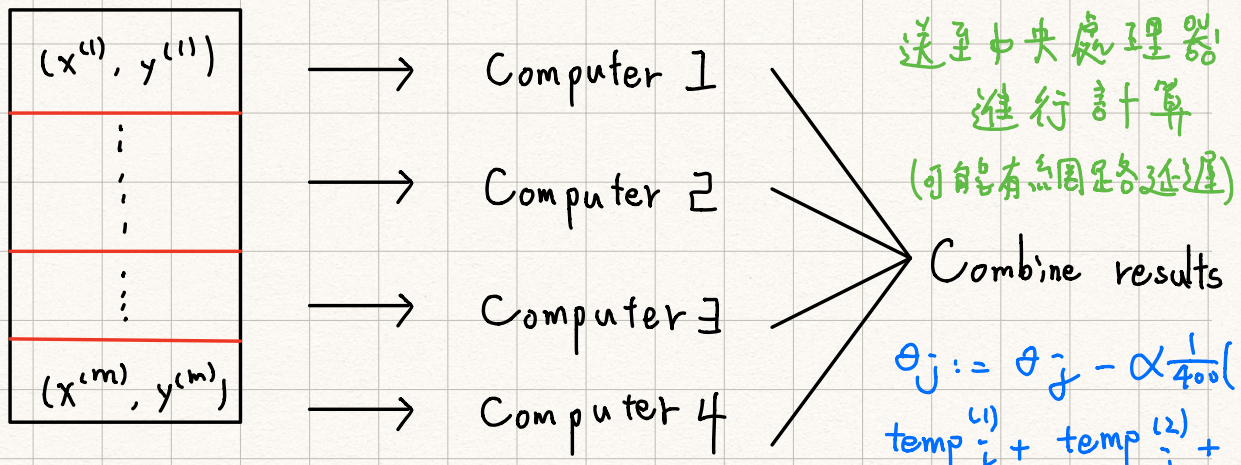
Map reduce and Data Parallelism

數據太大，而不能在一台計算機計算

多台 computer 一同計算

Map - reduce

to batch-gradient-descent algorithm



Training set

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

Map-reduce and summation over the training set

許多算法可以拆成多塊後再進行計算

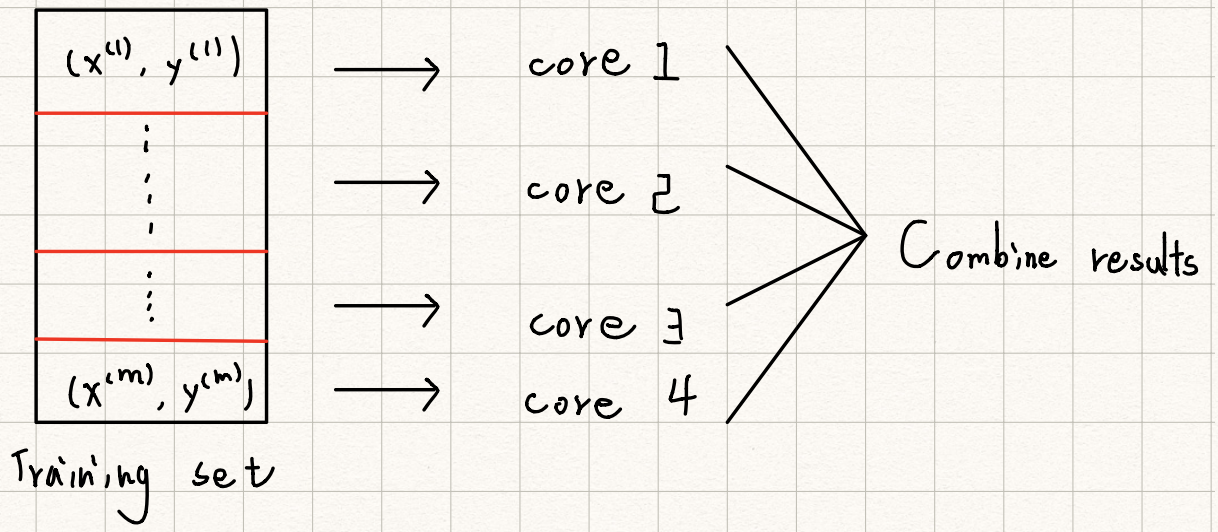
(logistic regression)

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

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

Multi - core machines

只有一台 pc 也能做 map - reduce (多核CPU)



如果 function 庫能自動實現多核計算
便不需 map - reduce