Online Learning:

Jun 1, 2021

e.g. Shipping service website

x includes user location, package destination, price

$y = 1$ if user chose to use, $y = 0$ otherwise

We want to learn $p(y = 1 | x; \theta)$ to optimize price.

Repeat forever {

Get (x, y) corresponding to user.

Update θ using (x, y):

$$\theta_j := \theta_j - \alpha (h_\theta(x) - y) x_j \quad \text{for } j = 0 \ldots n$$

}

* Algorithm adapts to changing user preference.

e.g. Product search

User search: "Android phone 1080p camera"

100 phones total, return 10 results.

x includes features of phone, # words in user query that match the phone's name, # words in query that match description of phone, etc

$y = 1$ if user clicks on link, $y = 0$ otherwise

Learn predicted click-through rate (CTR).

↳ $p(y=1 \mid x; \theta)$

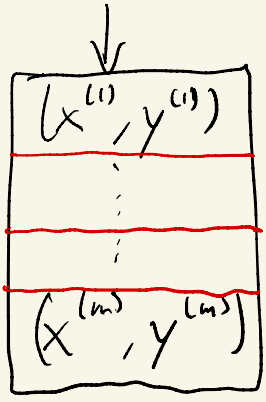
※ Use same logistic regression algorithm.

Map-Reduce:

Batch gradient descent:

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

$m = 400$



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

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

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

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

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

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

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

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

※ Data Parallelism

Combine!

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

for $j = 0 \ldots n$

However, due to network latency and other overheads, map-reduce will not translate directly to an N-fold speedup, where N is the # of computers.