Overfitting Neural Networks

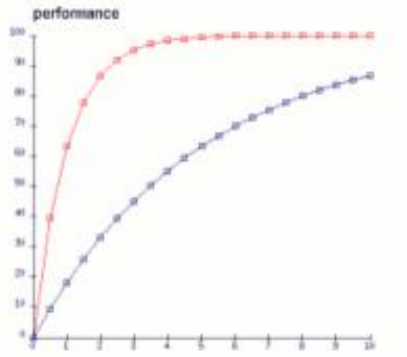
Steep Learning Curves and other erroneous metaphors

Interesting peculiarities of American English never cease to amuse non-native speakers like me. Many languages are full of idiotic idioms, but I find two idioms widely used in Corporate America particularly amusing because they reveal major misconceptions about simple mathematical concepts.

Have you ever wondered where the expression "steep learning curve" comes from? The first time you heard it was most likely from a suit-wearing business person talking about a new and intimidating piece of software he/she knew nothing about, as in: *I heard this Linux thing is cool but has a steep learning curve*. What he/she meant by that was that learning to use this software was a long and arduous process.

Now folks, hear me loud and clear: **steep learning curves are your friends**. Take my word for it, I am a machine learning guy, I know what I'm talking about.

Take the picture on the right. These are what we, learning folks, call learning curves. They represent skill, accuracy, or performance at a particular task (say as percentage of success) as a function of time. Here, the red curve corresponds to quick learning, and the blue curve to slower learning. Oddly enough, the steep curve is the red curve, the one that corresponds to a quickly learned task.



Behind the expression "steep learning curve" lies an everyday metaphor erroneously applied to a mathematical concept. The metaphor is that the poor learner must somehow "climb" the learning curve. So, steeper must be harder. But it's not like that at all. A Learning curve are more like a mountain with a ski lift: you get pulled at constant horizontal speed. The steeper it is, the sooner you get to the top.

So next time a suit tells you about a technology with a "steep learning curve", you can reply "that means even *you* could learn it quickly".

http://yann.lecun.com/ex/fun/index.html#steep

CSC411: Machine Learning and Data Mining, Winter 2017

Michael Guerzhoy

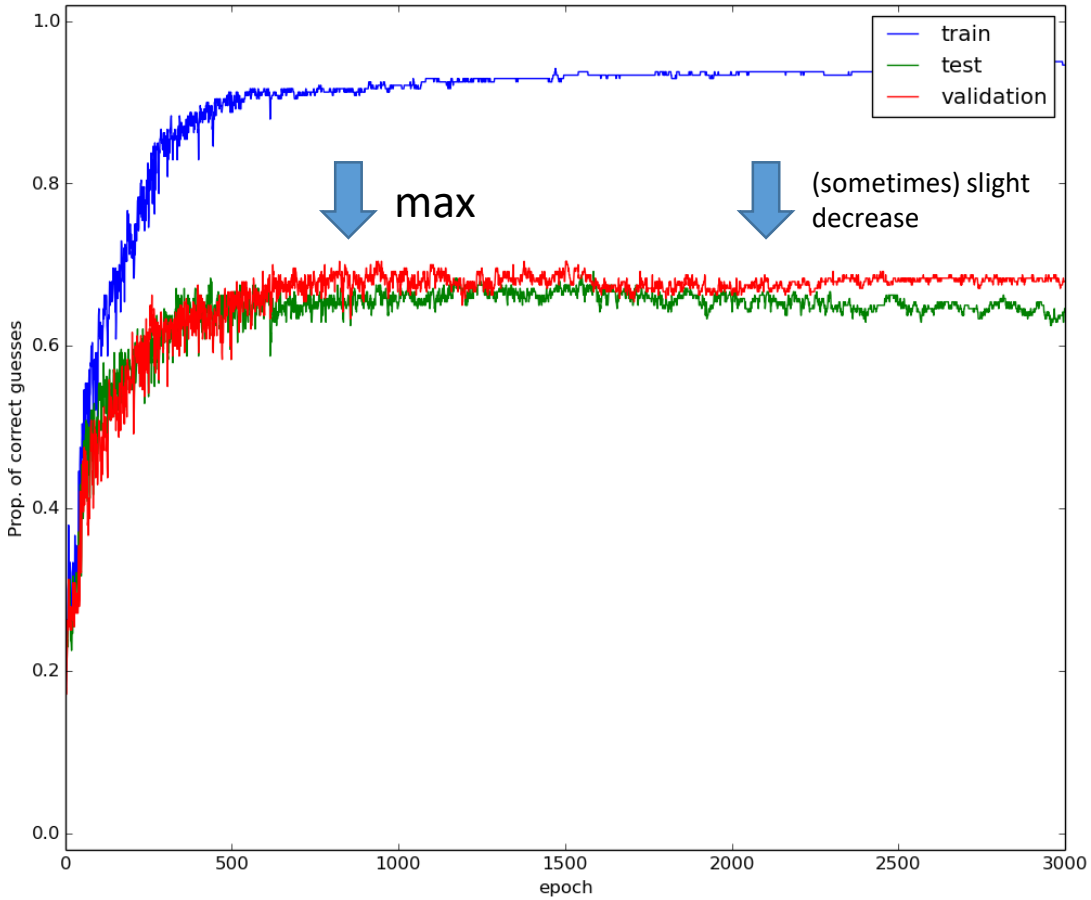
Some slides from Geoffrey Hinton

Learning Curves

- Split the data into a training set, validation set, and test set.
- Minimize the cost function on the training set, measure performance on all sets
- Plot the performance on the three sets vs. the number of optimization iterations performed
  - Optimization iteration i:

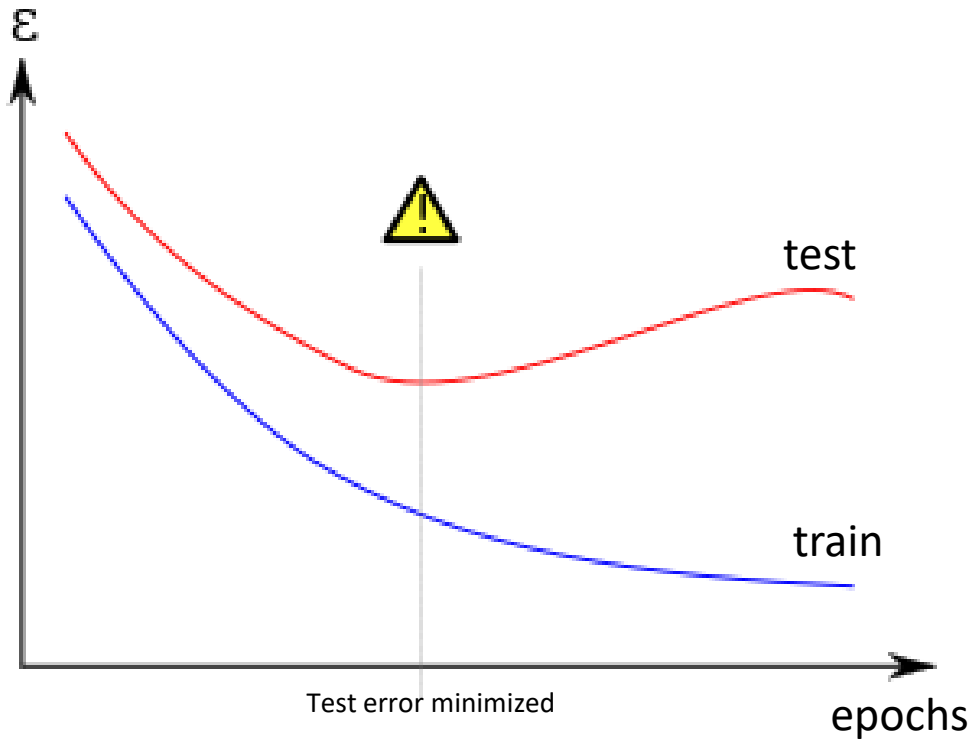
$$\theta_{i+1} \leftarrow \theta_i - \alpha \nabla cost(\theta, x_{train}, y_{train})$$

“Typical” Learning Curves



300-unit hidden layer. 6 people, 80 examples each. Best test performance: 68%

Wikipedia version



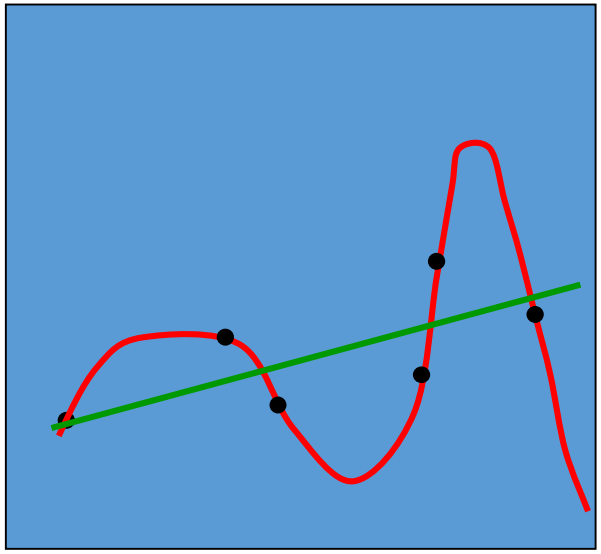
(Basically a fairytale: the moral of the story is true, but things rarely look this nice)

Learning Curves

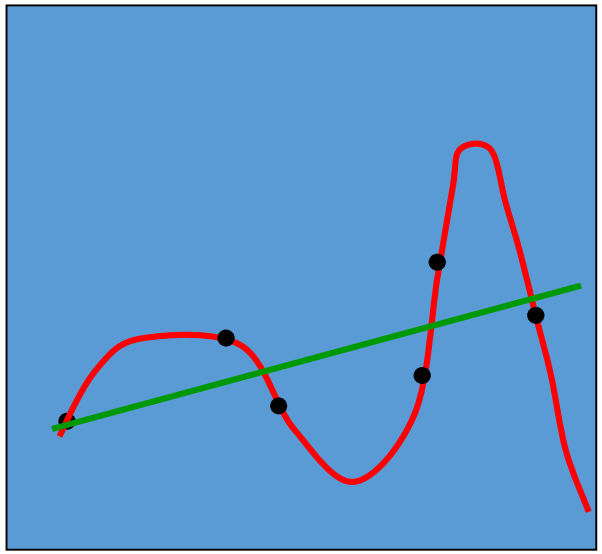
- The training performance always increases
  - (Assuming the performance is closely enough related to the cost we're optimizing – we sometimes also plot the cost directly)
- The test and validation curve should be the same, up to sampling error (i.e., variation due to the fact that the sets are small and sometimes things work better on one of them by chance)
- The training and validation performance sometimes initially decreases and the decreases as we minimize the cost function

Overfitting

- Overfitting happens when the model (e.g., the Neural Network) models the specific training set rather than the underlying data from which the training set is taken
  - I.e., because the training set is too small, the network can do extremely well on the training set by modelling its peculiarities

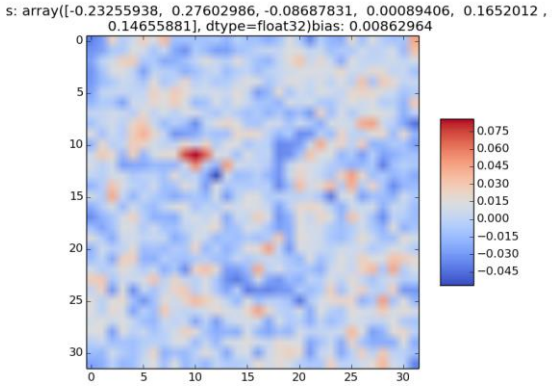


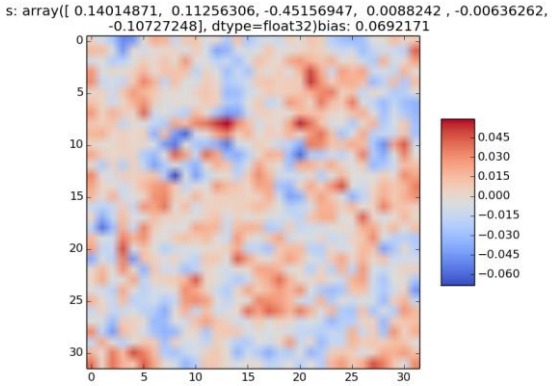
A Simple Example of Overfitting



- Which model do you believe?
  - The complicated model fits the data better.
  - But it is not economical
- A model is convincing when it fits a lot of data surprisingly well.
  - It is not surprising that a complicated model can fit a small amount of data.

Overfitting and Faces





- Above you see examples of W^0 that give near-100% performance on the training set
- The random spots you see are random regularities in the small training set being exploited – exploiting them on the test set won't work, and will possibly lead to bad performance

300 Hidden Units, 3000 epochs, no regularization, 40 example per actor, 6 actors

How Overfitting Faces might Work

- Have each hidden neuron “memorize” one example face
  - Then have the output neurons light up for the hidden units that memorized the corresponding actor
- This happens to some extent if you use more sophisticated optimization methods than what I’d used