Learning With Large Datasets

This set of notes look at large scale machine learning - **how do we deal with big datasets?**

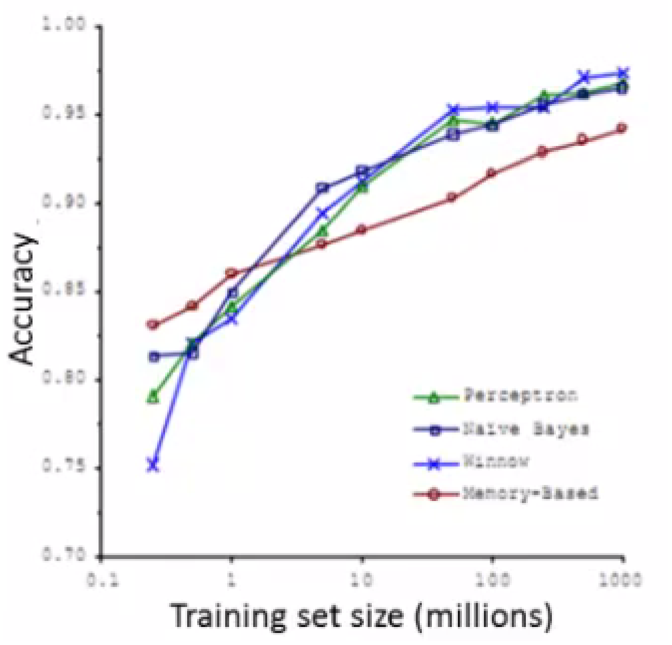
If we look back at a recent 5 or 10 years history of machine learning. One of the reasons that learning algorithms work so much better now than even say, 5-years ago, is just the sheer amount of data that we have now and that we can train our algorithms on.

Machine Learning and data

We've already seen that one of the best ways to get a high performance machine learning system, is if we take a low-bias learning algorithm, and train that on a lot of data.

- E.g. Classify between confusable words.
  - Confusable words: {to, two, too}, {the, than}.
  - E.g. For breakfast I ate ***two*** eggs

We saw in this example, these sorts of results, where so long as we feed the algorithm a lot of data, it seems to do very well.



And so it's results like these that has let to the saying in machine learning that often:

- **"It's not who has the best algorithm that wins. It's who has the most data."**

Learning with large datasets

So it's good to learn with large datasets

- But learning with large datasets comes with its own computational problems

For example, say we have a data set where m = 100,000,000

- **This is pretty realistic for many datasets:**
  - Census data (where m is about 300,000,000 people in USA), we can usually get hundreds of millions of records.
  - Website traffic data - If we look at the amount of traffic that popular websites get, we easily get training sets that are much larger than hundreds of millions of examples.

Let's say we want to train a linear regression model, or maybe a logistic regression model:

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

So you have to sum over 100,000,000 terms per step of gradient descent

- Because of the computational cost of this massive summation, we'll look at more efficient ways around this later.

First thing to do is ask if we can train on 1,000 examples instead of 100,000,000

- Randomly pick a small selection
- Can we develop a system which performs as well?

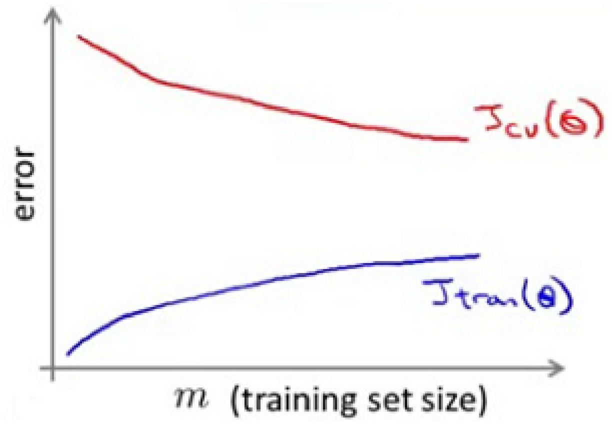
Video Question: Suppose you are facing a supervised learning problem and have a very large dataset (m = 100,000,000). How can you tell if using all of the data is likely to perform much better than using a small subset of the data (say m = 1,000)?

- There is no need to verify this; using a larger dataset always gives much better performance.
- Plot $J_{\text{train}}(\theta)$ as a function of the number of iterations of the optimization algorithm (such as gradient descent).
- Plot a learning curve ($J_{\text{train}}(\theta)$ and $J_{\text{CV}}(\theta)$, plotted as a function of m) for some range of values of m (say up to m = 1,000) and verify that the algorithm has bias when m is small.

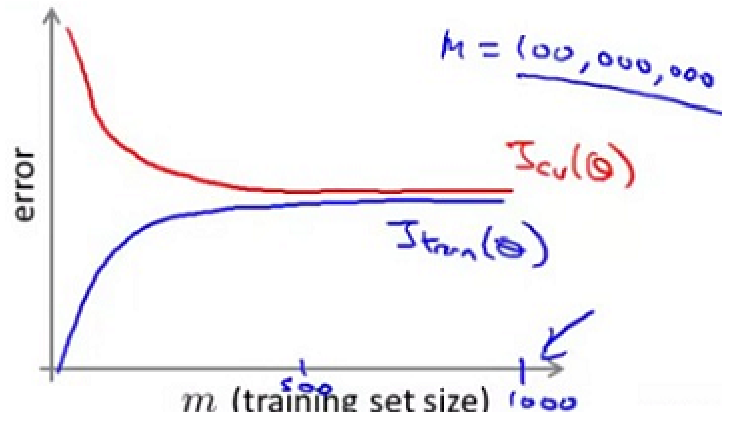
> Plot a learning curve for a range of values of m and verify that the algorithm has high variance when m is small.

To see if taking a smaller sample works, we can sanity check by **plotting error vs. training set size (m)**

- If our plot looked like this:



- Looks like a **high variance problem**:
  - More examples should improve performance *If plot looked like this:



- This looks like a high bias problem
  - More examples may not actually help - save a lot of time and effort if we know this before hand
  - One natural thing to do here might be to;
    - Add extra features
    - Add extra hidden units (if using neural networks)

We mainly benefit from a very large dataset when our algorithm has high variance when m is small. Recall that if our algorithm has high bias, more data will not have any benefit.

Datasets can often approach such sizes as m = 100,000,000. In this case, our gradient descent step will have to make a summation over all one hundred million examples.