

For Wednesday

- Read chapter 22, sections 1-3
- No homework

Program 4

- Any questions?

Picking the Best Literal

- Based on information gain (similar to ID3).

$$|p| * (\log_2 (|p| / (|p| + |n|)) - \log_2 (|P| / (|P| + |N|)))$$

P is number of positives before adding literal L

N is number of negatives before adding literal L

p is number of positives after adding literal L

n is number of negatives after adding literal L

- Given n predicates of arity m there are $O(n2^m)$ possible literals to choose from, so branching factor can be quite large.

Other Approaches

- Golem
- CHILL
- Foidl
- Bufoidl

Domains

- Any kind of concept learning where background knowledge is useful.
- Natural Language Processing
- Planning
- Chemistry and biology
 - DNA
 - Protein structure

Why Neural Networks?

Why Neural Networks?

- Analogy to biological systems, the best examples we have of robust learning systems.
- Models of biological systems allowing us to understand how they learn and adapt.
- **Massive parallelism** that allows for computational efficiency.
- Graceful degradation due to **distributed representations** that spread knowledge representation over large numbers of computational units.
- Intelligent behavior is an **emergent property** from large numbers of simple units rather than resulting from explicit symbolically encoded rules.

Neural Speed Constraints

- Neuron “switching time” is on the order of milliseconds compared to nanoseconds for current transistors.
- A factor of a **million** difference in speed.
- However, biological systems can perform significant cognitive tasks (vision, language understanding) in seconds or tenths of seconds.

What That Means

- Therefore, there is only time for about a hundred serial steps needed to perform such tasks.
- Even with limited abilities, current AI systems require orders of magnitude more serial steps.
- Human brain has approximately 10^{11} neurons each connected on average to 10^4 others, therefore must exploit massive parallelism.

Real Neurons

- Cells forming the basis of neural tissue
 - Cell body
 - Dendrites
 - Axon
 - Syntaptic terminals
- The electrical potential across the cell membrane exhibits spikes called action potentials.
- Originating in the cell body, this spike travels down the axon and causes chemical neurotransmitters to be released at syntaptic terminals.
- This chemical difuses across the synapse into dendrites of neighboring cells.

Real Neurons (cont)

- Synapses can be excitatory or inhibitory.
- Size of synaptic terminal influences strength of connection.
- Cells “add up” the incoming chemical messages from all neighboring cells and if the net positive influence exceeds a threshold, they “fire” and emit an action potential.

Model Neuron (Linear Threshold Unit)

- Neuron modelled by a unit (j) connected by weights, w_{ji} , to other units (i):
- Net input to a unit is defined as:

$$net_j = \sum w_{ji} * o_i$$

- Output of a unit is a threshold function on the net input:
 - 1 if $net_j > T_j$
 - 0 otherwise

Neural Computation

- McCollough and Pitts (1943) show how linear threshold units can be used to compute logical functions.
- Can build basic logic gates
 - AND: Let all w_{ji} be $(T_j/n) + \varepsilon$ where $n =$ number of inputs
 - OR: Let all w_{ji} be $T_j + \varepsilon$
 - NOT: Let one input be a constant 1 with weight $T_j + \varepsilon$ and the input to be inverted have weight $-T_j$

Neural Computation (cont)

- Can build arbitrary logic circuits, finite-state machines, and computers given these basis gates.
- Given negated inputs, two layers of linear threshold units can specify any boolean function using a two-layer AND-OR network.

Learning

- Hebb (1949) suggested if two units are both active (firing) then the weight between them should increase:

$$w_{ji} = w_{ji} + \eta o_j o_i$$

- η is a constant called the learning rate
- Supported by physiological evidence

Alternate Learning Rule

- Rosenblatt (1959) suggested that if a target output value is provided for a single neuron with fixed inputs, can incrementally change weights to learn to produce these outputs using the [perceptron learning rule](#).
 - Assumes binary valued input/outputs
 - Assumes a single linear threshold unit.
 - Assumes input features are detected by fixed networks.

Perceptron Learning Rule

- If the target output for output unit_j is t_j

$$w_{ji} = w_{ji} + \eta(t_j - o_j)o_i$$

- Equivalent to the intuitive rules:
 - If output is correct, don't change the weights
 - If output is low ($o_j = 0, t_j = 1$), increment weights for all inputs which are 1.
 - If output is high ($o_j = 1, t_j = 0$), decrement weights for all inputs which are 1.
- Must also adjust threshold:

$$T_j = T_j + \eta(t_j - o_j)$$

- or equivalently assume there is a weight $w_{j0} = -T_j$ for an extra input unit 0 that has constant output $o_0 = 1$ and that the threshold is always 0.

Perceptron Learning Algorithm

- *Repeatedly iterate through examples adjusting weights according to the perceptron learning rule until all outputs are correct*

Initialize the weights to all zero (or randomly)

Until outputs for all training examples are correct

For each training example, e , do

 Compute the current output o_j

 Compare it to the target t_j and update the weights according to the perceptron learning rule.

Algorithm Notes

- Each execution of the outer loop is called an **epoch**.
- If the output is considered as concept membership and inputs as binary input features, then easily applied to concept learning problems.
- For multiple category problems, learn a separate perceptron for each category and assign to the class whose perceptron most exceeds its threshold.
- When will this algorithm terminate (converge) ??

Representational Limitations

- Perceptrons can only represent linear threshold functions and can therefore only learn data which is **linearly separable** (positive and negative examples are separable by a hyperplane in n-dimensional space)
- Cannot represent exclusive-or (xor)

Perceptron Learnability

- System obviously cannot learn what it cannot represent.
- Minsky and Papert(1969) demonstrated that many functions like parity (n-input generalization of xor) could not be represented.
- In visual pattern recognition, assumed that input features are local and extract feature within a fixed radius. In which case no input features support learning
 - Symmetry
 - Connectivity
- These limitations discouraged subsequent research on neural networks.

Perceptron Convergence and Cycling Theorems

- **Perceptron Convergence Theorem:** If there are a set of weights that are consistent with the training data (i.e. the data is linearly separable), the perceptron learning algorithm will converge (Minsky & Papert, 1969).
- **Perceptron Cycling Theorem:** If the training data is not linearly separable, the Perceptron learning algorithm will eventually repeat the same set of weights and threshold at the end of some epoch and therefore enter an infinite loop.

Perceptron Learning as Hill Climbing

- The **search space** for Perceptron learning is the space of possible values for the weights (and threshold).
- The **evaluation metric** is the error these weights produce when used to classify the training examples.
- The perceptron learning algorithm performs a form of hill-climbing (gradient descent), at each point altering the weights slightly in a direction to help minimize this error.
- Perceptron convergence theorem guarantees that for the linearly separable case there is only one local minimum and the space is well behaved.

Perceptron Performance

- Can represent and learn conjunctive concepts and M-of-N concepts (true if any M of a set of N selected binary features are true).
- Although simple and restrictive, this high-bias algorithm performs quite well on many realistic problems.
- However, the representational restriction is limiting in many applications.

Multi-Layer Neural Networks

- **Multi-layer networks** can represent arbitrary functions, but building an effective learning method for such networks was thought to be difficult.
- Generally networks are composed of an **input layer**, **hidden layer**, and **output layer** and activation feeds forward from input to output.
- Patterns of activation are presented at the inputs and the resulting activation of the outputs is computed.
- The values of the weights determine the function computed.
- A network with **one hidden layer** with a sufficient number of units can represent **any boolean function**.

Basic Problem

- General approach to the learning algorithm is to apply gradient descent.
- However, for the general case, we need to be able to differentiate the function computed by a unit and the standard threshold function is not differentiable at the threshold.

Differentiable Threshold Unit

- Need some sort of non-linear output function to allow computation of arbitrary functions by multilayer networks (a multi-layer network of linear units can still only represent a linear function).
- Solution: Use a nonlinear, differentiable output function such as the sigmoid or logistic function

$$o_j = 1 / (1 + e^{-(net_j - T_j)})$$

- Can also use other functions such as tanh or a Gaussian.

Error Measure

- Since there are multiple continuous outputs, we can define an overall error measure:

$$E(W) = 1/2 * (\sum_{d \in D} \sum_{k \in K} (t_{kd} - o_{kd})^2)$$

where D is the set of training examples, K is the set of output units, t_{kd} is the target output for the k^{th} unit given input d, and o_{kd} is network output for the k^{th} unit given input d.

Gradient Descent

- The derivative of the output of a sigmoid unit given the net input is

$$\partial o_j / \partial \text{net}_j = o_j(1 - o_j)$$

- This can be used to derive a learning rule which performs gradient descent in weight space in an attempt to minimize the error function.

$$\Delta w_{ji} = -\eta(\partial E / \partial w_{ji})$$

Backpropagation Learning Rule

- Each weight w_{ji} is changed by

$$\Delta w_{ji} = \eta \delta_j o_i$$

$$\delta_j = o_j (1 - o_j) (t_j - o_j) \quad \text{if } j \text{ is an output unit}$$

$$\delta_j = o_j (1 - o_j) \sum \delta_k w_{kj} \quad \text{otherwise}$$

where η is a constant called the learning rate,
 t_j is the correct output for unit j ,
 d_j is an error measure for unit j .

- First determine the error for the output units, then backpropagate this error layer by layer through the network, changing weights appropriately at each layer.

Backpropagation Learning Algorithm

- Create a three layer network with N hidden units and fully connect input units to hidden units and hidden units to output units with small random weights.

Until all examples produce the correct output within ϵ or the mean-squared error ceases to decrease (or other termination criteria):

 Begin epoch

 For each example in training set do:

 Compute the network output for this example.

 Compute the error between this output and the correct output.

 Backpropagate this error and adjust weights to decrease this error.

 End epoch

- Since continuous outputs only approach 0 or 1 in the limit, must allow for some ϵ -approximation to learn binary functions.

Comments on Training

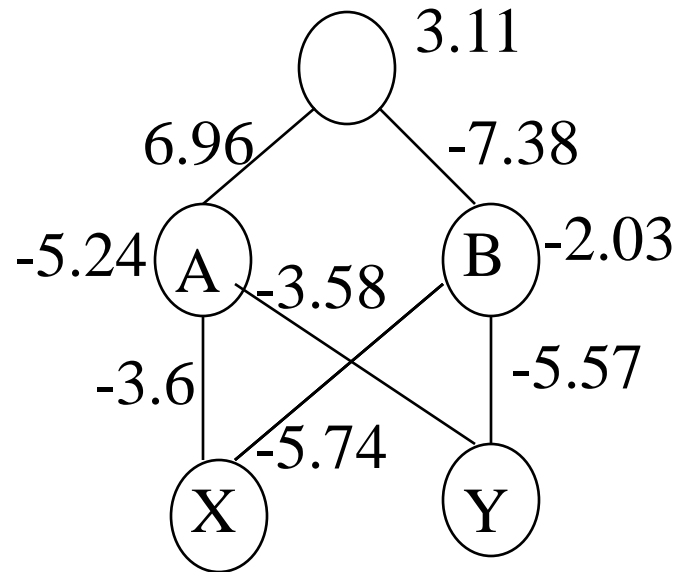
- There is no guarantee of convergence, may oscillate or reach a local minima.
- However, in practice many large networks can be adequately trained on large amounts of data for realistic problems.
- Many epochs (thousands) may be needed for adequate training, large data sets may require hours or days of CPU time.
- Termination criteria can be:
 - Fixed number of epochs
 - Threshold on training set error

Representational Power

Multi-layer sigmoidal networks are very expressive.

- **Boolean functions:** Any Boolean function can be represented by a two layer network by simulating a two-layer AND-OR network. But number of required hidden units can grow exponentially in the number of inputs.
- **Continuous functions:** Any bounded continuous function can be approximated with arbitrarily small error by a two-layer network. Sigmoid functions provide a set of basis functions from which arbitrary functions can be composed, just as any function can be represented by a sum of sine waves in Fourier analysis.
- **Arbitrary functions:** Any function can be approximated to arbitrary accuracy by a three-layer network.

Sample Learned XOR Network



Hidden unit A represents $\neg(X \wedge Y)$

Hidden unit B represents $\neg(X \vee Y)$

Output O represents:

$$A \wedge \neg B$$
$$\neg(X \wedge Y) \wedge (X \vee Y)$$
$$X \oplus Y$$

Hidden Unit Representations

- Trained hidden units can be seen as newly constructed features that re-represent the examples so that they are linearly separable.
- On many real problems, hidden units can end up representing interesting recognizable features such as vowel-detectors, edge-detectors, etc.
- However, particularly with many hidden units, they become more “distributed” and are hard to interpret.

Input/Output Coding

- Appropriate coding of inputs and outputs can make learning problem easier and improve generalization.
- Best to encode each binary feature as a separate input unit and for multi-valued features include one binary unit per value rather than trying to encode input information in fewer units using binary coding or continuous values.

I/O Coding cont.

- Continuous inputs can be handled by a single input by scaling them between 0 and 1.
- For disjoint categorization problems, best to have one output unit per category rather than encoding n categories into $\log n$ bits. Continuous output values then represent certainty in various categories. Assign test cases to the category with the highest output.
- Continuous outputs (regression) can also be handled by scaling between 0 and 1.

Neural Net Conclusions

- Learned concepts can be represented by networks of linear threshold units and trained using gradient descent.
- Analogy to the brain and numerous successful applications have generated significant interest.
- Generally much slower to train than other learning methods, but exploring a rich hypothesis space that seems to work well in many domains.
- Potential to model biological and cognitive phenomenon and increase our understanding of real neural systems.
 - Backprop itself is not very biologically plausible