CAS CS 640
Artificial Intelligence
Lectures 5, 6, and 7, 2024

Margrit Betke

Machine Learning

Learning by Training Neural Nets

Forms of Machine Learning

AI system "learns" if it improves its performance *based on observations & feedback from its environment*

Unsupervised learning (=clustering):

Input: vector of attributes (features). No explicit feedback.

Supervised learning:

Input: vector of attributes (= features). Feedback = output of continuous or discrete value(s) = labels of input examples.

Reinforcement Learning:

Actions are rewarded or punished.

Russell and Norvig

Supervised Learning

Training set = N example input-output pairs

(x_1,y_1), (x_2,y_2), ..., (x_N,y_N)

where each y_j was generated by an unknown function f, such that $f(x) = y$. Function f needs to be learned.

- The AI system designer finds a function h that approximates f. The designer trains, e.g., a neural net that computes $h(x_j)=y_j$ for all examples in the training set.
- There are no guarantees that, for new inputs, $h(x_{new}) \approx f(x_{new})$.
- To measure accuracy (Is $h \approx f$?), we use a test set of labeled examples = input-output pairs (≠ training set!):

A neural net is trained well if $h(x_{test}) \approx y_{test}$ for all test example pairs (x_{test}, y_{test}).

Russell and Norvig

Example of Supervised Learning Task and Solution

Task: Predict if a Titanic passenger survived

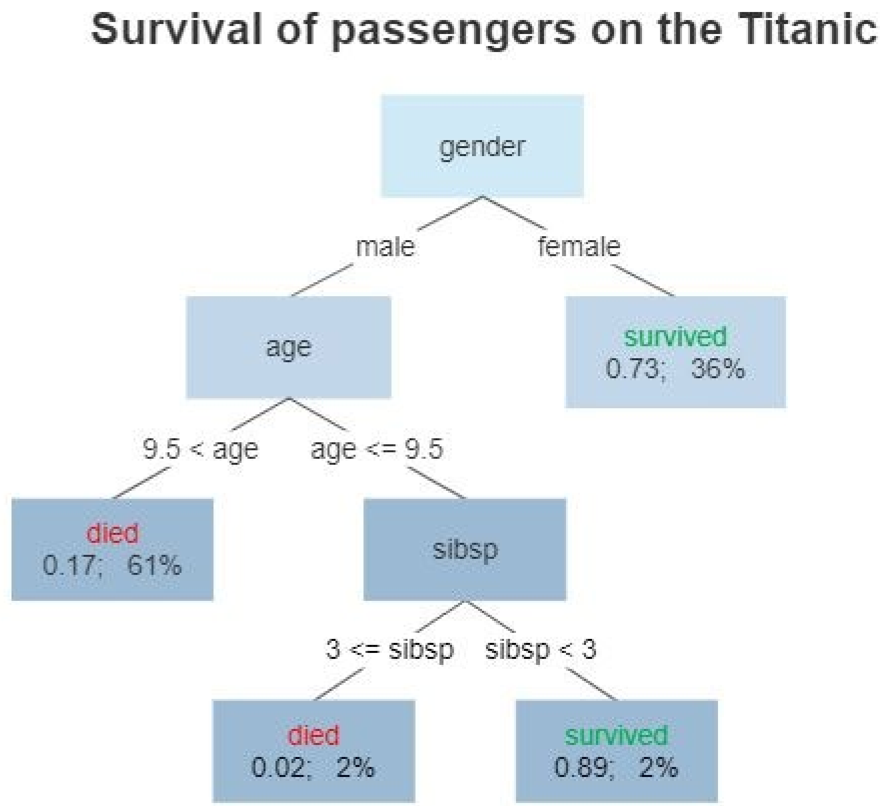
Training set = Example pairs (x_i, y_i) = (Attributes of person i, Survived? Yes/No)

3 attributes: gender, age, number of siblings/spouse on board (sibsp)

Model: Decision Tree

Non-leaf nodes: decisions on features

Leaf nodes = output label y



Wikipedia link

"Handcrafted tree"
Automated methods -> CS 542

Forms of Machine Learning

AI system "learns" if it improves its performance *based on observations & feedback from its environment*

Unsupervised learning (=clustering):

Input: vector of attributes (features). No explicit feedback.

Supervised learning:

Input: vector of attributes (= features). Feedback = output of continuous or discrete value(s) = labels of input examples.

Reinforcement Learning:

Actions are rewarded or punished.

Russell and Norvig

CAS CS 640

Artificial Intelligence

Lectures by Margrit Betke

Learning by Training Neural Nets

Adopted from P. Winston, Artificial Intelligence, 1992

CAS CS 640

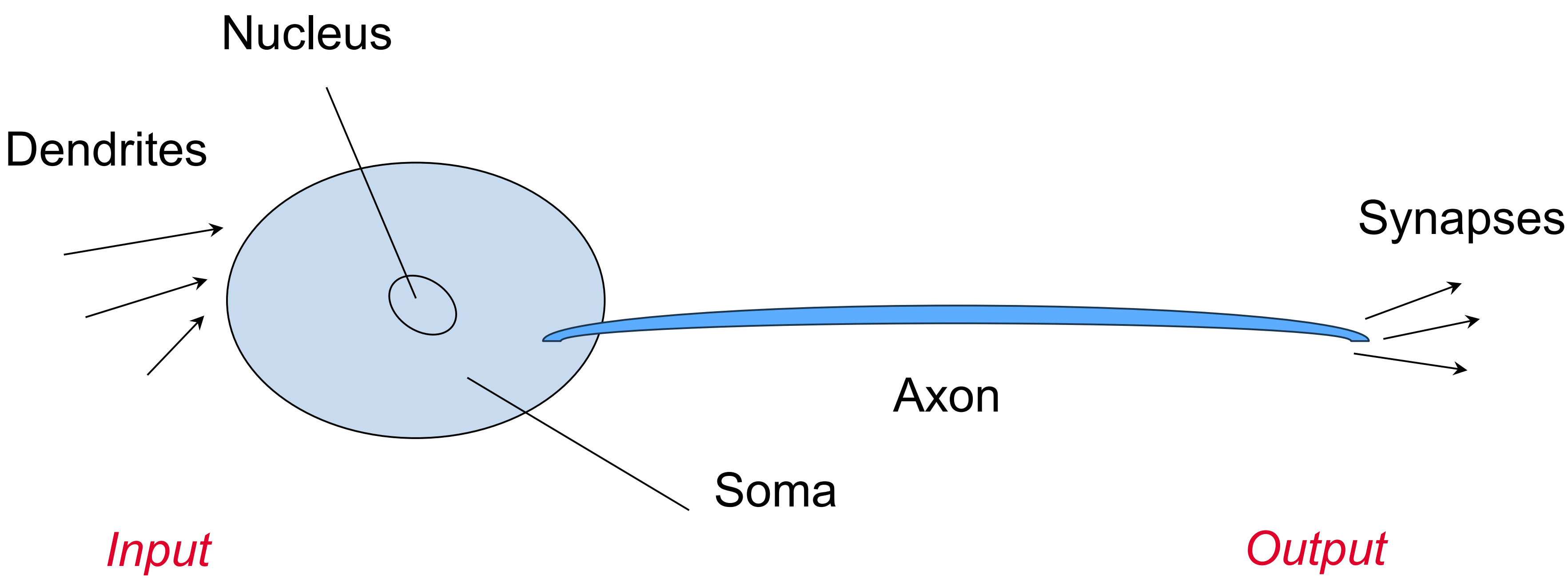
Artificial Intelligence

Lectures by Margrit Betke

Learning by Training Neural Nets, Part 1

Adopted from P. Winston, Artificial Intelligence, 1992

Real Neuron



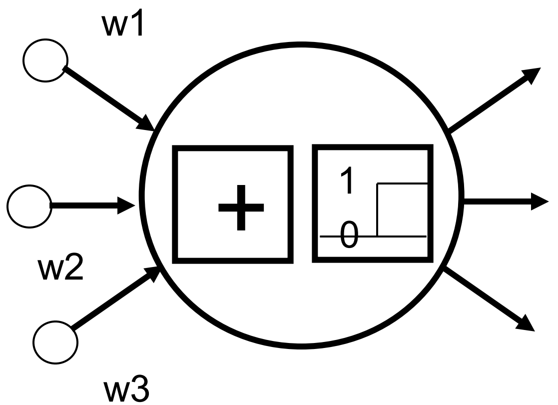
When collective input at dendrites reaches threshold, pulse travels down axon, causes excitation or inhibition of next neuron.

Real Neural Nets

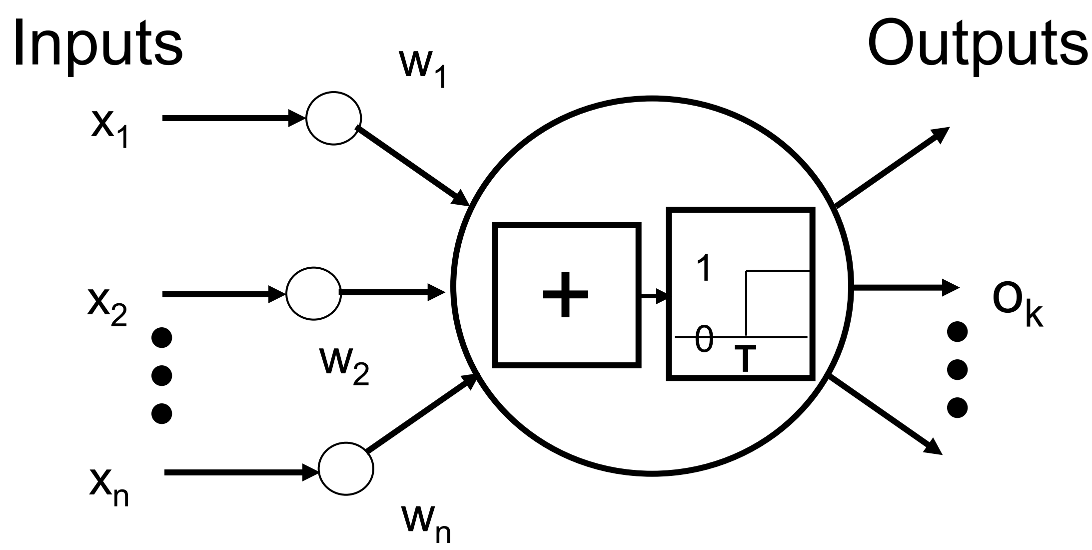
- Number of neurons in human brain: $\sim 10^{11}$
- Synapses per neuron in cerebellum (motor control): $\sim 10^5$
- Synapses per brain: $\sim 10^{16}$
  - Approximately equivalent to 300 times the characters in all books of US Library of Congress

Simulated Neural Nets

- NN consists of neurons or nodes
- NN have links simulating axon-synapse-dendrite connections
- Each link has weight. Like synapse, weight determines nature & strength of connection:
  - Large positive weight → strong excitation
  - Small negative weight → weak inhibition
- Like dendritic mechanisms, the activation function combines input: threshold function sums input values and passes them through threshold; output is 0 or 1.



Simulated Neuron



Activation function sums n products of input x_i and weight w_i and compares result to threshold T:

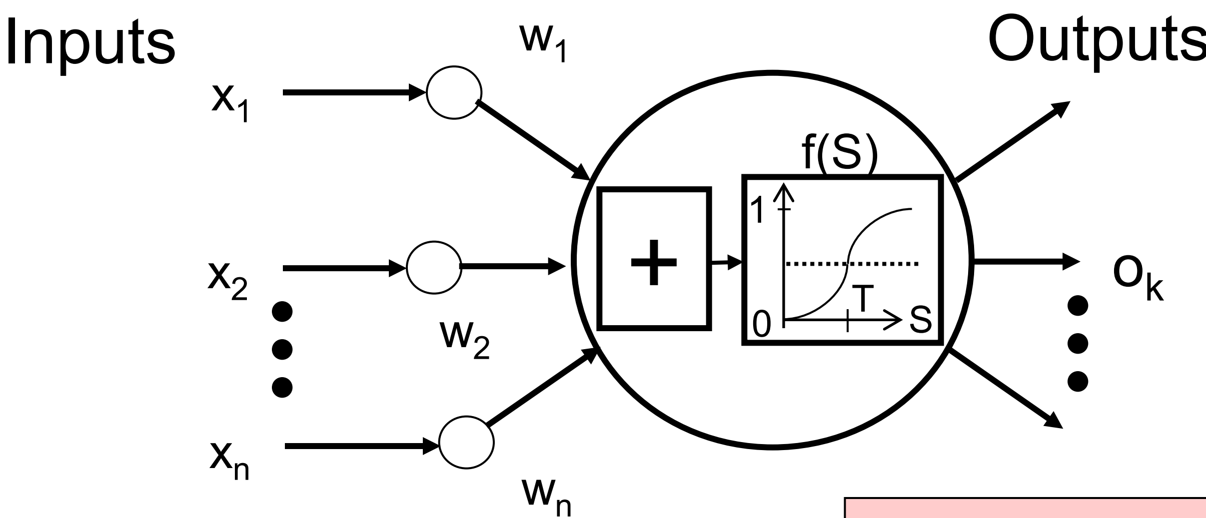
"Fire" if result >= T

$$o_k = 1 \text{ if } \sum_{i=1}^{n} x_i w_i >= T$$

$$o_k = 0 \text{ else}$$

Simulated Neuron: Trainable Node

differentiable f



Activation function sums n products of input x_i and weight w_i, passes result S into function f, and outputs f(S).

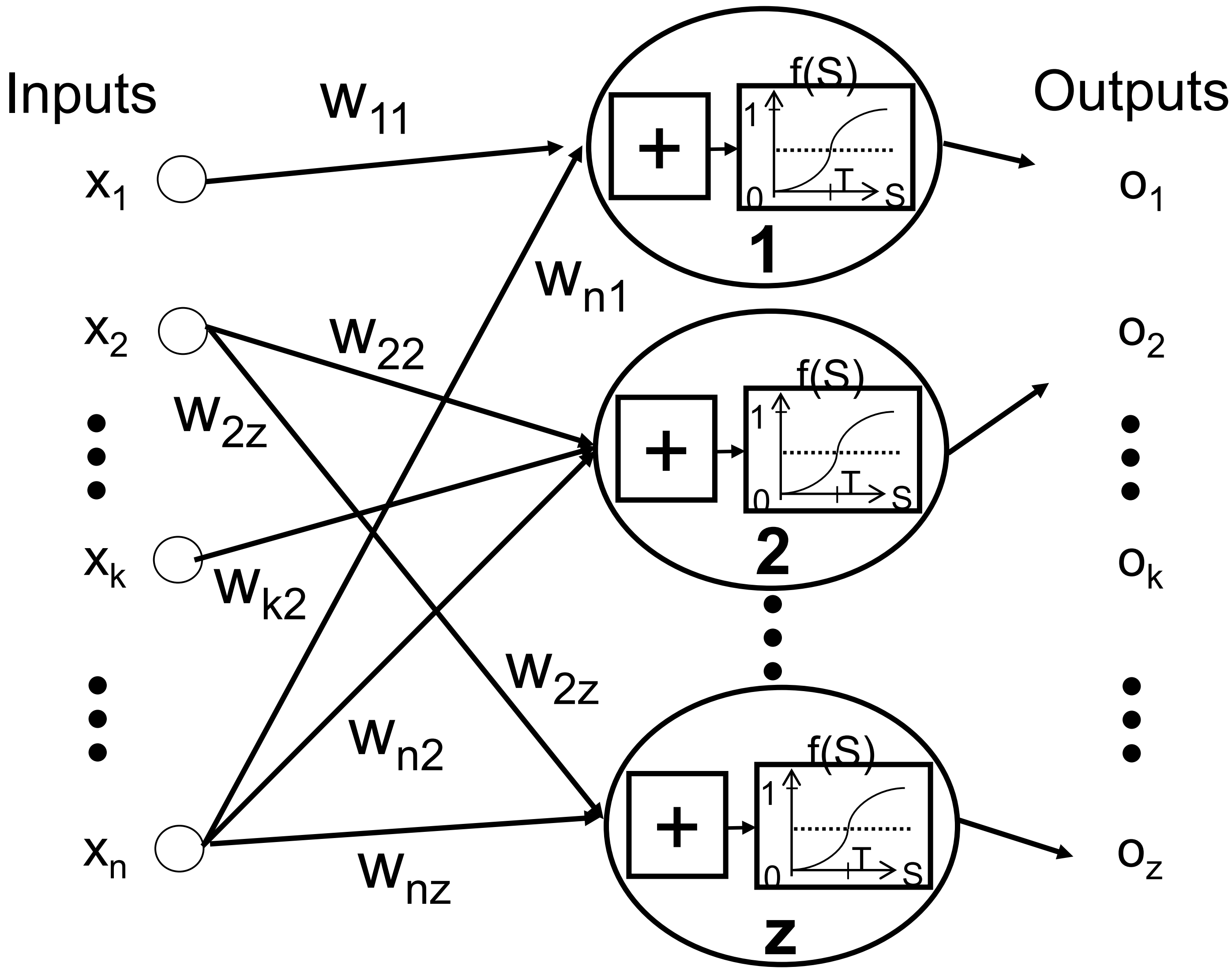
If f(S) above threshold T, output >0.5, otherwise <0.5.

$$S = \sum_{i=1}^{n} x_i w_i$$

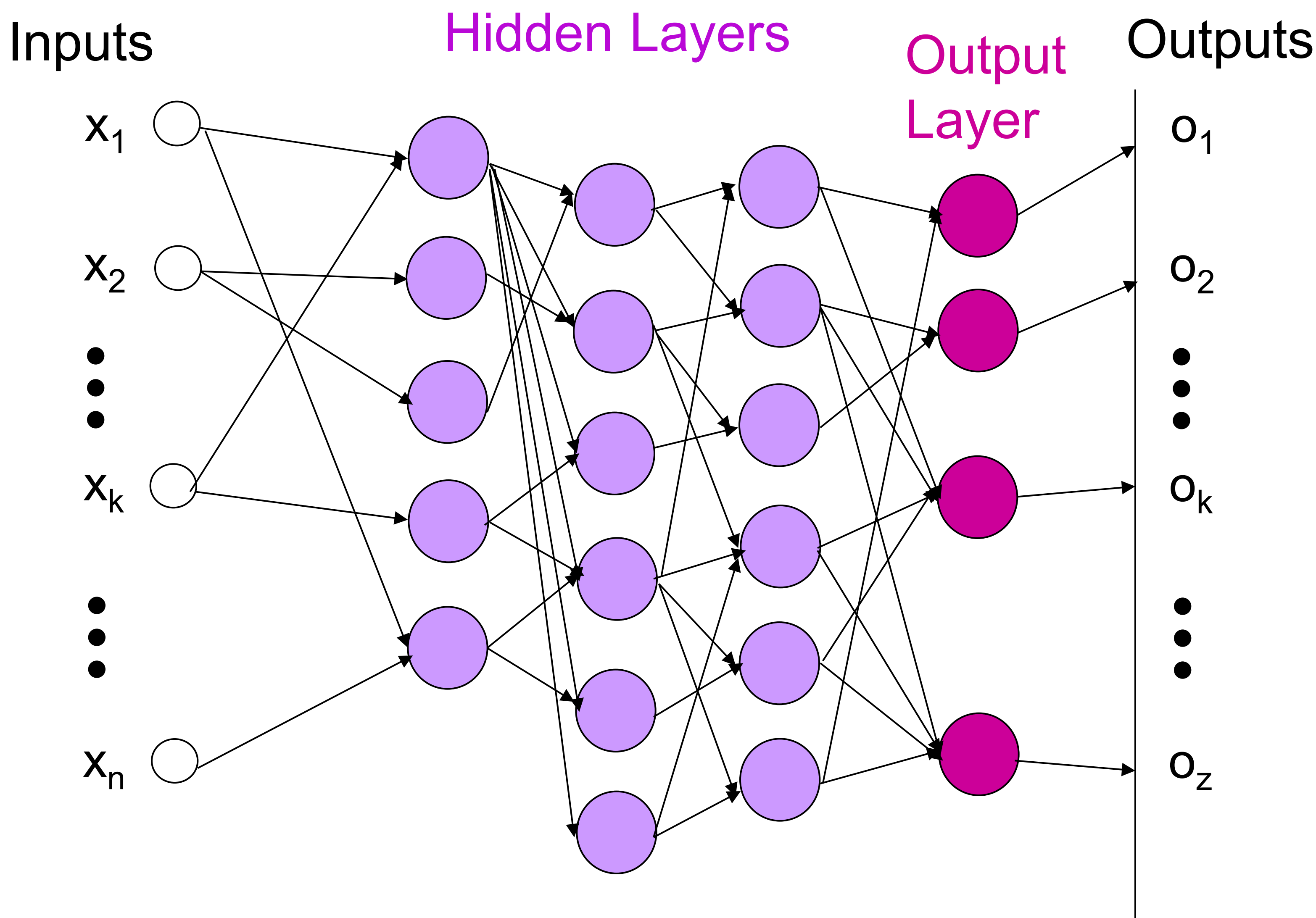
$$o_k = f(S)$$

if $f(S) >= T$
$o_k >= 0$

2-Layer Neural Networks



Multilayer Neural Networks



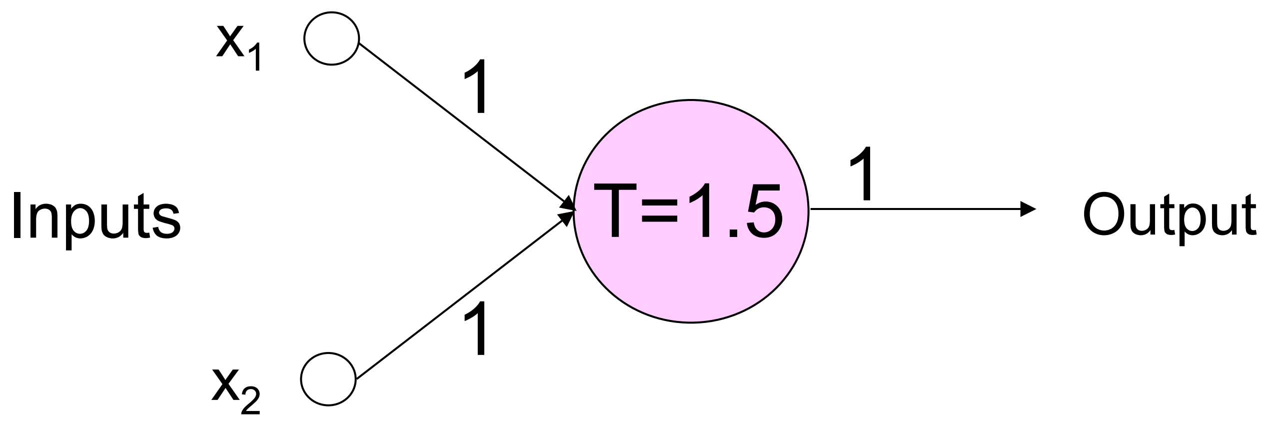
Tasks for Neural Networks

- Evaluation Problem
- Training Problem

Single Node Neural Net

$$o_k = 1 \text{ if } \sum_{i=1}^{n} x_i w_i >= T$$

$$o_k = 0 \text{ else}$$



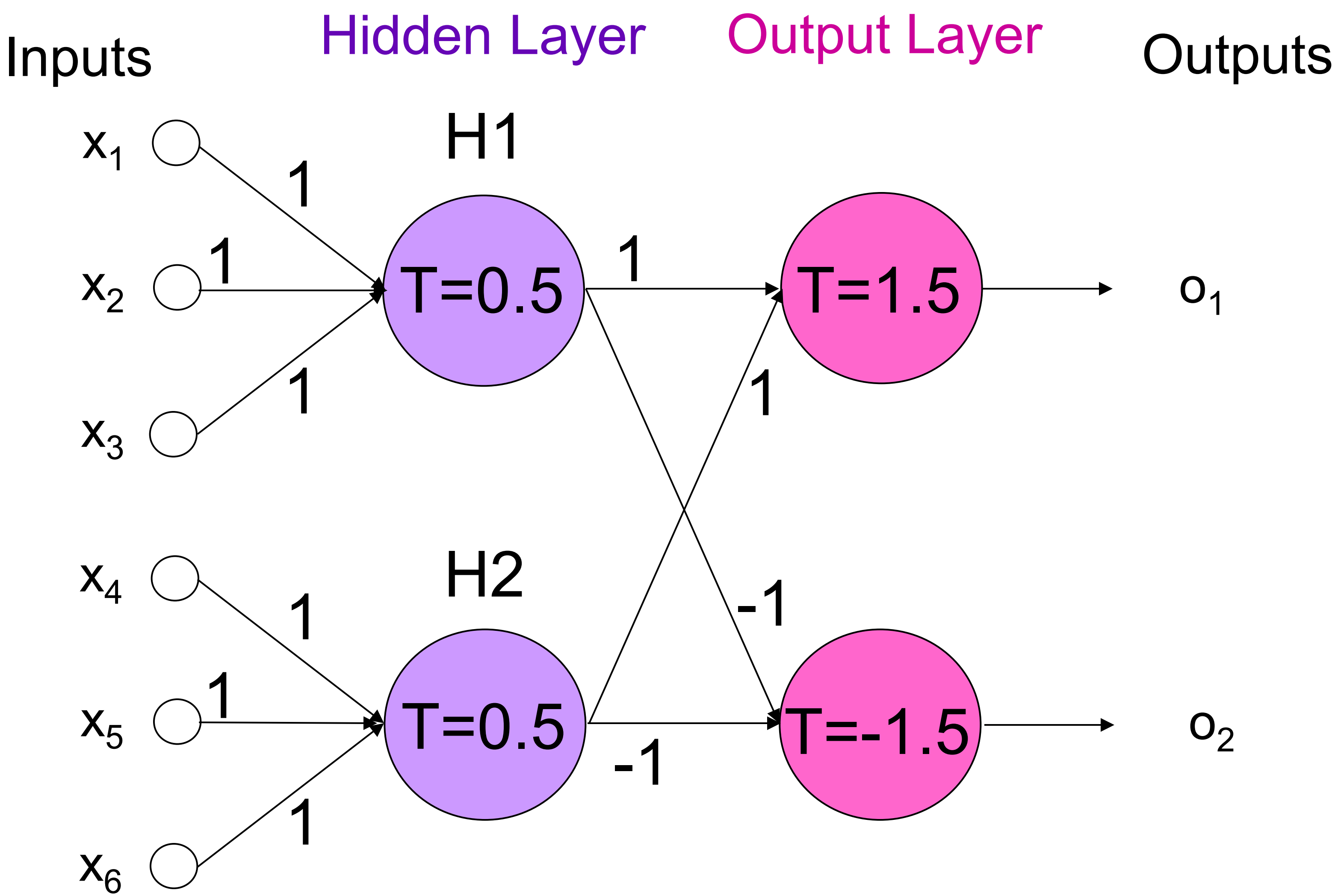
x_1	x_2	Computation	Output
0	0	0(1)+0(1)=0<1.5	0
0	1	0(1)+1(1)=1<1.5	0
1	0	1(1)+0(1)=1<1.5	0
1	1	1(1)+1(1)=2>1.5	1

How can we create a NN to recognize $Or(x_1,x_2)$?

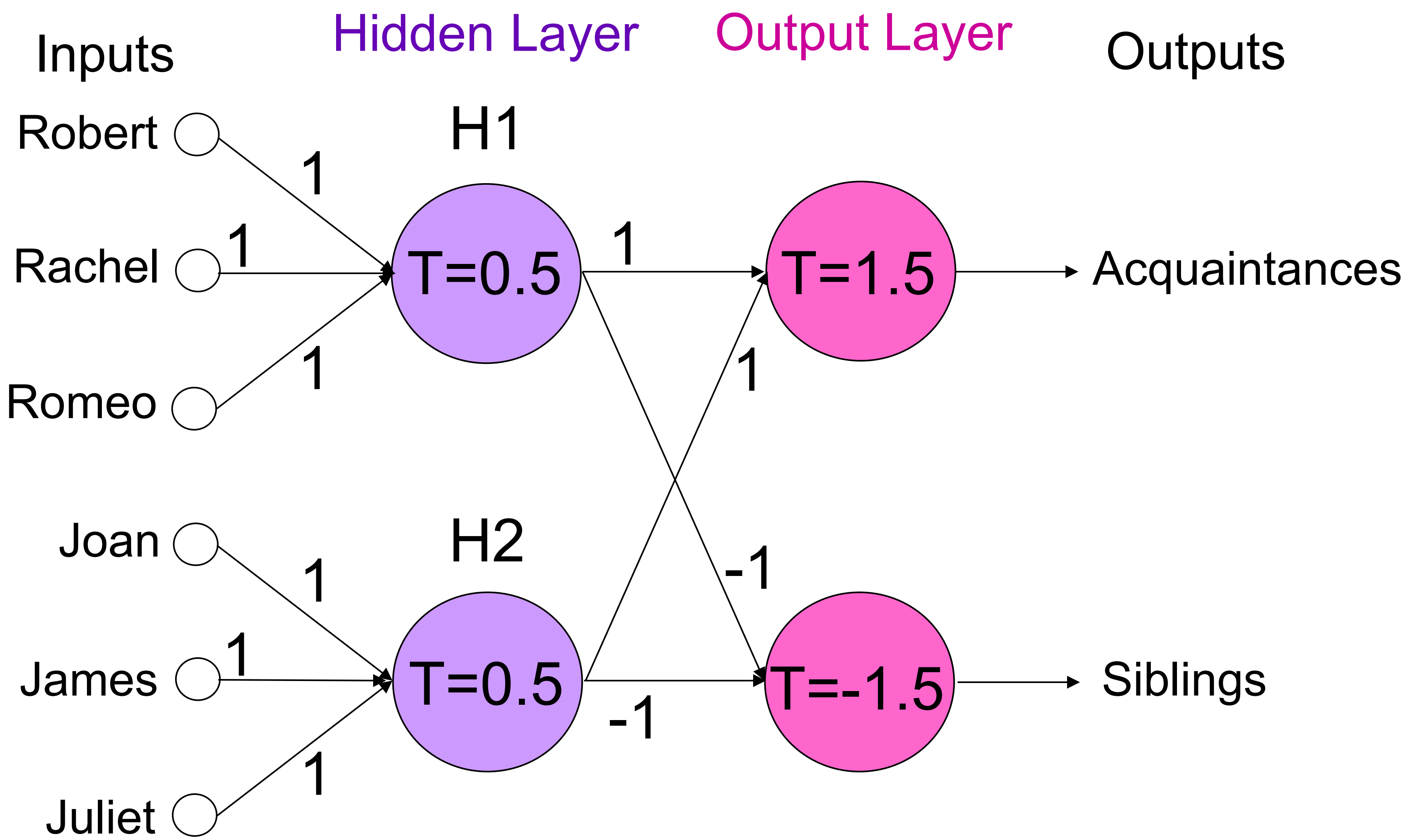
x_1	x_2	Computation	Output
0	0	$0(1)+0(1)=0 < T$	0
0	1	$0(1)+1(1)=1 > T$	1
1	0	$1(1)+0(1)=1 > T$	1
1	1	$1(1)+1(1)=2 > T$	1

T ?

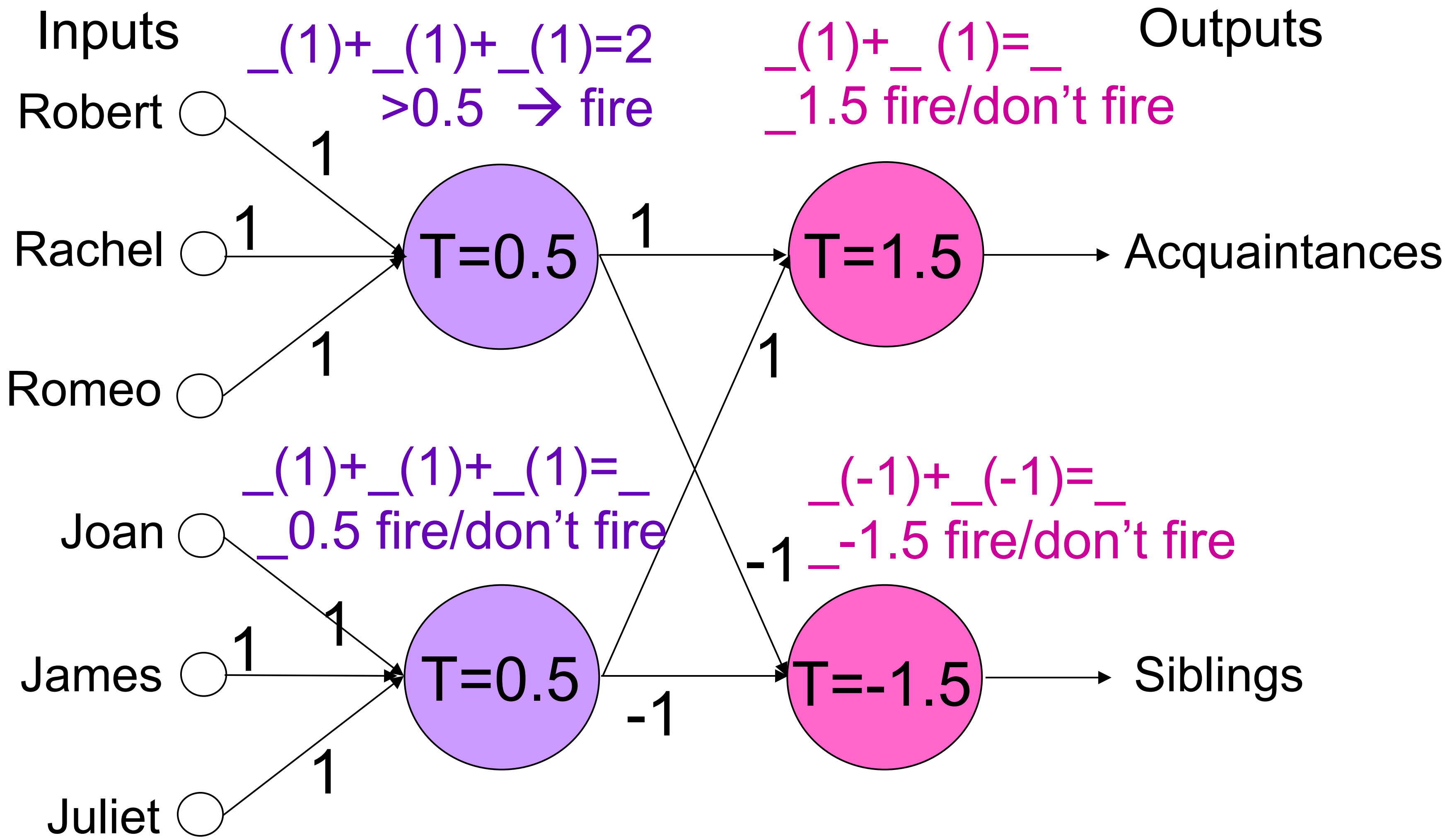
Example of a 3-layer Net



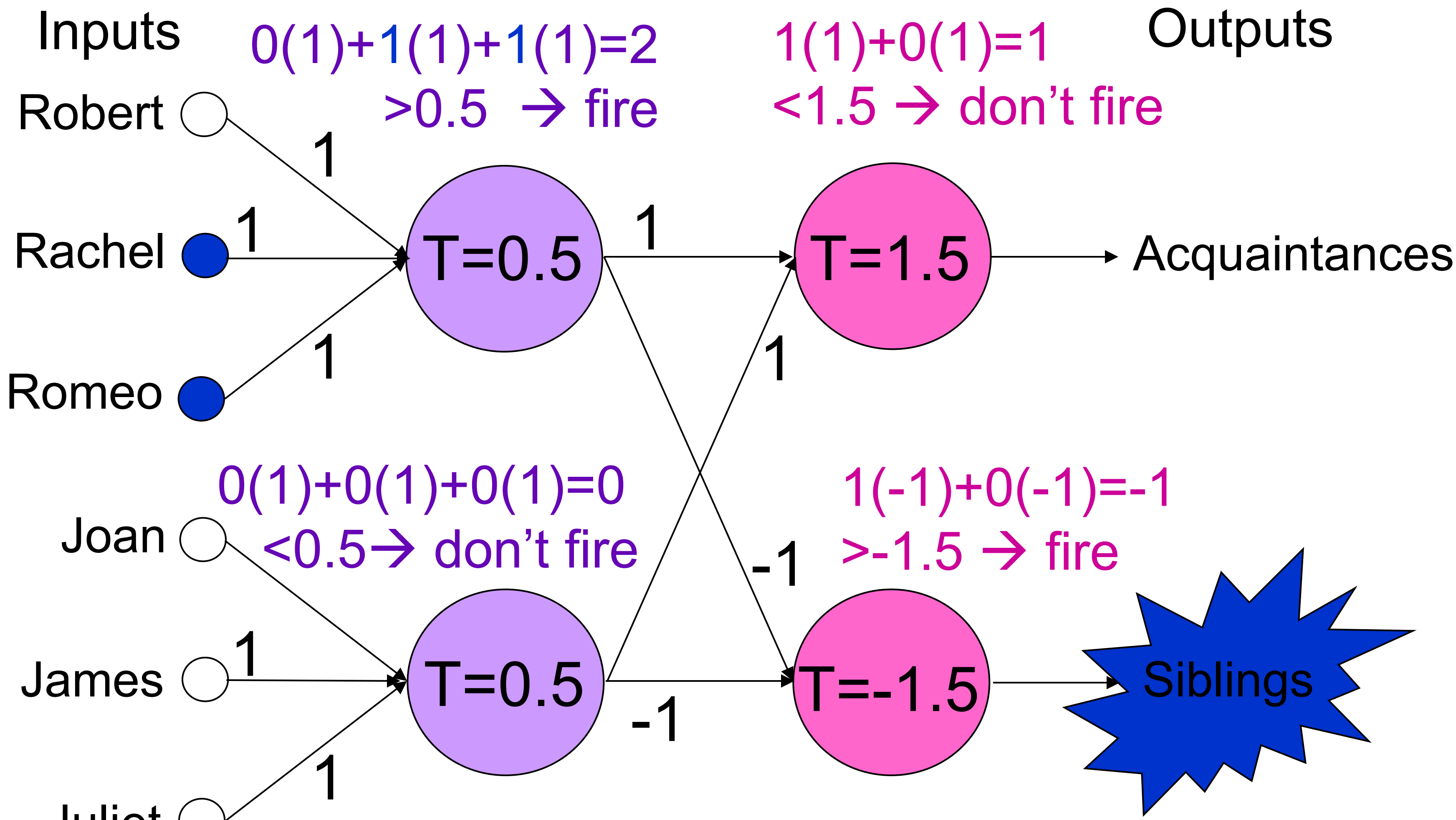
Acquaintance or Sibling Net



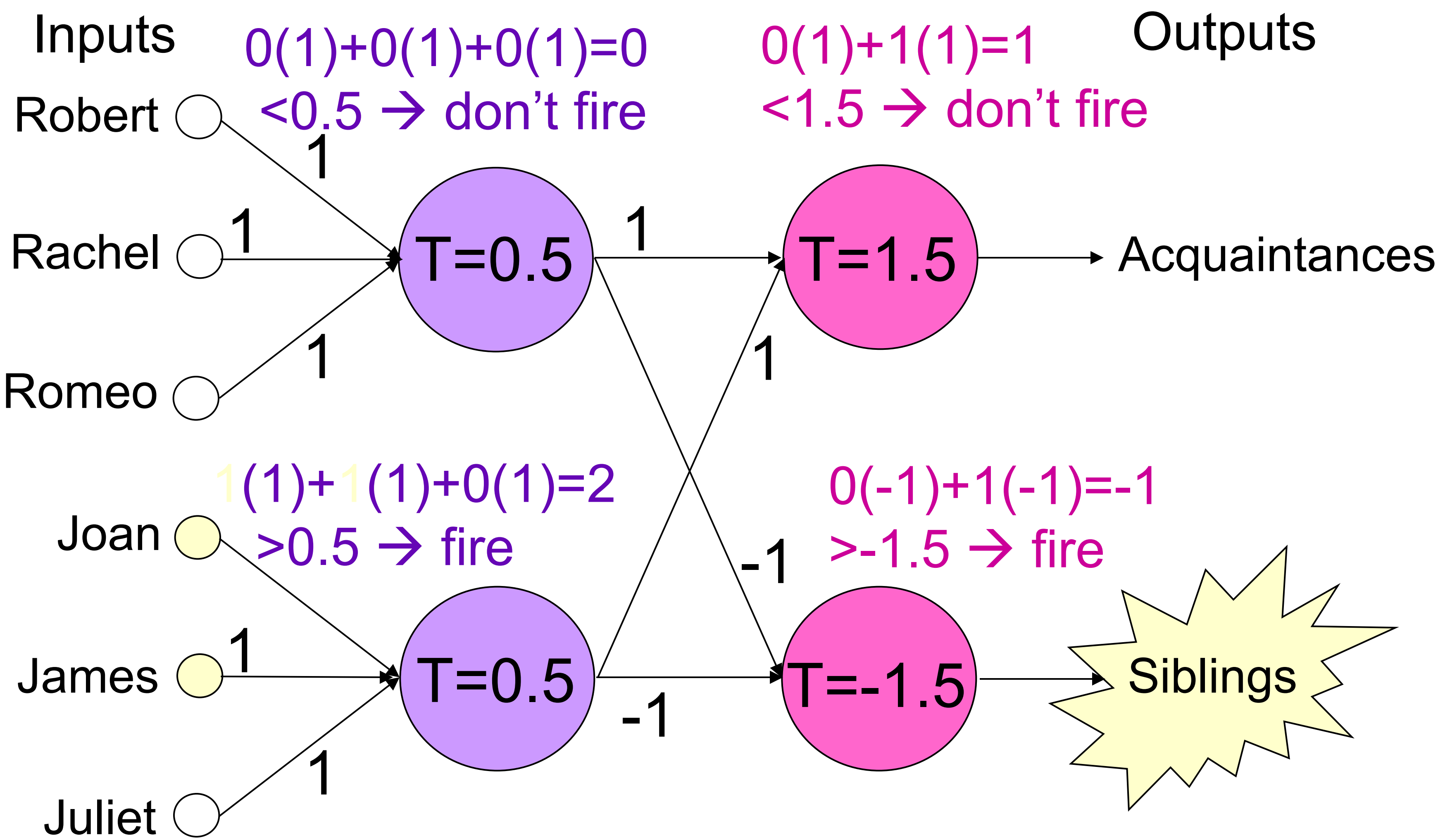
Acquaintances or Siblings?



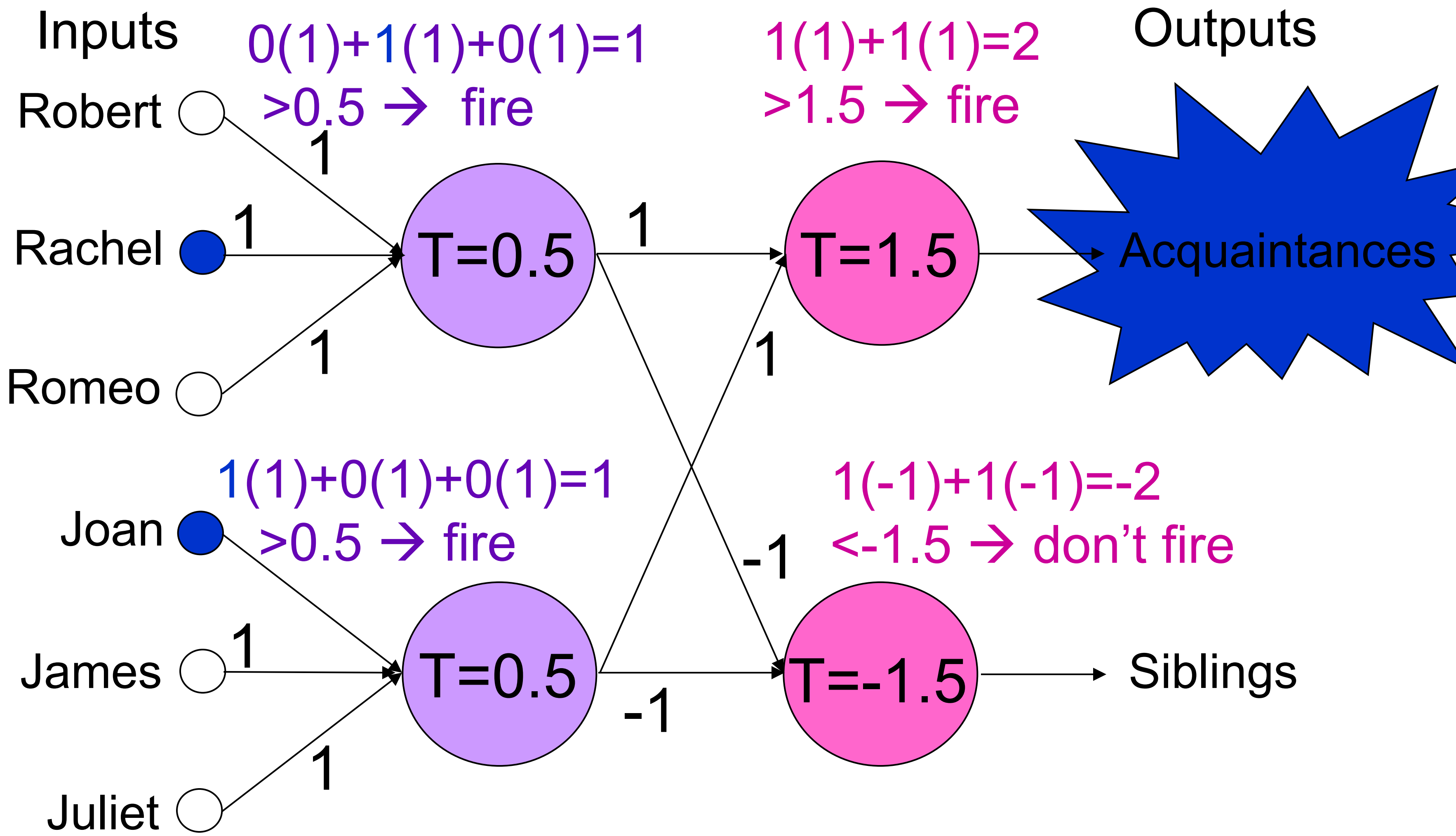
Acquaintances or Siblings?



Acquaintances or Siblings?



Acquaintances or Siblings?



Tasks for Neural Networks

- Evaluation Problem

  Given a neural net N and input vector X:

  Does N recognize X, i.e., $N(X) = 1$?

  Solution: Compute output of nodes, layer by layer

  Examples: "And" network
  "Or" network
  "Acquaintances/Siblings" network

Tasks for Neural Networks

- **Evaluation Problem**

  Given a neural net N and input vector X:

  Does N recognize X, i.e., $N(X) = 1$?

- **Training Problem**

  Given training set $X_{training}$ of input vectors:

  Find neural net N that recognizes inputs in

  training set $X_{training}$ and

  test set X_{test}.

Training Problem – Version 1

- Training set $X_{training} = (X_{positive}, X_{negative})$ of input vectors is given
- Number of nodes is given
- Number of layers is given
- Shape of activation function is given
- Links are given

Goal: Learn weights and thresholds, such that

$$N(X_{positive,\, i}) = 1 \text{ and } N(X_{negative,\, j}) = 0$$

for all inputs i in $X_{positive}$ and j in $X_{negative}$

Training Problem – Version 2

You, the designer of the AI system, must determine:

- Number of nodes
- Number of layers
- Shape of each activation function and its threshold
- Links between nodes
- Weights on connections
- Representative set $X_{training}$

Goal: $N(X_{positive,\, i}) = 1$ and $N(X_{negative,\, j}) = 0$

where $X_{positive,\, i}$ and $X_{negative,\, j}$ in X_{test}

Training/Design Problem – Version 2

You, the designer of the AI system, must determine:

- Number of nodes
- Number of layers
- Shape of each activation function and its threshold
- Links between nodes
- Weights on connections
- Representative set $X_{training}$

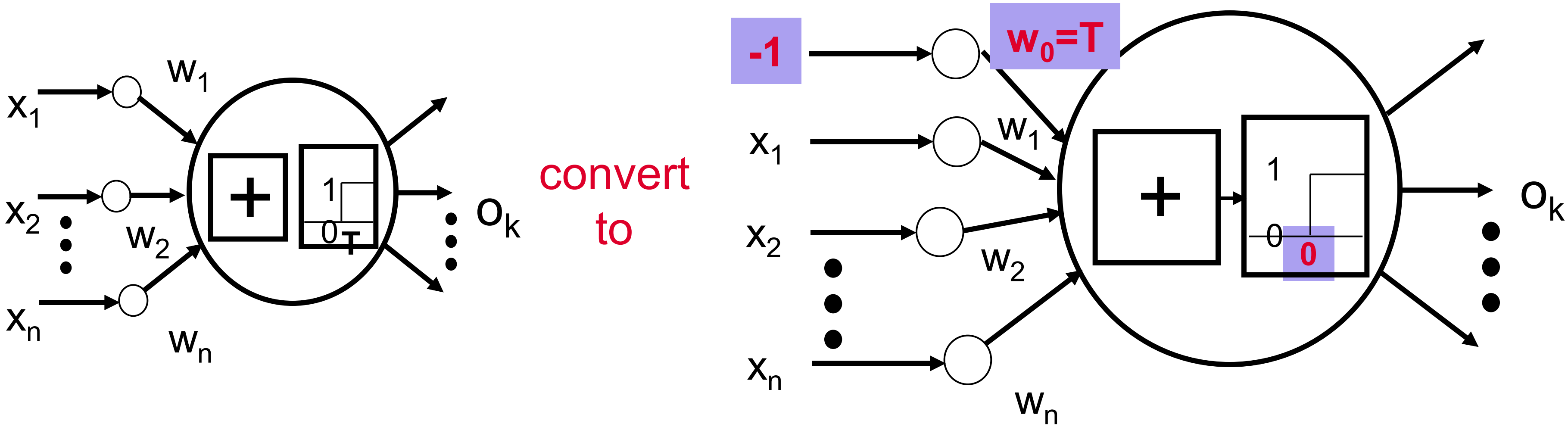
Network Architecture

Goal: $N(X_{positive,\ i}) = 1$ and $N(X_{negative,\ j}) = 0$

where $X_{positive,\ i}$ and $X_{negative,\ j}$ in X_{test}

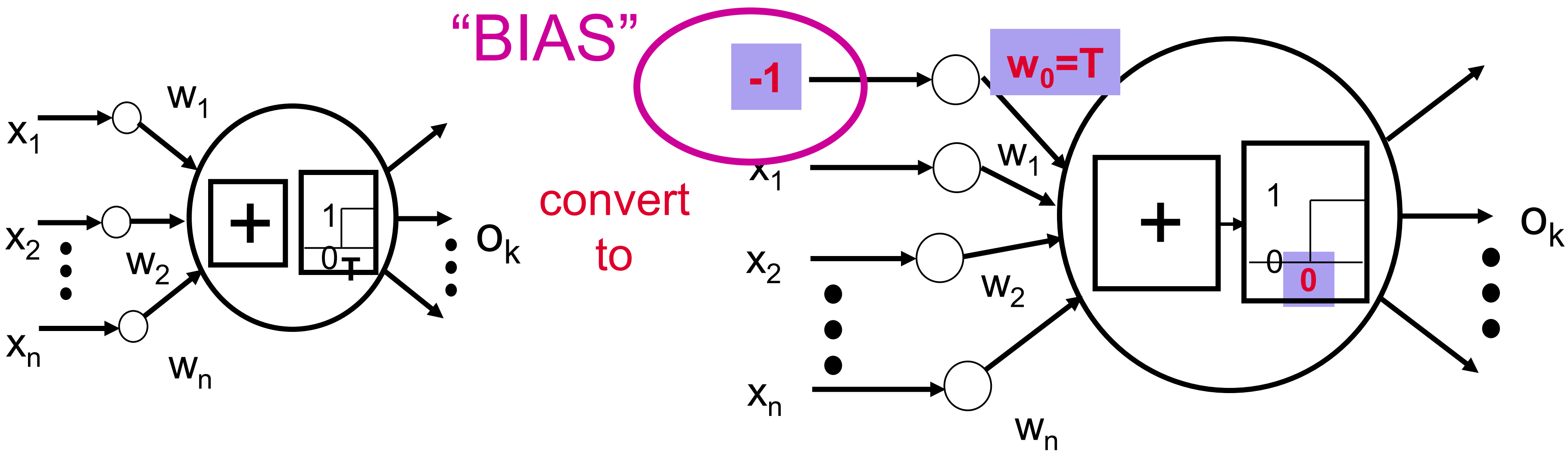
Training Problem – Version 1

- Assume number of nodes, number of layers, shape of activation function, and links are given.
- Task: Learn weights and thresholds.
- 1st step: Convert thresholds into weights and avoid having to learn two kinds of parameters:



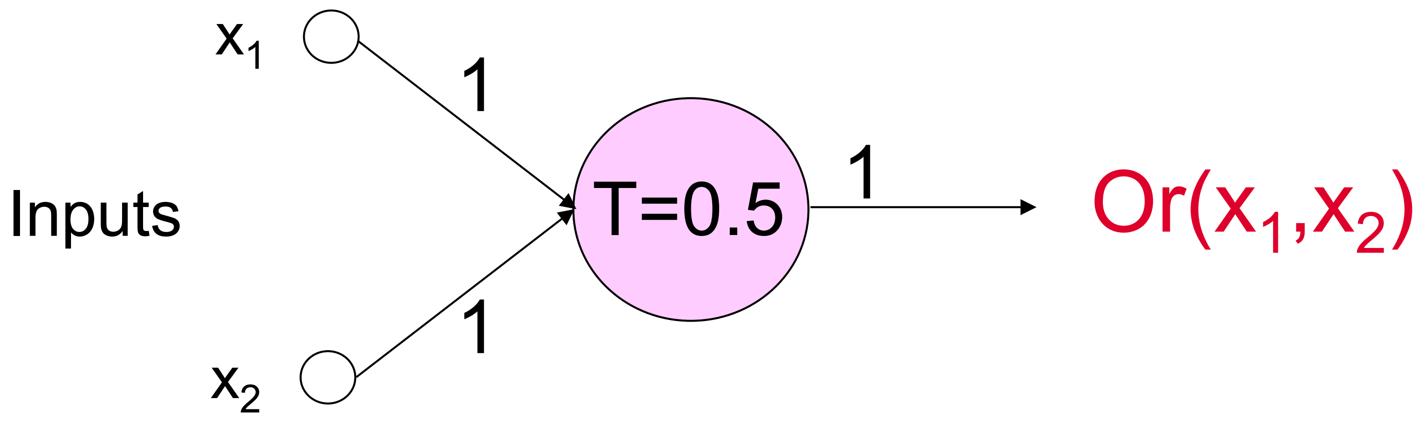
Training Problem – Version 1

- Assume number of nodes, number of layers, shape of activation function, and links are given.
- Task: Learn weights and thresholds.
- 1st step: Convert thresholds into weights and avoid having to learn two kinds of parameters:



Single Node Neural Net

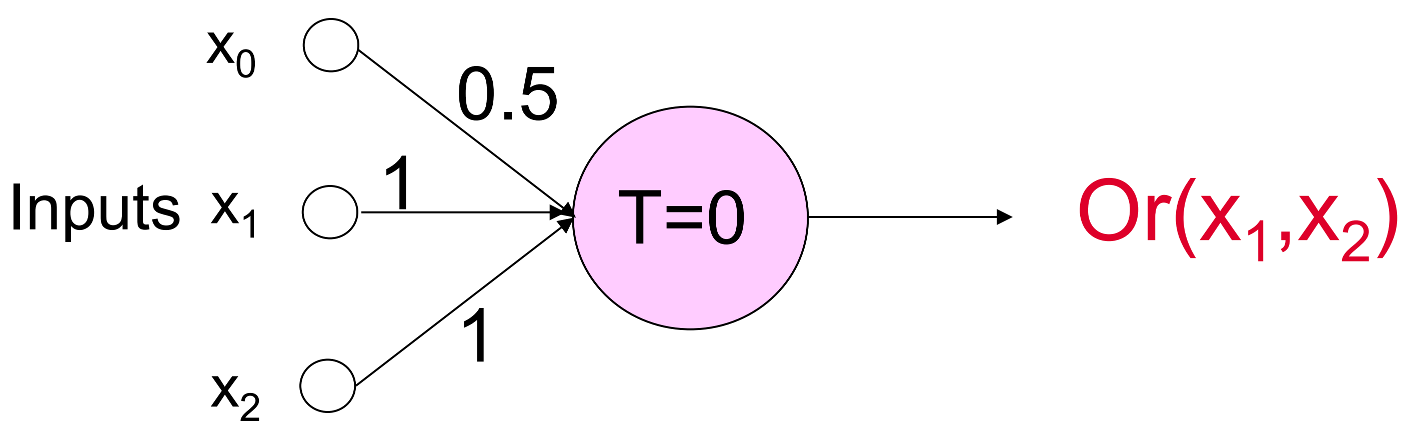
x_1	x_2	Computation	Output
0	0	$0(1)+0(1)=0 < 0.5$	0
0	1	$0(1)+1(1)=1 > 0.5$	1
1	0	$1(1)+0(1)=1 > 0.5$	1
1	1	$1(1)+1(1)=2 > 0.5$	1



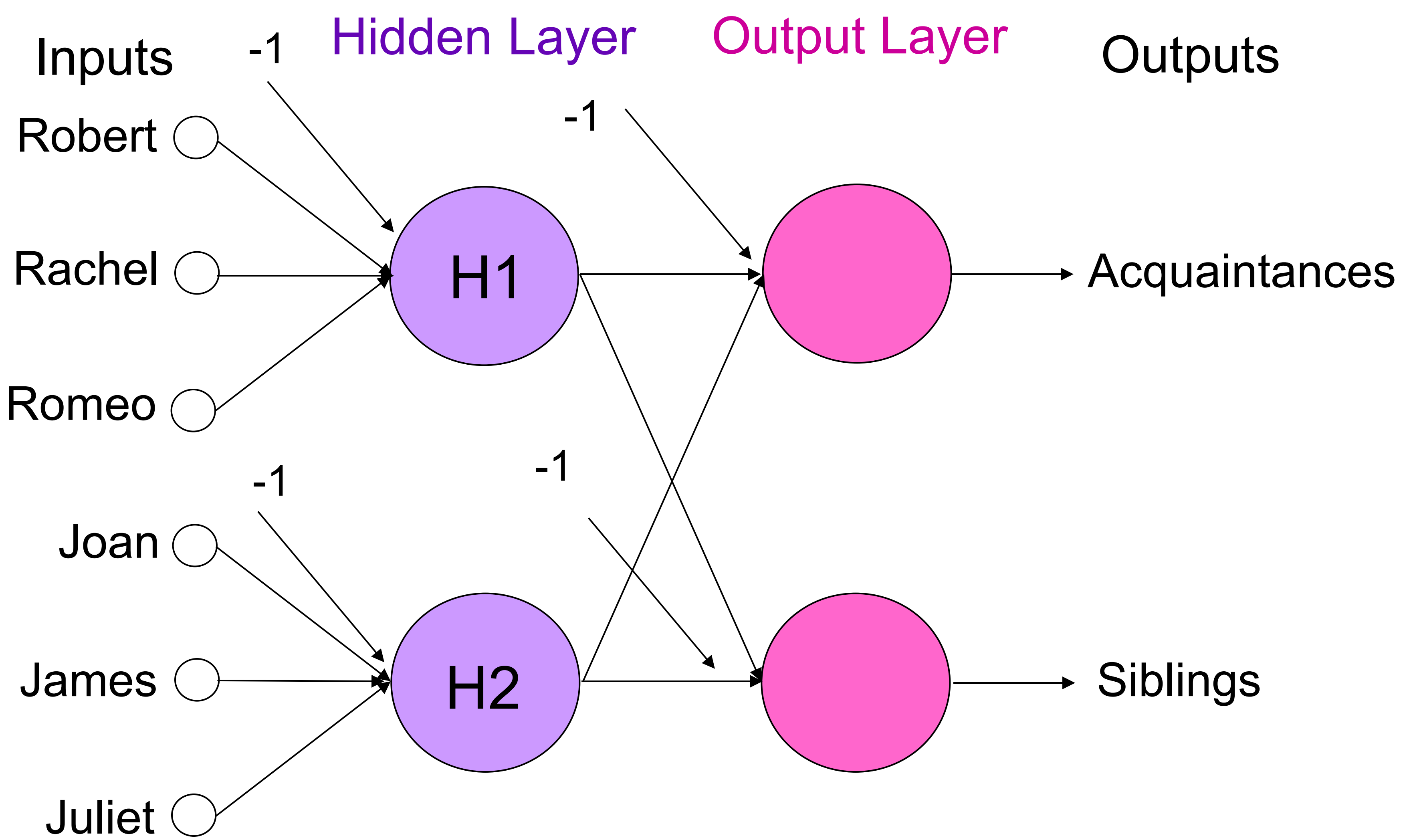
Example:
Converting Threshold to Weight

"BIAS"

x_0	x_1	x_2	Computation	Output
-1	0	0	-1(.5)+ 0(1)+0(1)=-0.5 < 0	0
-1	0	1	-1 (.5)+0(1)+1(1)=0.5 > 0	1
-1	1	0	-1 (.5)+1(1)+0(1)=0.5 > 0	1
-1	1	1	-1 (.5)+1(1)+1(1)=1.5> 0	1



Acquaintance or Sibling Net with converted thresholds



Training Problem – Version 1

Example:

Acquaintance/Sibling Network

After thresholds were converted to weights, the only parameters to learn are the weights.

Solution procedure: Backpropagation

Training Neural Networks

- Version 1:

  Given a neural net N, $X_{training}$, $X_{validation}$, and $X_{testing}$:

  Find weights of neural net

Solution: Backpropagation procedure

- Version 2:

  Given dataset X:

  1. Find neural network architecture
  2. Design training protocol with $X_{training}$, $X_{validation}$, and $X_{testing}$
  3. Run Backpropagation procedure

Training Neural Networks

- Version 1:

  Easier Problem!

  Given a neural net N, $X_{training}$, $X_{validation}$, and $X_{testing}$:

  Find weights of neural net

Solution: Backpropagation procedure

- Version 2:

  Given dataset X:

  1. Find neural network architecture
  2. Design training protocol with $X_{training}$, $X_{validation}$, and $X_{testing}$
  3. Run Backpropagation procedure

Training Neural Networks

- Version 1:

  Given a neural net N, $X_{training}$, $X_{validation}$, and $X_{testing}$:

  Find weights of neural net

Solution: Backpropagation procedure

- Version 2:

  Given dataset X:

  Toolbox or Research!

  1. Find neural network architecture
  2. Design training protocol with $X_{training}$, $X_{validation}$, and $X_{testing}$
  3. Run Backpropagation procedure

Training Neural Networks

- Version 1:

  Given a neural net N, $X_{training}$, $X_{validation}$, and $X_{testing}$:

  Find weights of neural net

Solution: Backpropagation procedure

NEXT TOPIC!!!

- Version 2:

  Given dataset X:

  1. Find neural network architecture
  2. Design training protocol with $X_{training}$, $X_{validation}$, and $X_{testing}$
  3. Run Backpropagation procedure

Part 1 Learning Objectives: Be able to

- Define machine learning terms such as supervised learning, decision trees, neural networks, activation function, hidden layer, output layer
- Explain the similarity of simulated neurons to real neurons
- Solve the evaluation (inference) problem for a neural net
- Design nodes to compute functions (AND, OR)
- Convert node thresholds into weights
- Explain the two versions of the training/design problem