Take Home Messages

Traditional Vs. Neural Computation

- Traditional:
  - serial
  - damage intolerant
  - programmed
  - specific, precise input-output behavior
- Neural
  - massively parallel
  - damage tolerant
  - trained
  - approximate, generalization behavior

Early NNs

- McCullugh-Pitts demonstrates a logical calculus in a neural net
  - excitatory and inhibitory connnections
  - logical implications
  - memory cells (reccurrent connections)
- Hebb
  - weight training as the mechanism of learning

Single Layer Nets (Perceptrons)

- The idea of decision surfaces is key!
- A single layer net (regardless of node function) has inherent limitations

Mutli-Layer Nets

- "Three" (aka "Two") layers + nonlinear hidden node functions is sufficient for any mapping…
- given sufficient numbers of hidden layer nodes.

The Delta Rule

- Update each weight by taking steps in the negative gradient direction, with respect to that weight…
- In it's multi-layer form, this gives the *backpropagation algorithm*
- This is the most widely used NN training algorithm

The most common NN

- Three layers
  - sufficient
- Sigmoidal node functions
  - continuous approximations of thresholds
  - can form "activation bumps"
- Fully connected layers (feed forward)
- backpropagation
  - well founded in the calculus

The most common NN problems

- Three layers
  - sufficient, but with how many hidden layer nodes?
- backpropagation
  - can be too slow with large, fully-connected nets
  - can get stuck in local minima

Generalization...

- Finding "good" training data is the key.
- Overfitting:
  - great on training data, terrible on everything else
  - too many hidden layer nodes
- Underfitting:
  - terrible on everything, including training data
  - too few hidden layer nodes

Kohonen Networks

- Clustering, which amounts to
- mapping a high dimensional space to a lower dimensional space
- this is so-called "unsupervised" learning

Self-Organizing Maps

- Mapping from a high dimensional space to one of lower dimension,
- while maintaining "gross" topology
- Understand the update rule, in terms of the input space and the node space.

Radial Basis Functions

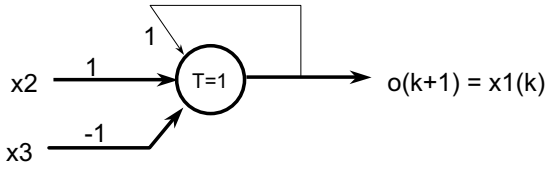
- Node functions where output is directly dependent on some measure of distance between the input and the nodes "center"
- This is a form of "receptive field", allowing for "spatially localized learning"
  - less "forgetting"
  - faster (training and execution)

Note...

- Each of the things we've covered:
  - sigmoidal nodes, linear nodes, BP, SOM, RBF
- are all design tools, not necessarily end products

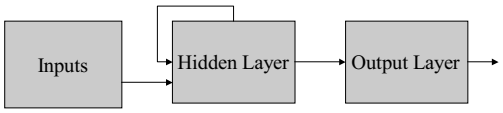
One thing we didn't cover (well, lots of things, really)

- Recurrent connections
- Remember the McCulloch-Pitts Memory Cell:



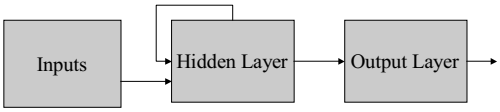
Recurrent Networks for Internal Memory

- Couldn't be simpler...



Recurrent Networks for Internal Memory

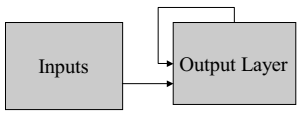
- Couldn't be more complicated



Note that all outputs, and therefore all errors, and their deriv atives depend on a recursive series of terms...so we use truncated Backpropagation through time (BPTT)

Hopfield ("Settling") Networks for Optimization, Memory

- Apply inputs



•And iterate until the network finds an equilibrium,
• Then the output is your solution
•Opinions offered here

Other Uncovered Topics

- Support Vector Machines
  - Largely networks with alternative formulations of the hidden layer
- Boltzmann Machines
  - Weight determination through simulated anealing (in my opinion)
- Neurodynamic Programming
  - Largely related to reinforcement learning theory
- Lots of other stuff, I'm sure