

# Neural Networks

CPS 271  
Ron Parr

## Motivation for non-linear Classifiers

- Linear methods are “weak”
  - Make strong assumptions
  - Can only express relatively simple functions of inputs
- Coming up with good features can be hard
- Why not make the classifier to more work for us?
  - What does the space of hypotheses look like?
  - How do we navigate in this space?

## Neural Network Motivation

- Human brains are only known example of actual intelligence
- Individual neurons are slow, boring
- Brains succeed by using massive parallelism
- Idea: Copy what works
- Raises many issues:
  - Is the computational metaphor suited to the computational hardware?
  - How do we know if we are copying the important part?
  - Are we aiming too low?



## Why Neural Networks?

Maybe computers should be more brain-like:

	Computers	Brains
Computational Units	$10^9$ gates/CPU	$10^{11}$ neurons
Storage Units	$10^{10}$ bits RAM $10^{12}$ bits HD	$10^{11}$ neurons $10^{14}$ synapses
Cycle Time	$10^{-9}$ S	$10^{-3}$ S
Bandwidth	$10^{10}$ bits/s*	$10^{14}$ bits/s
Compute Power	$10^{10}$ Ops/s	$10^{14}$ Ops/s

## Comments on Blue Gene

- Blue Gene: World’s Fastest Supercomputer
- 360 Teraflops
- Currently at 131,000 processors
- $10^{13}$  ->  $10^{14}$  Ops/s (brain level?)
- 16 TB memory ( $10^{13}$  bits)
- 14 Megawatts power (\$1M/year in electricity)
- 2500 sq ft size (nice sized house)
- Pictures and other details:
  - [http://domino.research.ibm.com/comm/pr.nsf/pages/rsc.bluegene\\_2004.html](http://domino.research.ibm.com/comm/pr.nsf/pages/rsc.bluegene_2004.html)

## More Comments on Blue Gene

- What is wrong with this picture?
  - Weight
  - Size
  - Power Consumption
- What is missing?
  - Still can’t replicate human abilities (though vastly exceeds human abilities in many areas)
  - Are we running the wrong programs?
  - Is the architecture well suited to the programs we might need to run?

## Artificial Neural Networks

- Develop *abstraction* of function of actual neurons
- Simulate large, massively parallel artificial neural networks on conventional computers
- Some have tried to build the hardware too
- Try to approximate human learning, robustness to noise, robustness to damage, etc.

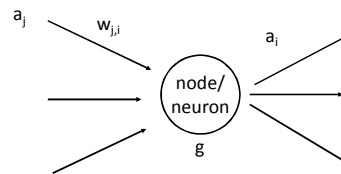
## Use of neural networks

- Trained to pronounce English
  - Training set: Sliding window over text, sounds
  - 95% accuracy on training set
  - 78% accuracy on test set
- Trained to recognize handwritten digits
  - >99% accuracy
- Trained to drive (Pomerleau's no-hands across America)

## Neural Network Lore

- Neural nets have been adopted with an almost religious fervor within the AI community - several times
- Often ascribed near magical powers by people, usually those who know the least about computation or brains
- For most AI people, magic is gone, but neural nets remain extremely interesting and useful mathematical objects

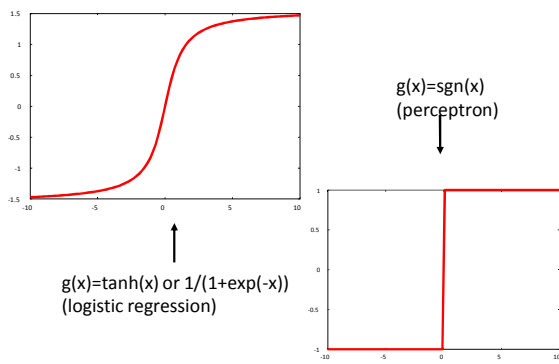
## Artificial Neurons



$$a_i = f\left(\sum_j w_{j,i} a_j\right)$$

$f$  can be any function, but usually a smoothed step function

## Threshold Functions



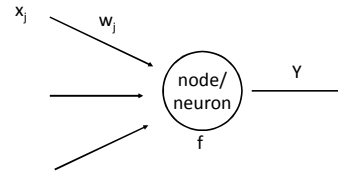
## Network Architectures

- Cyclic vs. Acyclic
  - Cyclic is tricky, but more biologically plausible
    - Hard to analyze in general
    - May not be stable
    - Need to assume latches to avoid race conditions
  - Hopfield nets: special type of cyclic net useful for associative memory
- Single layer (perceptron)
- Multiple layer

## Feedforward Networks

- We consider acyclic networks
- One or more computational layers
- Entire network can be viewed as computing a complex non-linear function
- Typical uses in learning:
  - Classification (usually involving complex patterns)
  - General continuous function approximation

## Perceptron



$f$  is a simple step function (sgn)

## Perceptron Learning

- We are given a set of inputs  $\mathbf{x}^{(1)} \dots \mathbf{x}^{(n)}$
- $t^{(1)} \dots t^{(n)}$  is a set of target outputs (boolean)  $\{-1, 1\}$
- $\mathbf{w}$  is our set of weights
- output of perceptron =  $\mathbf{w}^T \mathbf{x}$
- Perceptron\_error( $\mathbf{x}^{(i)}, \mathbf{w}$ ) =  $-\text{net}(\mathbf{x}^{(i)}, \mathbf{w}) t^{(i)}$
- Goal: Pick  $\mathbf{w}$  to optimize:

$$\min_{\mathbf{w}} \sum_{i \in \text{misclassified}} \text{perceptron\_error}(\mathbf{x}^{(i)}, \mathbf{w})$$

## Update Rule

Repeat until convergence:

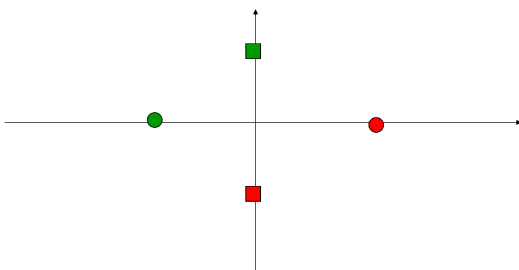
$$\forall_{i \in \text{misclassified}} \forall_j : w_j \leftarrow w_j + \alpha x_j^{(i)} t^{(i)}$$

↑  
"Learning Rate"  
(can be any constant)

- $i$  iterates over samples
- $j$  iterates over weights

<http://neuron.eng.wayne.edu/java/Perceptron/New38.html>

## Visualizing Linearly Separable Functions



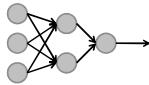
Is red linearly separable from green?  
Are the circles linearly separable from the squares?

## Observations

- Linear separability is fairly weak
- We have other tricks:
  - Functions that are not linearly separable in one space, may be linearly separable in another space
  - If we engineer our inputs to our neural network, then we change the space in which we are constructing linear separators
  - Every function has a linear separator (in some space)
- Perhaps other network architectures will help

## Multilayer Networks

- Once people realized how simple perceptrons were, they lost interest in neural networks for a while
- Multilayer networks turn out to be much more expressive (with a smoothed step function)
  - Use sigmoid, e.g.,  $\tanh(w^T x)$
  - With 2 layers, can represent any continuous function
  - With 3 layers, can represent many discontinuous functions
- Tricky part: How to adjust the weights



## Smoothing Things Out

- Idea: Do gradient descent on a smooth error function
- Error function is sum of squared errors
- Consider a single training example first

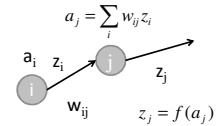
$$E = 0.5 \text{error}(X^{(i)}, w)^2$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial a_j} \frac{\partial a_j}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial a_j} = \delta_j$$

$$\frac{\partial a_j}{\partial w_{ij}} = z_i$$

$$\frac{\partial E}{\partial w_{ij}} = \delta_j z_i$$



## Propagating Errors

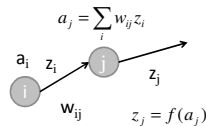
- For output units (assuming no weights on outputs)

$$\frac{\partial E}{\partial a_j} = \delta_j = y - t$$

- For hidden units

$$\frac{\partial E}{\partial a_i} = \delta_i = \sum_k \frac{\partial E}{\partial a_k} \frac{\partial a_k}{\partial a_i} = f'(a_i) \sum_k w_{ki} \delta_k$$

All upstream nodes from  $i$



## Putting it together

- Apply input  $x$  to network (sum for multiple inputs)
  - Compute all activation levels
  - Compute final output
- Compute  $\delta$  for output units

$$\delta = y - t$$

- Backpropagate  $\delta$ 's to hidden units

$$\delta_j = \sum_k \frac{\partial E}{\partial a_k} \frac{\partial a_k}{\partial a_j} = f'(a_j) \sum_k w_{kj} \delta_k$$

- Compute gradient update  $\frac{\partial E}{\partial w_{ij}} = \delta_j a_i$

## Summary of Gradient Update

- Gradient calculation, parameter update have recursive formulation
- Decomposes into:
  - Local message passing
  - No transcendentals:  $g'(x) = 1 - g(x)^2$  for  $\tanh(x)$
- Highly parallelizable
- Biologically plausible(?)
- Celebrated *backpropagation* algorithm

## Good News

- Can represent any continuous function with two layers (1 hidden)
- Can represent essentially any function with 3 layers
- (But how many hidden nodes?)
- Multilayer nets are a universal approximation architecture with a highly parallelizable training algorithm

## Back-prop Issues

- Backprop = gradient descent on error function
- Function is nonlinear (= powerful)
- Function is nonlinear (= local minima)
- Big nets:
  - Many parameters
    - Many optima
    - Slow gradient descent
    - Risk of overfitting
  - Biological plausibility  $\neq$  Electronic plausibility
- Many NN experts became experts in numerical analysis (by necessity)

## Neural Network Tricks

- Many gradient descent acceleration tricks
- Early stopping
- Methods of enforcing transformation invariance
  - Modify error function
  - Transform/augment training data
  - Weight sharing
- Handcrafted network architectures

## Neural Nets in Practice

- Many applications for pattern recognition tasks
- Very powerful representation
  - Can overfit
  - Can fail to fit with too many parameters, poor features
- Very widely deployed AI technology, but
  - Few open research questions (Best way to get a machine learning paper rejected: “Neural Network” in title.)
  - Connection to biology still uncertain
  - Results are hard to interpret
- “Second best way to solve any problem”
  - Can do just about anything w/enough twiddling
  - Now third or fourth to SVMs, boosting, and ???