Chapter 11 (part 2): Learning

Examples of Representations

1. **Neural Networks**
2. **Case-based Reasoning**

D. Poole, A. Mackworth, and R. Goebel, *Computational Intelligence: A Logical Approach*, Oxford University Press, January 1998

Neural Networks (NN)

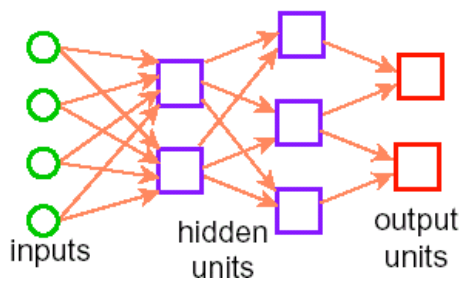
- These representations are inspired by neurons and their connections in the brain (dendrite, axon, synapse).
- Artificial neurons, or units, have inputs, and an output. The output can be connected to the inputs of other units.
- The output of a unit is a parameterized non-linear function of its inputs.
- Learning occurs by adjusting parameters to fit data.
- NNs can represent an approximation to any function.

Why Neural Networks?

- As part of neuroscience, in order to understand real neural systems, researchers are simulating the neural systems of simple animals such as worms.
- It seems reasonable to try to build the functionality of the brain via the mechanism of the brain (suitably abstracted).
- The brain inspires new ways to think about computation.
- NNs provide a different measure of simplicity as a learning bias.

Feed-forward NNs

- Feed-forward neural networks are the most common models.
- These are directed acyclic graphs:



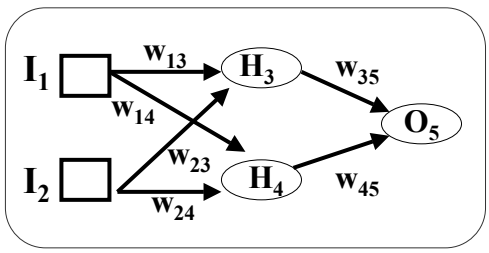
The Units

- A unit with *k* inputs is like the parameterized logic program:

prop(Obj, output, V) ←
prop(Obj, in1, I1) ∧ *% input 1 = I1*
prop(Obj, in2, I2) ∧ *% input 2 = I2*
…
prop(Obj, ink, Ik) ∧ *% input k = Ik*
V is f $(w_0 + w_1 * I_1 + w_2 I_2 + \ldots + w_k * I_k)$.

- I_j are real-valued inputs.
- w_j are adjustable real parameters.
- *f* is an activation function.

Example of a two-layer NNs

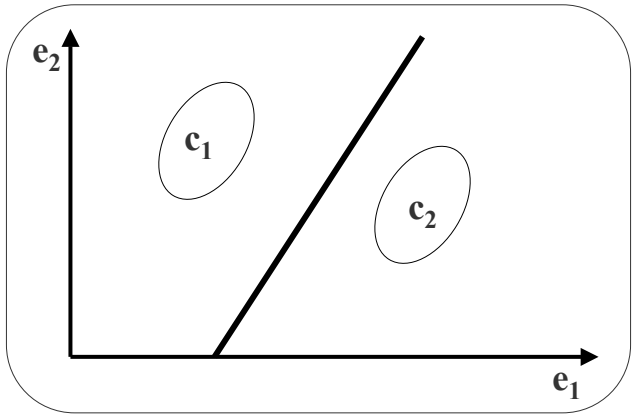


Two-layer feed-forward network with 2 hidden nodes, one output node

a_i = output of node i

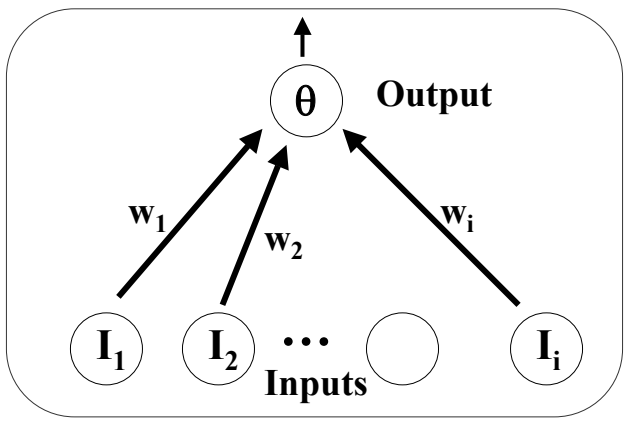
$a_5 = g(w_{35}a_3 + w_{45}a_4)$
$= g(w_{35}\, g(w_{13}a_1 + w_{23}a_2) + w_{45}\, g(w_{14}a_1 + w_{24}a_2))$

Thresholds and linear discriminant



$w_1e_1 + w_2e_2 \ldots + w_de_d - w_0$

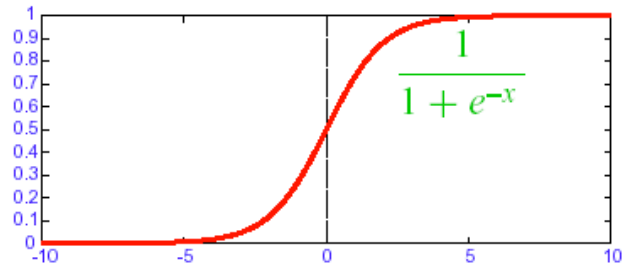
Perceptrons (linear discriminant)



$$\text{output} = \begin{cases} 1 & \text{if } \sum_i w_i I_i + \theta > 0 \\ 0 & \text{otherwise} \end{cases}$$

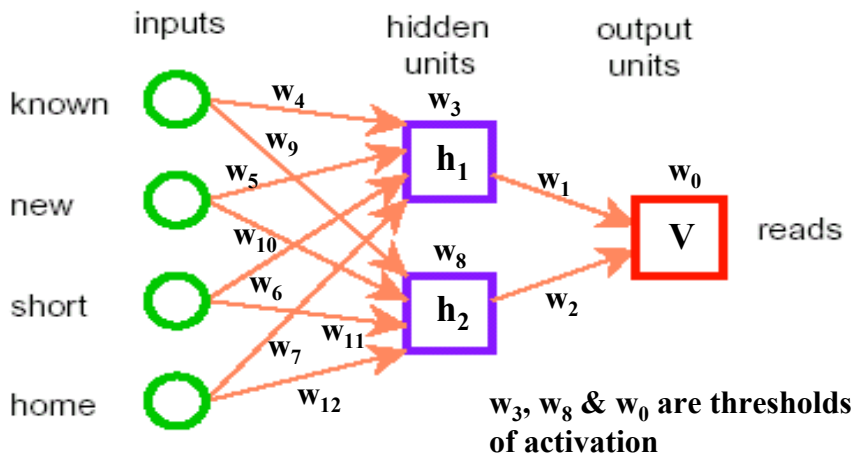
Activation function

- A typical activation function is the sigmoid function:



$$f(x) = \frac{1}{1+e^{-x}} \qquad f'(x) = f(x)(1-f(x))$$

Neural Network for the news example



Axiomatizing the Network

- The values of the attributes are real numbers.
- Thirteen parameters $w_0, \ldots, w_{12}$ are real numbers.
- The attributes h_1 and h_2 correspond to the values of hidden units.
- There are 13 real numbers to be learned. The hypothesis space is thus a 13-dimensional real space.
- Each point in this 13-dimensional space corresponds to a particular logic program that predicts a value for *reads* given *known, new, short,* and *home*.

predicted_prop(Obj, reads, V) ←
prop(Obj, h_1, I_1) ∧ *prop(Obj, h_2, I_2)* ∧
*V is f (w_0 + w_1*I_1 + w_2*I_2).*
prop(Obj, h_1, V) ←
prop(Obj, known, I_1) ∧ *prop(Obj, new, I_2)* ∧
prop(Obj, short, I_3) ∧ *prop(Obj, home, I_4)* ∧
*V is f (w_3 + w_4*I_1 + w_5*I_2 + w_6*I_3 + w_7*I_4).*
prop(Obj, h_2, V) ←
prop(Obj, known, I_1) ∧ *prop(Obj, new, I_2)* ∧
prop(Obj, short, I_3) ∧ *prop(Obj, home, I_4)* ∧
*V is f (w_8 + w_9*I_1 + w_{10}*I_2 + w_{11}*I_3 + w_{12}*I_4).*

Activation of V

$V = g(w_0 + w_1 h_1 + w_2 h_2)$

$= g(w_0 + w_1\, g(w_3 + w_4 a_1 + w_5 a_2 + w_6 a_3 + w_7 a_4)$
$+ w_2\, g(w_8 + w_9 a_1 + w_{10} a_2 + w_{11} a_3 + w_{12} a_4))$

$= g(w_0 + w_1\, g(w1\, g(w3 + w_4 I_1 + w_5 I_2 + w_6 I_3 + w_7 I_4)$
$+ w_2\, g(w_8 + w_9 I_1 + w_{10} I_2 + w_{11} I_3 + w_{12} I_4))$

Prediction Error

- For particular values for the parameters $\overline{\mathbf{w}} = w_0, \ldots, w_m$ and a set *E* of examples, the sum-of-squares error is

$$\mathbf{Error}_{\mathbf{E}(\overline{\mathbf{w}})} = \sum_{\mathbf{e}\in\mathbf{E}} \left(\mathbf{p}_{\mathbf{e}}^{\overline{\mathbf{w}}} - \mathbf{o}_{\mathbf{e}}\right)^2,$$

  - $\mathbf{p}_{\mathbf{e}}^{\overline{\mathbf{w}}}$ is the predicted output by a neural network with parameter values given by $\overline{\mathbf{w}}$ for example *e*
  - o_e is the observed output for example *e*.

- The aim of neural network learning is, given a set of examples, to find parameter settings that minimize the error.

Neural Network Learning

- Aim of neural network learning: given a set of examples, find parameter settings that minimize the error.
- Back-propagation learning is gradient descent search through the parameter space to minimize the sum-of-squares error.

Backpropagation Learning

- Inputs:
  - A network, including all units and their connections
  - Stopping Criteria
  - Learning Rate (constant of proportionality of gradient descent search)
  - Initial values for the parameters
  - A set of classified training data
- Output: Updated values for the parameters

Backpropagation Learning Algorithm

- Repeat
  - evaluate the network on each example given the current parameter settings
  - determine the derivative of the error for each parameter
  - change each parameter in proportion to its derivative
- until the stopping criteria is met

Gradient Descent for Neural Net Learning

- At each iteration, update parameter w_i

$$\mathbf{w_i} \leftarrow \left(\mathbf{w_i} - \eta \frac{\partial \mathbf{error(p_i)}}{\partial \mathbf{p_i}} \right)$$

  η is the learning rate
- You can compute partial derivative:
- numerically: for small Δ

$$\frac{\mathbf{error(p_i + \Delta) - error(p_i)}}{\Delta}$$

- analytically: $f'(x) = f(x)(1 - f(x))$ + chain rule

Simulation of Neural Net Learning

Para-meter	iteration 0 Value	iteration 0 Deriv	iteration 1 Value	iteration 80 Value
w_0	0.2	0.768	−0.18	−2.98
w_1	0.12	0.373	−0.07	6.88
w_2	0.112	0.425	−0.10	−2.10
w_3	0.22	0.0262	0.21	−5.25
w_4	0.23	0.0179	0.22	1.98
Error:	4.6121		4.6128	0.178

Neural Networks and Logic

- Meaning is attached to the input and output units.
- There is no a priori meaning associated with the hidden units.
- What the hidden units actually represent is something that's learned.

Case-based Reasoning

- Idea: experiences themselves are stored. These are called cases.
- Given a new example, the most appropriate case(s) in the knowledge base are found and these are used to predict properties of the new example.

Extremes of Case-based Reasoning

- The cases are simple and for each new example the agent has seen many identical instances. Use the statistics of the cases.
- The cases are simple but there are few exact matches.
  Use a distance metric to find the closest cases.
- The cases are complex, there are no matches. You need sophisticated reasoning to determine why an old case is like the new case.
  Examples: legal reasoning, case-based planning.

k-nearest Neighbors

- Need a distance metric between examples.
- Given a new example, find the *k* nearest neighbors of that example.
- Predict the classification by using the mode, median, or interpolating between the neighbors.
- Often want *k* > *1* because there can be errors in the case base and k = odd (avoid ties).

Euclidean Distance

- Define a metric for each dimension (convert the values to a numerical scale).
- The Euclidean distance between examples *x* and *y* is:

$$\mathbf{d(x,y) = \sqrt{\sum_{A} w_A (x_A - y_A)^2}}$$

  - x_A is the numerical value of attribute *A* for example *x*
  - w_A is a nonnegative real-valued parameter that specifies the relative weight of attribute *A*.

kd-tree

- Examples are stored at the leaves.
- The aim is to build a balanced tree; so a particular example can be found in log *n* time when there are *n* examples.
- Not all leaves will be an exact match for a new example.
- Any exact match can be found in $d = log\ n$ time
- All examples that miss on just one attribute can be found in $O(d^2)$ time.