Large Scale ML 大规模机器学习

Tips：通过绘制 learning curves 来判断欠拟合/过拟合，从而修改算法或增减数据。

Part 1：大数据集下的梯度下降

Stochastic gradient decent 随机梯度下降

与普通的梯度下降（batch gradient decent，批量梯度下降）对比：

1. 批量梯度下降每走一步都要遍历全体数据集中的 m 个样本（计算总梯度），而随机梯度下降每一步仅参考单个样本，遍历的任务通过循环来做，运算速度快；
2. 批量梯度下降走“直线”，随机梯度下降走很曲折的线路；
3. 批量梯度下降最后收敛到一个点，随机梯度下降最后收敛在一个范围。

算法：

Step1 打乱数据

Step2 参数更新——

循环执行下列内容 1-10 次：

内部循环：从 1-m 逐步遍历数据集，每一步中使用该单一样本(x_i, y_i)对参数进行更新。

- 本算法中 cost function 变成针对单一训练样本的。

Mini-batch gradient decent 小批量梯度下降

介于 batch 梯度下降和 stochastic 梯度下降之间，每步更新参数时使用 b 个样本（而不是 m 个或 1 个）。

b 的一般取值范围在 2-100，常取 10。

Stochastic gradient decent convergence 敛散性

如何监测随机梯度下降算法在最优化代价函数时的表现，即判断其是否收敛？

每次更新参数前，先算出针对该样本的 cost。然后每迭代若干次，譬如 1000 次，计算此前处理的 1000 个样本的 cost 之平均值，绘制图像（cost – number of iterations），看是否下降。

如果不下降，把 1000 增大到 5000 试试，点会变稀疏（取样频率低了）但曲线会变平滑，这时如果仍然不下降，则说明学习速率α取大了，或者算法中 features 等选取不合适需进行修改。

如果希望最终数据更精细地收敛到全局最小点而不是在附近振荡，可以随着迭代次数增加而减小α，如$\alpha = \frac{常数1}{迭代次数+常数2}$。不过一般还是取常数。

Part 2：在线学习——从数据流中学习

与前面拥有一个固定数据集不同，从流动的数据中学习。

每当新样本进来，使用它对参数进行更新，然后丢弃数据。

好处：automatically adapt to changes.

应用：邮费问题；预测点击率问题（产品搜索推荐）。

$$p(y = 1|x; \theta)$$

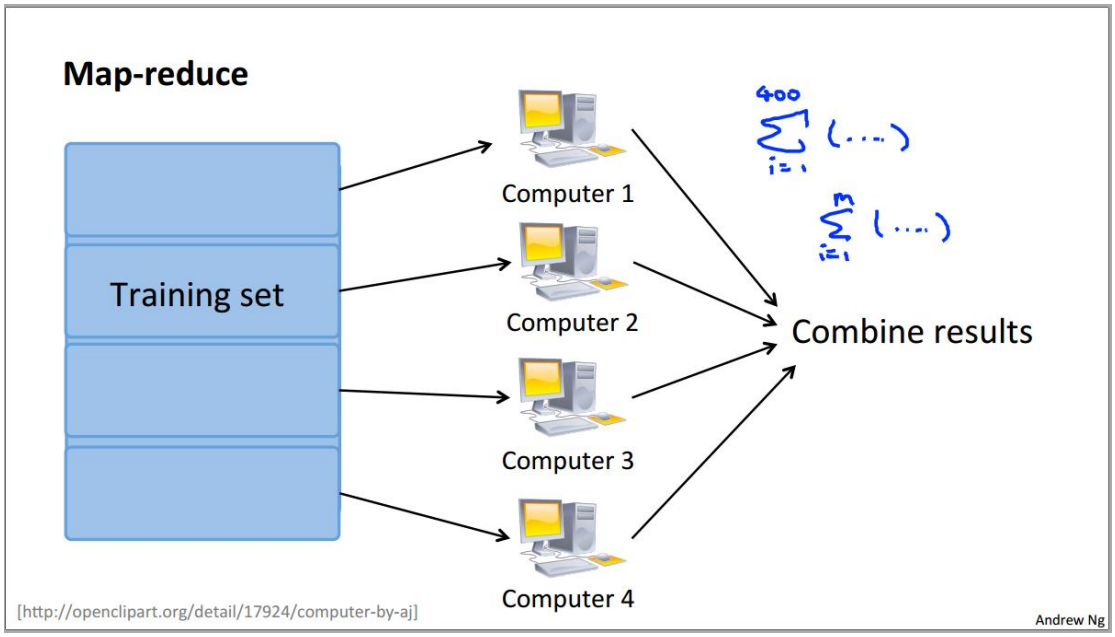
Part 3：映射化简（多核处理）Map-reduce

对于计算过程中一些计算量很大的步骤可以用求和形式表达的算法，

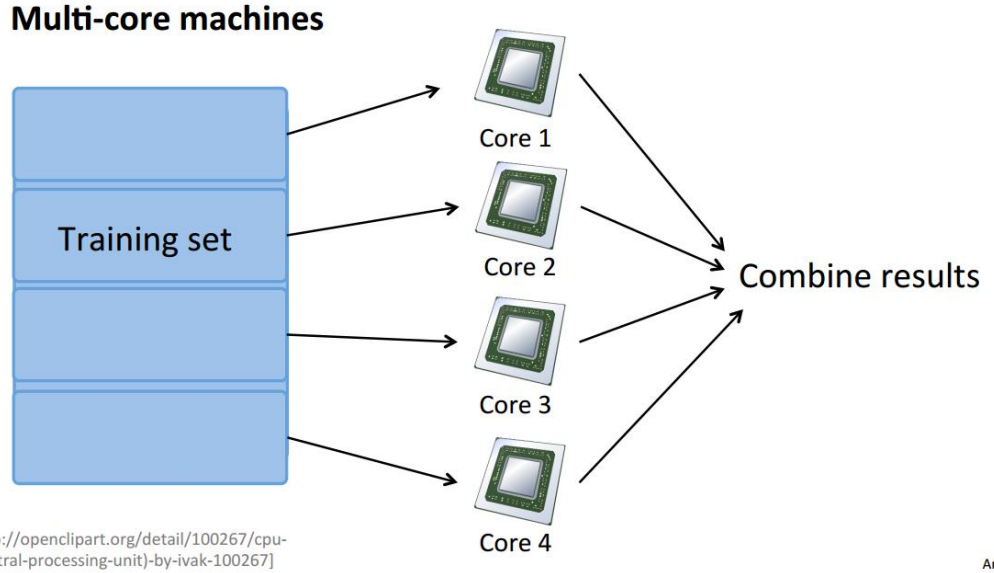
可以使用 map-reduce 方法，即：

将数据集分成若干份，在多个计算机上并行计算再汇总，或：

将数据集分成若干份，在一个计算机的多个核上并行计算。



Multi-core machines



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

Andrew Ng

Map-reduce

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

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

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 $(x^{(101)}, y^{(101)}), \ldots, (x^{(200)}, y^{(200)})$.

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

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:

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

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

[Jeffrey Dean and Sanjay Ghemawat]

Andrew Ng