Analyzing Backprop

3-4-16
Reading Quiz

Q1: If a neural network has 3 layers with 10 input, 6 hidden, and 8 output units, what is the dimension of backpropagation’s local search space?

a) $10 + 6 + 8 = 24$

b) $10 + 6 \times 8 = 58$

c) $10 \times 6 + 6 \times 8 = 108$

d) $10 \times 6 + 10 \times 8 + 6 \times 8 = 188$

e) $10 \times 6 \times 8 = 480$
Reading Quiz

Q2: An arbitrary function can be approximated by a neural network with ____ (non-input) layers.

a) 1
b) 2
c) 3
d) 4
e) infinite
Backpropagation Review

for 1:epochs

    for each example in training_data:
        run example through network
        compute error for each output node
    for each layer (starting from output):
        for each node in layer:
            update_weights(node)
Updating weights

for each incoming edge $i$:

$$w_i = w_i + \eta \delta x_i$$

if node is in the output layer:

$$\delta_o = o_o (1 - o_o) (t_o - o_o)$$

if node is in a hidden layer:

$$\delta_h = o_h (1 - o_h) \left( \sum_k \delta_k w_{hk} \right)$$

all nodes in the next layer
Local search issues

Backpropagation is performing local search in a high-dimensional space.

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

High dimensionality helps a bit, because it’s hard to be at a local minimum in every dimension 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).
  - Non-random moves would mean computing average error over all training examples before doing a backpropagation step.
- random restarts
  - In conx, the function n.reset() gives new random initial weights.
- momentum
  - Keep moving in the same direction: \( \Delta w = \eta \delta x_i + \alpha \Delta w \)
Overfitting

Don’t just run `n.train()`!!!
This will learn the training data perfectly and fit the test data badly.

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.
Output representation

For classification:
- Round the output sigmoids (treat them as thresholds).
- 1-of-n is better than more compact representations. Why?

For regression:
- Sigmoid output is continuous, but bounded between 0 and 1.
- Normalize the targets to the range [0,1] before training.

For dimensionality reduction:
- Throw away the output layer and make the hidden units the output.
A perspective from 15 years ago

- Backpropagation is extremely slow to converge and requires tons of input data on networks with many hidden layers.
- Having multiple hidden layers makes the network hard to interpret.
- A 3-layer network can represent any function.
- Why bother with deep (many-layer) networks?
A more recent perspective

- Shallow networks with huge hidden layers make the learning problem harder.
- We can use GPU parallelization to speed up training.
- If we need tons of data, we can get it.
- We can set backpropagation up for success by how we design the network.
Deep Learning

Convolutional neural networks
- Hidden layer units connected to only a small subset of the previous layer.
- Connections have spatial locality (input from several nearby pixels).
- These hidden units “convolve” the input (like a blurring filter).

Deep belief networks
- Unsupervised pre-training of hidden layers (like the encoder example).
- Use weight reduction or smaller layers to avoid exact matching.
- Puts the backprop starting point in a good region of weight space.