Online Learning

Online Learning allows us to model problems where we have a continuous stream of data we want an algorithm to learn from. Web companies use various types of online learning algorithms to learn from traffic.

- Can (for example) learn about user preferences and hence optimize our website

Online Learning example: Shipping service website

Shipping service website where user comes, specifies origin and destination, we offer to ship their package for some asking price, and users sometimes choose to use our shipping service ($y = 1$), sometimes not ($y = 0$). So let's say that we want a learning algorithm to help us to optimize what is the asking price that we want to offer to our users.

Specifically features x capture properties of user, of origin/destination and asking price. We want to learn $p(y = 1|x; \theta)$ to optimize price (what the probability of a user selecting the service is).

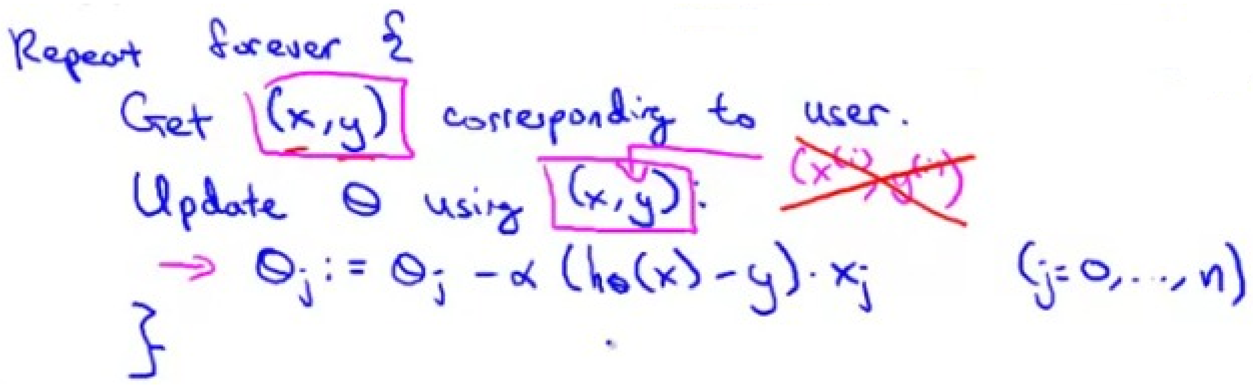
- $p(y = 1|x; \theta)$, probability that $y = 1$, given x, parameterized by θ
- So we can learn this property of $y = 1$, given any price and given the other features we could really use this to choose appropriate prices as new users come to us.

So in order to model the probability of $y = 1$, what we can do is use:

- Logistic regression
- Neural network
- Or some other algorithm.

If we have a website that runs continuously an online learning algorithm would do something like this:

- Get (x, y) corresponding to user. Where:
  - x - feature vector including price we offer, origin, destination
  - y - if they chose to use our service or not
  - Once we get the (x, y) pair, what an online learning algorithm does is:
    - **update the parameters θ using just the example (x, y)**
  - So we basically update all the θ parameters every time we get some new data


Repeat forever {
Get (x,y) corresponding to user.
~~(x^(i), y^(i))~~
Update θ using (x,y).
→ θ_j := θ_j − α (h_θ(x) − y) · x_j   (j=0,...,n)
}

While in previous examples we might have described the data example as $(x^{(i)}, y^{(i)})$, in this online learning setting where actually discarding the notion of there being a fixed training set, instead we have a **continuous stream of data** so indexing is largely irrelevant as **we're not storing the data**.

Now what happens is: we get an example and then **we learn using that example** and then we **throw that example away** and so that's why we just look at one example at a time. We're also doing away with this notion of there being this sort of fixed training set indexed by i.

If we have a major website where we have a **massive stream of data** then this kind of algorithm is pretty reasonable:

- We're free of the need to deal with all our training data
- If we had a **small number of users** we could save their data and then run a **normal algorithm** on a dataset.

An online algorithm can adapt to changing user preferences:

- So over time users may become more price sensitive
- The algorithm adapts and learns to this
- So **our system is dynamic**

Other online learning example - Application in product search

Product search (learning to search)

- User searches for "Android phone 1080p camera"
  - The user type a query like: "Android phone 1080p camera"
- Have 100 phones in store. Will return 10 results
  - We want to offer the user 10 phones per query

What we'd like to do is have a learning algorithm help us figure out what are the ten phones out of the 100 we should return the user in response to a user-search.

How do we do this:

What we like to do is estimate the probability that a user will click on the link for a specific phone, because we want to show the user phones that they are likely to want to buy, want to show the user phones that they have high probability of clicking on in the web browser.

- For each phone and given a specific user query, we create a feature vector (x):
  - **x = features of phone, how many words in user query match name of phone, how many words in query match description of phone, etc.**
    - Basically how well does the phone match the user query
  - We want to estimate the probability of a user selecting a phone, so define:
  - **y = 1 if user clicks on link. y = 0 otherwise.**
  - **We want to learn, $p(y$ = 1$|x;\theta)$ --> this is the problem of learning the predicted click through rate (CTR)**
    - CTR just means learning the probability that the user will click on the specific link that we offer them,
  - If we can estimate the CTR for any phone we can use this to show the highest probability phones first:
  - **Use to show user the 10 phones they're most likely to click on.**
    - i.e. user can click through one or more, or none of them, which defines how well the prediction performed
  - **Other examples: Choosing special offers to show user; customized selection of new articles; product recommendation;**$\ldots$

Video Question: Some of the advantages of using an online learning algorithm are:

> It can adapt to changing user tastes (i.e., if $p(y|x;\theta)$ changes over time).

- There is no need to pick a learning rate α.

> It allows us to learn from a continuous stream of data, since we use each example once then no longer need to process it again.

- It does not require that good features be chosen for the learning task.

Concretely: With a continuous stream of users to a website, we can run an endless loop that gets (x, y), where we collect some user actions for the features in x to predict some behavior y.

We can update θ for each individual (x, y) pair as we collect them. This way, we can adapt to new pools of users, since we are continuously updating θ.