# Online Learning & Map Reduce

Online learning is useful when we have a new large-scale machine learning that continuous stream of data is coming in and we would like to write an algorithm for it. Suppose we have a shipping service website, where user specifies the origin and destination of where they want their package ship and the website gives offers to choose. User sometimes choose the shipping service (y=1), sometimes not (y=0). We want to learn what's the probability that they will elect to ship the package, using our shipping service. Learn p (y=1|x; $\theta$) to optimize the price. X has the price we are asking for and properties of user (origin and destinations). Logistic regression or neural network is best to use for this system. Here is what online learning does:

```
Repeat forever (continuous run) {
        Get (x, y) corresponding to user
        Update θ using (x, y):
                θ_j := θ_j − α(h_θ(x) − y).x_j for j=0…, n
}
```

In this algorithm, we learn from data (x, y) and then we throw that away and never use that example again. This is great for huge countinous data streamimg. *It can be adapt to user preference*. For example, if user becomes more price sensitive, the model adapts. It's because the update changes as user preferences changes by time.

Another example is product search like "Android phone with 1080p camera". We have 100 phones in our store and we like to return like 10 phones with that specification user searched. For this example, x is features of the phone (how many words in query matches the description) and y=1 if user click on a link, otherwise 1. We like to learn to learn the estimate of p (y=1|x; $\theta$). This is called predicted click through rate (CTR). Then we can use to show the 10 phones they are most likely to click on. In this case, we get 10 training set (x, y). We can use online leaning algorithm to update the parameters using essentially 10 steps of gradient descent or 10 examples.

Other examples are choosing special offers to show, customize selection of new articles, product recommendations, and etc. If we have collaborative system, we have extra to feed into logistic regression.

**Online algorithm is great for those system that are changing slowly by time.**

## Map Reduce and Data Parallelism

Sometimes, we need several computers to run the algorithm on data. Suppose we have batch gradient descent for our data: $\theta_j := \theta_j - \alpha \frac{1}{400} \sum_{i=1}^{400} (h_\theta(x^i) - y^i) x_j^i$. Assume m=400 (in real life we have much bigger data like 400,000,000). In map-reduce, training set is divided into several machines, imagine 4 here:

Machine 1: use $(x^i, y^i)$ for i=1 till 100 and calculated the batch gradient descent (temp1)
Machine 2: use $(x^i, y^i)$ for i=1001 till 200 and calculated the batch gradient descent (temp2)
Machine 3: use $(x^i, y^i)$ for i=2001 till 300 and calculated the batch gradient descent (temp3)
Machine 4: use $(x^i, y^i)$ for i=301 till 400 and calculated the batch gradient descent (temp4)

At the end, the results are sent to master server and combine the results and update the parameter $\theta_j$: $\theta_j := \theta_j - \alpha \frac{1}{400}(temp1_j + temp2_j + temp3_j + temp4_j)$. In practice, we get speed of less than 4x because of overheads and latency. Many algorithms can be expressed as computing sum of functions over training set.

*In summary, map-reduce is used for parallelization. parallelization can come from multiple machines, CPUs, cores in each CPU. Therefore, even a single computer can do parallelization*. Parallelization over CPUs won't have any latency since it's over one single machine.