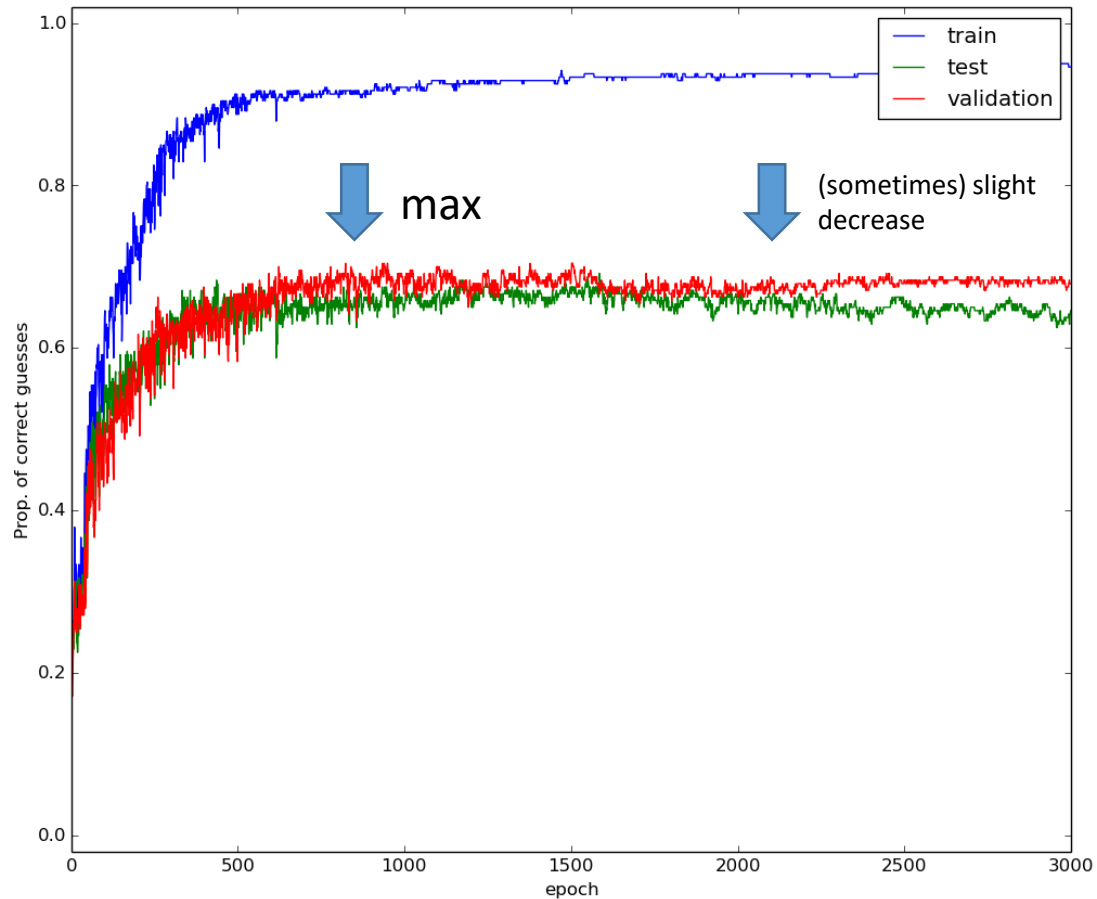


# Learning Curves

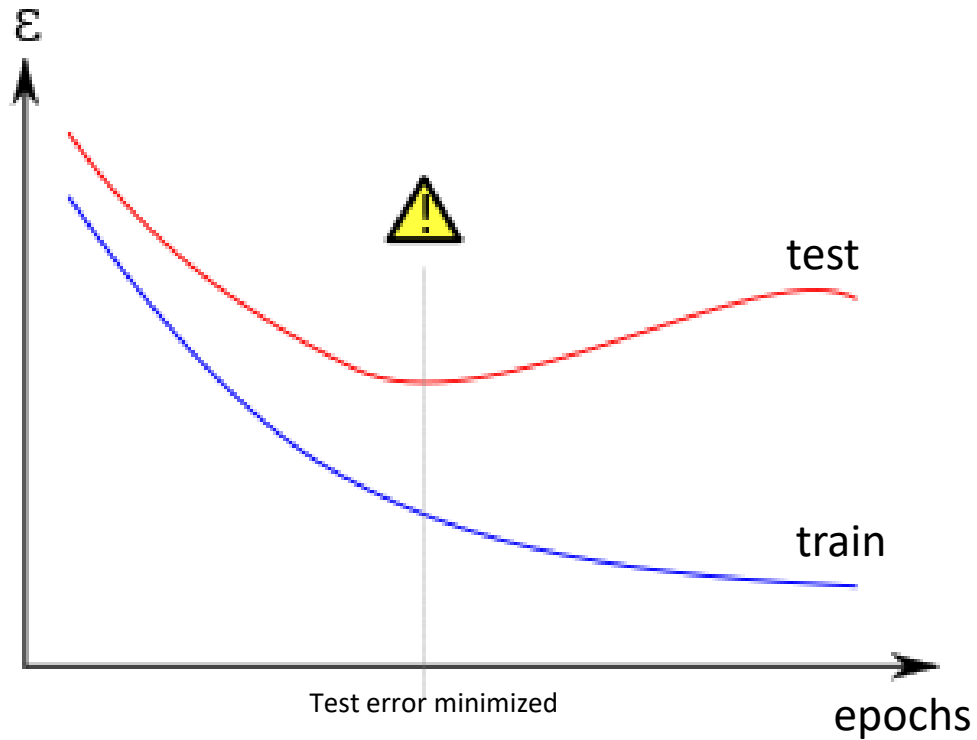
- Split the data into a training set, validation set, and test set.
- Minimize the cost function on the training set, measure performance on all sets
- Plot the performance on the three sets vs. the number of optimization iterations performed
  - Optimization iteration  $i$ :  
$$\theta_{i+1} \leftarrow \theta_i - \alpha \nabla \text{cost}(\theta, x_{\text{train}}, y_{\text{train}})$$

# “Typical” Learning Curves



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

# Wikipedia version



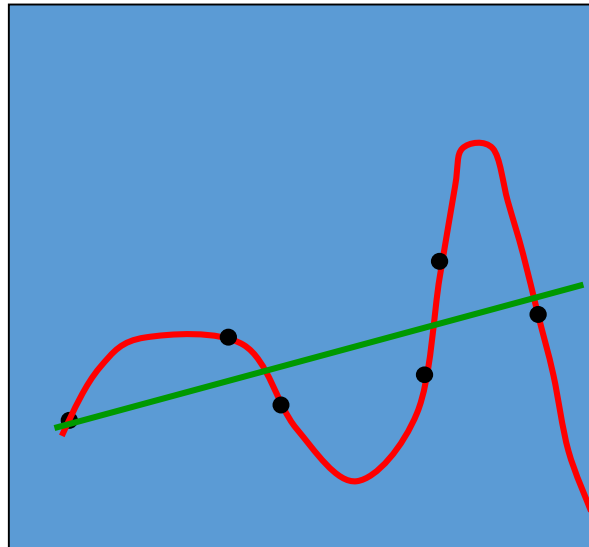
(Basically a fairytale: the moral of the story is true, but things rarely look this nice)

# Learning Curves

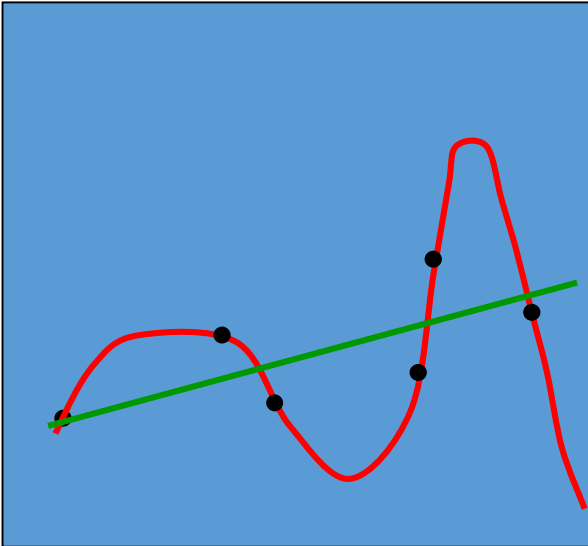
- The training performance always increases
  - (Assuming the performance is closely enough related to the cost we're optimizing – we sometimes also plot the cost directly)
- The test and validation curve should be the same, up to sampling error (i.e., variation due to the fact that the sets are small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



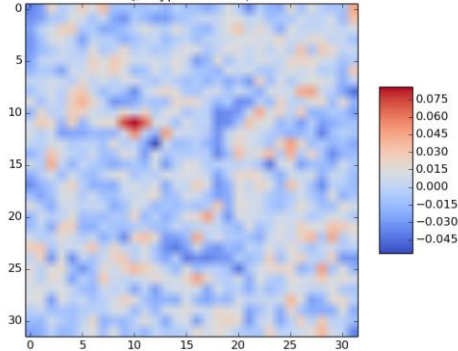
# A Simple Example of Overfitting



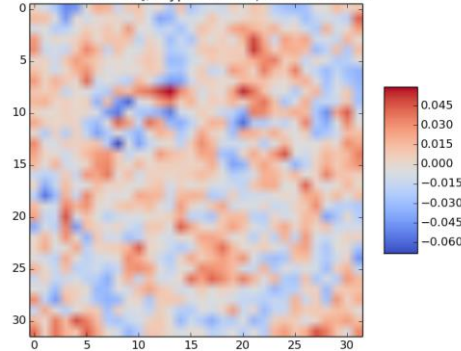
- Which model do you believe?
  - The complicated model fits the data better.
  - But it is not economical
- A model is convincing when it fits a lot of data surprisingly well.
  - It is not surprising that a complicated model can fit a small amount of data.

# Overfitting and Faces

s: array([-0.23255938, 0.27602986, -0.08687831, 0.00089406, 0.1652012, 0.14655881], dtype=float32) bias: 0.00862964



s: array([ 0.14014871, 0.11256306, -0.45156947, 0.0088242, -0.00636262, -0.10727248], dtype=float32) bias: 0.0692171



- 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

# 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