Beyond Backpropagation

3/22/17

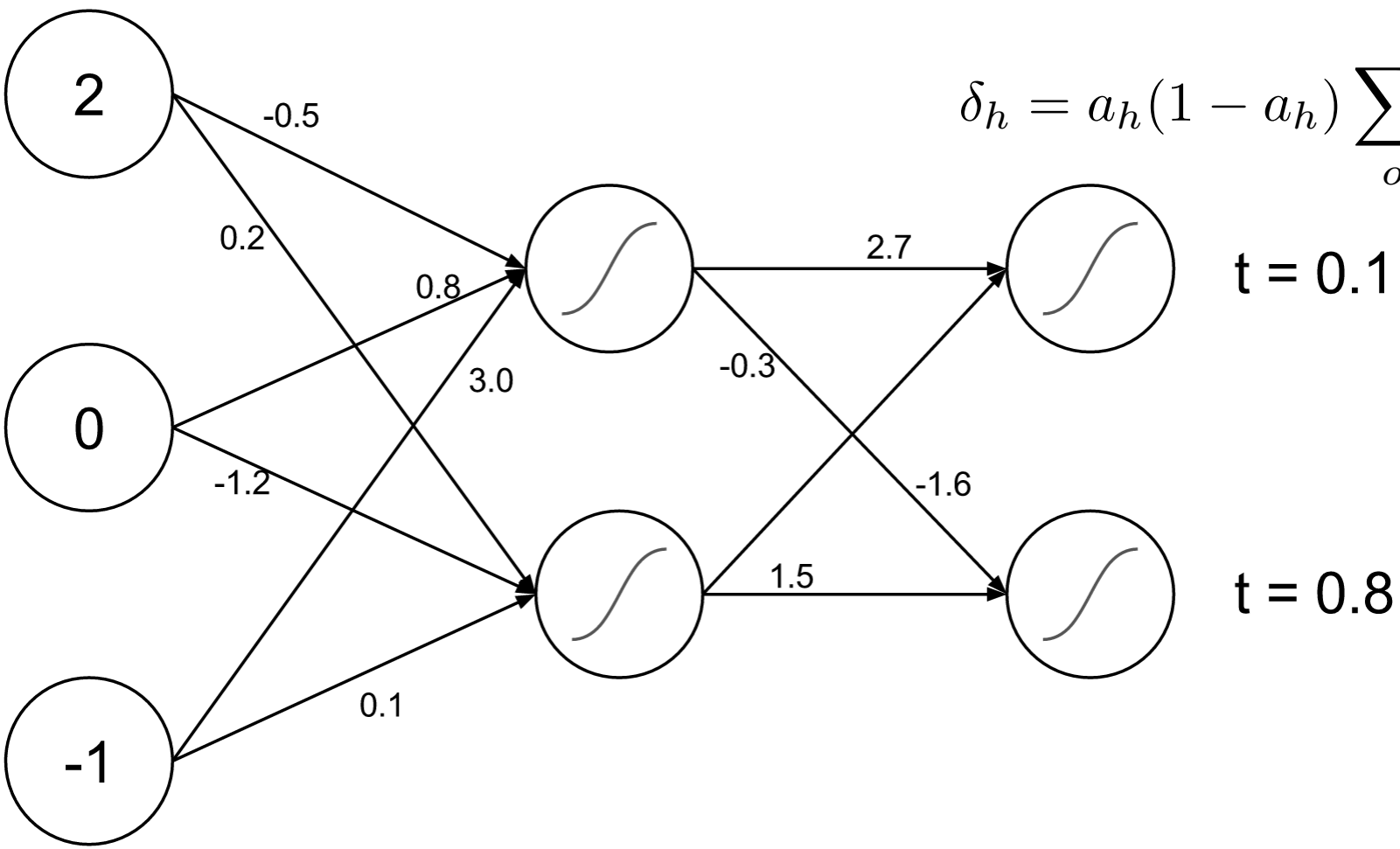
Example Backpropagation Update

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$w_{ij} \mathrel{+}= \alpha \delta_j a_i$$

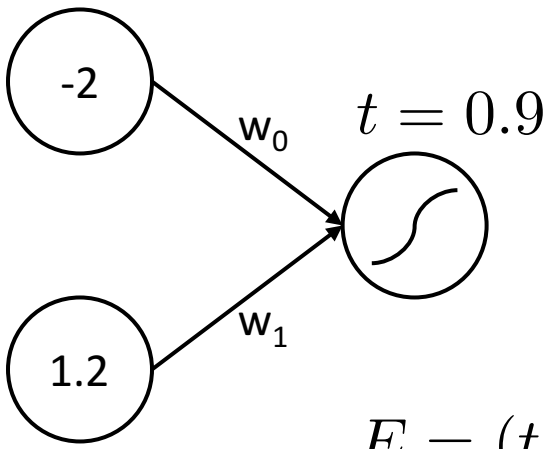
$$\delta_o = a_o(1-a_o)(t-a_o)$$

$$\delta_h = a_h(1-a_h)\sum_o w_{ho}\delta_o$$



Backpropagation Derivation

How would the error change if the weight changed?



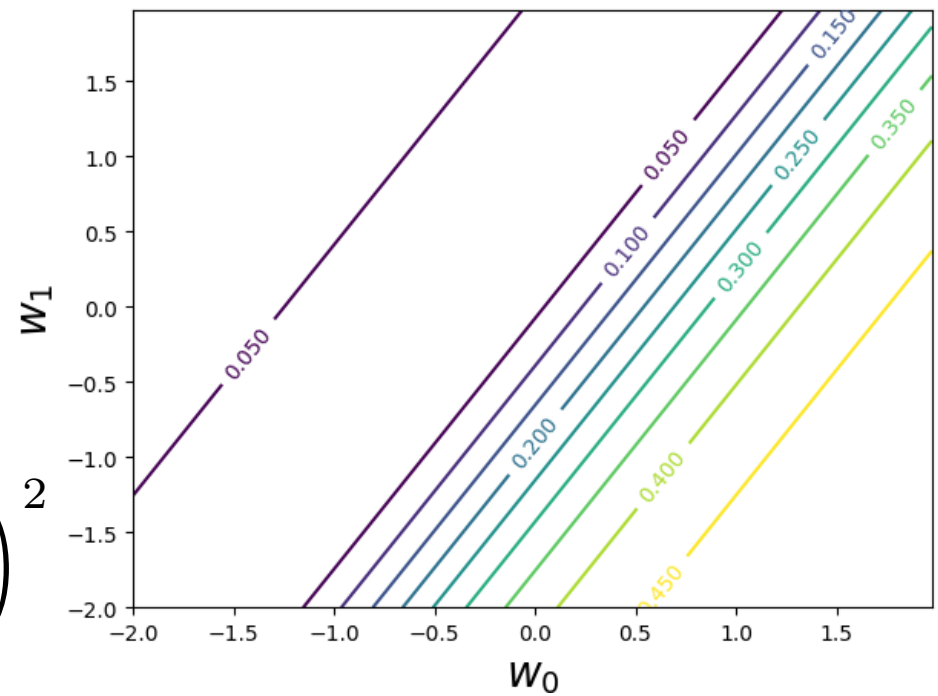
$$t = 0.9$$

$$E = (t - a_o)^2$$

$$E(t, \vec{a}_h, \vec{w}_{ho}) = \left(t - \frac{1}{1 + e^{-\vec{a}_h \cdot \vec{w}_{ho}}}\right)^2$$

$$\nabla_{\vec{w}}\ E = \begin{bmatrix} \frac{\partial E}{\partial w_0} \\ \\ \frac{\partial E}{\partial w_1} \end{bmatrix}$$

Error surface



Local Search Issues

- Backpropagation is performing stochastic gradient descent.
- This is a form of local search.

- Like other local search methods, it can get stuck in:
  - Local minima
  - Plateaus

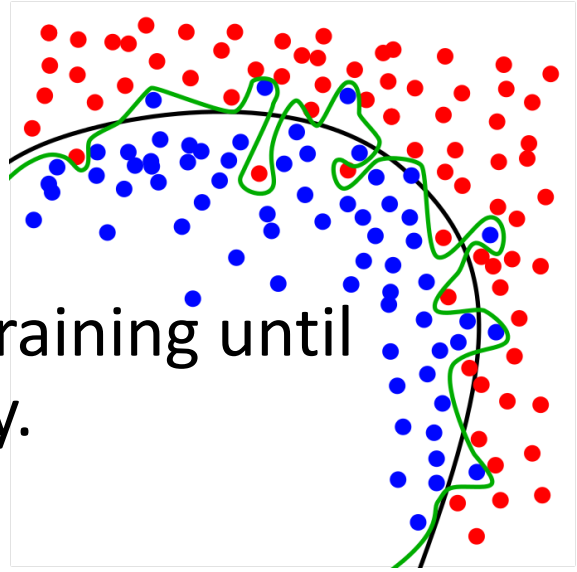
- High dimensionality helps a bit, because it’s hard to be stuck on all dimensions simultaneously.

Local Search Improvements

We can use the techniques we already know for improving local search.

- Random moves
  - We're already doing this (by randomly ordering training examples on each epoch), hence the "stochastic".
  - Non-random moves would mean computing average error over all training examples before doing a backpropagation step.
- Random restarts
  - More precisely: we want to start in a good region of the search space. This is a big issue in deep learning.
- Momentum
  - Keep going in the direction you were already going.

Overfitting



- In your lab implementation, you keep training until the whole data set is classified correctly.
- Why is this a bad idea?
  - We might learn an extremely complicated function.
  - A simpler function is likely to generalize better.

- Possible solutions:
  - Weight decay: dampen all weights by some small factor every round.
  - Learn with targets of 0.1 and 0.9 instead of 0 and 1.
  - Cross validation: split into training and test sets; stop training when performance stops improving on the **test set**.