

# Recommender System

## collaborative filtering

- Given  $x^{(1)} \dots x^{(n_m)}$  and movie ratings.

can estimate  $\theta^{(1)}, \theta^{(2)} \dots \theta^{(n_u)}$

- Given  $\theta^{(1)}, \dots \theta^{(n_u)}$ . can estimate  $x^{(1)}, \dots x^{(n_m)}$

⇒ Guess  $\theta \rightarrow x \rightarrow \theta \rightarrow x \dots$

| Movie                | Problem motivation |         |           |          | $\downarrow$       | $\downarrow$      |
|----------------------|--------------------|---------|-----------|----------|--------------------|-------------------|
|                      | Alice (1)          | Bob (2) | Carol (3) | Dave (4) | $x_1$<br>(romance) | $x_2$<br>(action) |
| Love at last         | 5                  | 5       | 0         | 0        | 0.9                | 0                 |
| Romance forever      | 5                  | ?       | ?         | 0        | 1.0                | 0.01              |
| Cute puppies of love | ?                  | 4       | 0         | ?        | 0.99               | 0                 |
| Nonstop car chases   | 0                  | 0       | 5         | 4        | 0.1                | 1.0               |
| Swords vs. karate    | 0                  | 0       | 5         | ?        | 0                  | 0.9               |

Large Datasets | Large scale → computational expensive

## Stochastic Gradient Decent.

### Batch gradient descent

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

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  $j = 0, \dots, n$ )

}

$m = 300,000,000$

### 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. ↵

2. Repeat {

$$\left[ \begin{array}{l} \text{for } i=1, \dots, m \text{ } \{ \\ \quad \theta_j := \theta_j - \alpha \frac{1}{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)} \\ \quad \} \text{ (for } j=0, \dots, n \text{ )} \end{array} \right] \rightarrow \frac{\partial}{\partial \theta_j} cost(\theta, (x^{(i)}, y^{(i)}))$$

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

第一步是打乱数据 第二步是算法的关键 是关于某个单一的训练样本( $x(i), y(i)$ )来对参数进行更新

对于随即梯度下降来说，有以下说法：

1. 当训练集的个数 $m$ 很大的时候，随即梯度下降比梯度下降要快很多。
2. 对于损失函数 $J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$ 来说，梯度下降每次迭代都会减小，而随即梯度下降不一定减小，甚至可能增大。
3. 随即梯度下降可以用在多种模拟的优化中。
4. 在随机梯度下降之前，最好（必须）打乱训练集的顺序。

## Mini-batch Gradient Decent

### Mini-batch gradient descent

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

Repeat {

> for  $i = 1, 11, 21, 31, \dots, 991$  {

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

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

}

}

→  $b$  examples

→ 1 example

Vectorization

http://blog.csdn.net/pinisorry

$b = 10$

尤其是，小批量梯度下降可能要超过随机梯度下降，只有当你有一个良好的实现时，通过使用适当的向量化来计算余下的项。

# Stochastic Gradient Decent Convergence

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



- 因为这些数字平均只有一千个例子，所以它们会有点吵，所以每一次迭代都不会减少。
- 图2红线：通过平均5000个例子，而不是1,000个，你可能会得到更加平滑的曲线。
- 图3平均数量较多的例子，我们在这里取平均值超过5000个例子，可能得到两种较平缓的曲线。如果得到的是洋红色水平线，你需要改变学习率或改变特征 改变算法的其他内容。
- 图4如果你看到一条曲线正在增加，那么这是一个信号，表明算法是发散的。你真正应该做的就是把学习速率 $\alpha$ 的值作为一个微不足道的值。降低学习速率 $\alpha$ 意味着每一次随机梯度下降的迭代将会采取一个较小的步骤，因此它可能会聚，而不是分歧。

总结：因此，如果曲线看起来过于嘈杂，或者如果它太多摆动，那么尝试增加你平均的例子的数量，这样你可以更好地看到情节的整体趋势。如果你看到错误是实际上增加，成本实际上是增加，尝试使用较小的alpha值。

## Stochastic gradient descent

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

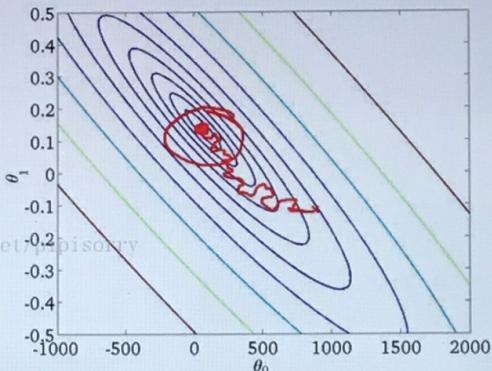
$$J_{\text{train}}(\theta) = \frac{1}{2m} \sum_{i=1}^m \text{cost}(\theta, (x^{(i)}, y^{(i)}))$$

1. Randomly shuffle dataset.

2. Repeat {

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

}



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

如果你想要随机梯度下降实际收敛到全局最小值，那么你可以做的一件事是你可以慢慢地降低 $\alpha$ 的学习速率。迭代次数是你运行的随机梯度下降的迭代次数，所以这真的是你见过的培训例子的数量。

# Advanced Topics of Large Scale Dataset.

- Online Learning

什么情况下使用在线学习？如果你运行一个主要的网站有一个连续的用户流，在线学习算法是非常合理的。因为数据本质上是免费的，如果你有这么多的数据，那么数据本质上是无限的当然，如果我们只有少量的用户，而不是使用在线学习算法，那么最好将所有的数据保存在一个固定的训练集中，然后运行一些算法。但是如果你真的有一个连续的数据流，那么在线学习算法可以是非常有效的。

这种在线学习算法的一个有趣效果就是，它能够适应不断变化的用户偏好。尤其是，如果随着时间的推移，由于经济的变化，用户的价格敏感度会降低，他们愿意支付更高的价格。如果你开始有新类型的用户来到你的网站。这种在线学习算法也可以适应不断变化的用户喜好，并跟踪你不断变化的用户愿意支付的种类。而且，因为如果你的用户池发生了变化，那么这些更新到你的参数theta将只是适应你的参数，

在线学习的优势：

1. 可以适应用户的品味变化。
2. 允许我们对流数据进行学习。

- Map-reduce & data parallelism

# Photo OCR (optical character recognition)

- **OCR pipeline**



- **Sliding window**

- **Getting lots of data & artificial data**

### Discussion on getting more data

1. Make sure you have a low bias classifier before expending the effort. (Plot learning curves). E.g. keep increasing the number of features/number of hidden units in neural network until you have a low bias classifier.
2. "How much work would it be to get 10x as much data as we currently have?"
  - Artificial data synthesis
  - Collect/label it yourself
  - "Crowd source" (E.g. Amazon Mechanical Turk)

hours?  $n = 1,000 \rightarrow 10 \text{ secs/example}$   $n = 10,000$

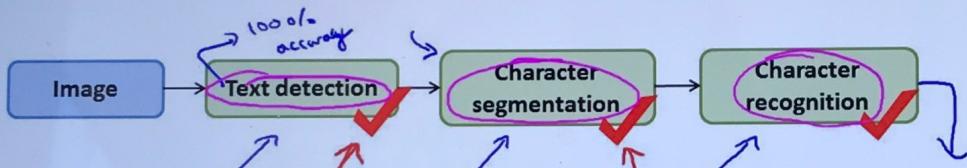
如果您面临机器学习问题，通常值得做两件事情：

其中一个就是头脑清楚，通过学习曲线，可以得到更多的数据。

其次，假设情况如此，请问：要获得十倍的数据将需要多少时间，但有时候，您可能会感到惊讶，原因可能是几天，几周甚至几天，这可以是一个很好的方式来给你的学习算法在性能上有巨大的提升。

- **Ceiling Analysis**

### Estimating the errors due to each component (ceiling analysis)



What part of the pipeline should you spend the most time trying to improve?  
<http://blog.csdn.net/pipisorry>

| Component              | Accuracy                                  |
|------------------------|---|
| Overall system         | 72%                                       |
| → Text detection       | 89% $\downarrow 17\%$                     |
| Character segmentation | 90% $\downarrow 1\%$<br>$\downarrow 10\%$ |

首先写出总的项目精度为72%；然后手工把某一个模块设置为“全对”，看模型提升了多少，提升的多说明这个工作有用；提升的少，说明在这里做工作没什么太大意义。

## Summary: Main topics

$$(x^{(i)}, y^{(i)})$$

### → Supervised Learning

- Linear regression, logistic regression, neural networks, SVMs

$$x^{(i)}$$

### → Unsupervised Learning

- K-means, PCA, Anomaly detection

### → Special applications/special topics

- Recommender systems, large scale machine learning.

### → Advice on building a machine learning system

- Bias/variance, regularization; deciding what to work on next: evaluation of learning algorithms, learning curves, error analysis, ceiling analysis.

