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 ($\neq$ 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

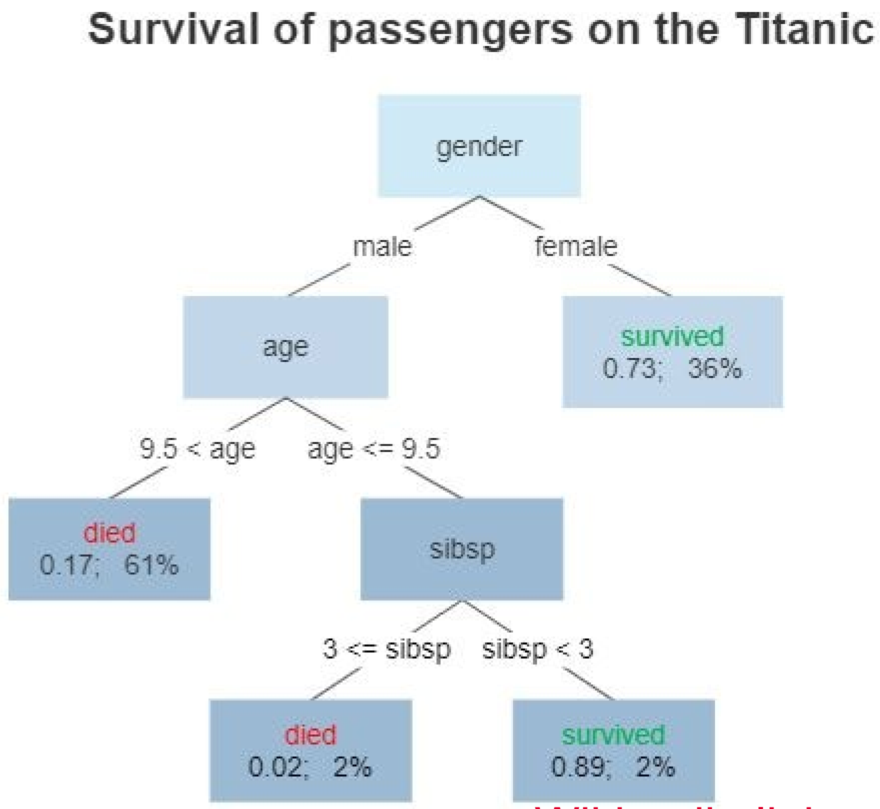
Training set = Example pairs (x_i, y_i) = (Attributes of person i, Survived? Yes/No)

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

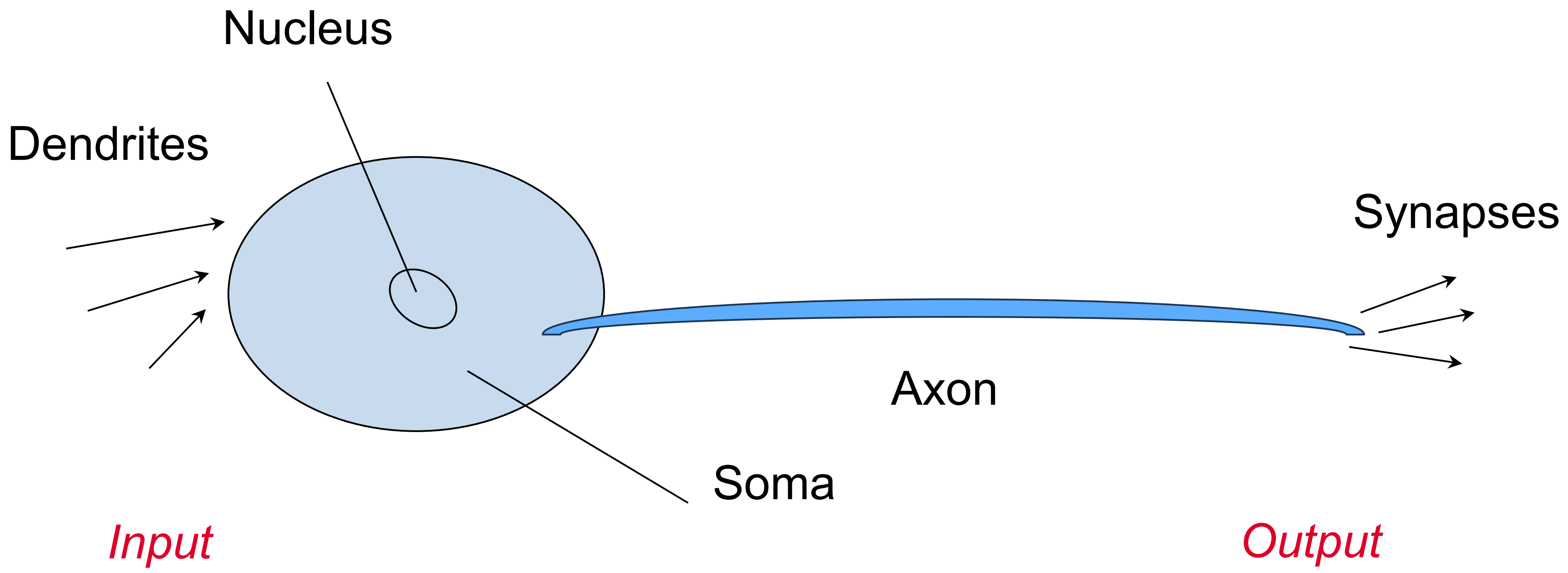
CAS CS 640
Artificial Intelligence

Lectures by Margrit Betke

Learning by Training Neural Nets, Part 1

Adopted from P. Winston, Artificial Intelligence, 1992

Real Neuron



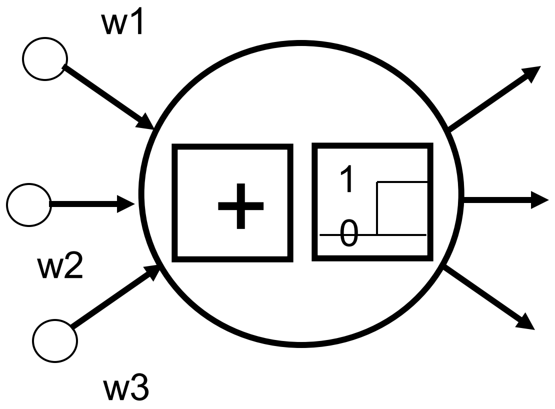
When collective input at dendrites reaches threshold, pulse travels down axon, causes excitation or inhibition of next neuron.

Real Neural Nets

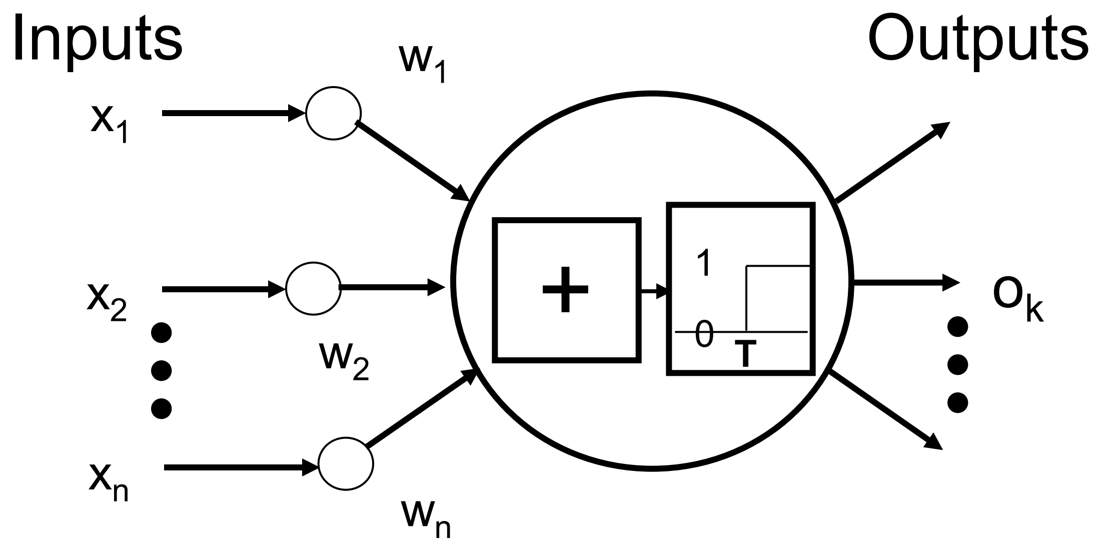
- Number of neurons in human brain: ~10^{11}
- Synapses per neuron in cerebellum (motor control): ~10^{5}
- Synapses per brain: ~10^{16}
  - Approximately equivalent to 300 times the characters in all books of US Library of Congress

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



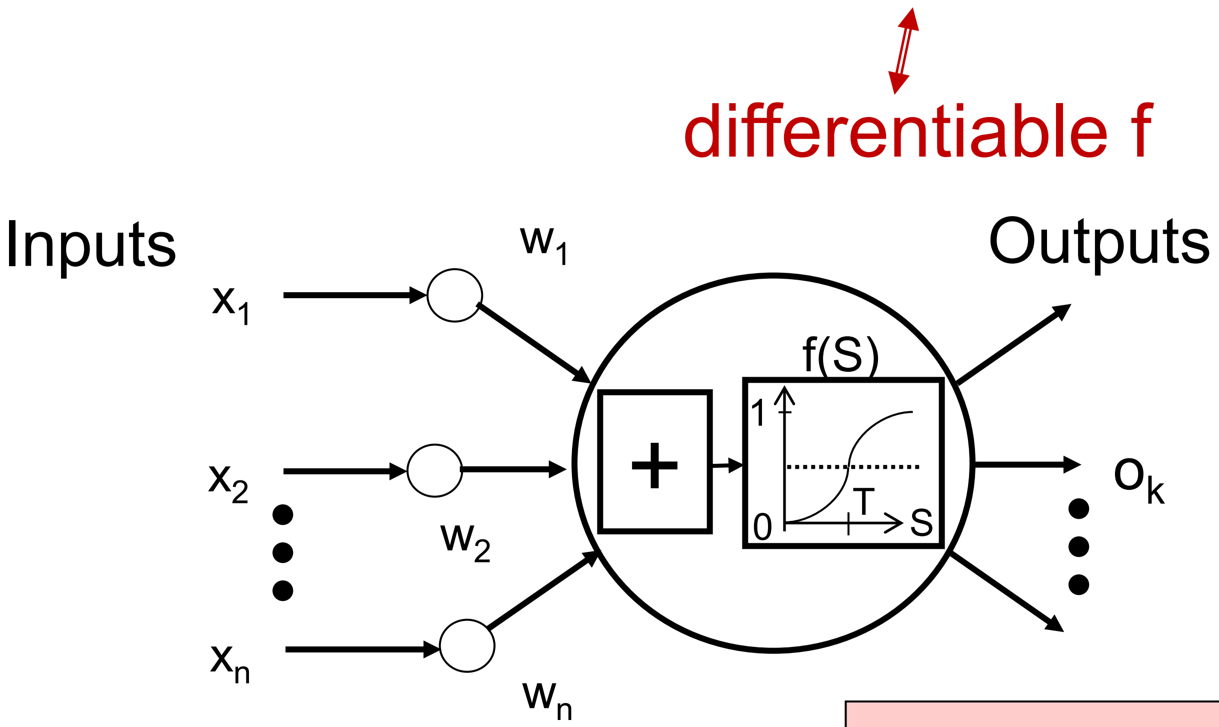
Activation function sums n products of input x_i and weight w_i and compares result to threshold T:

"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



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.

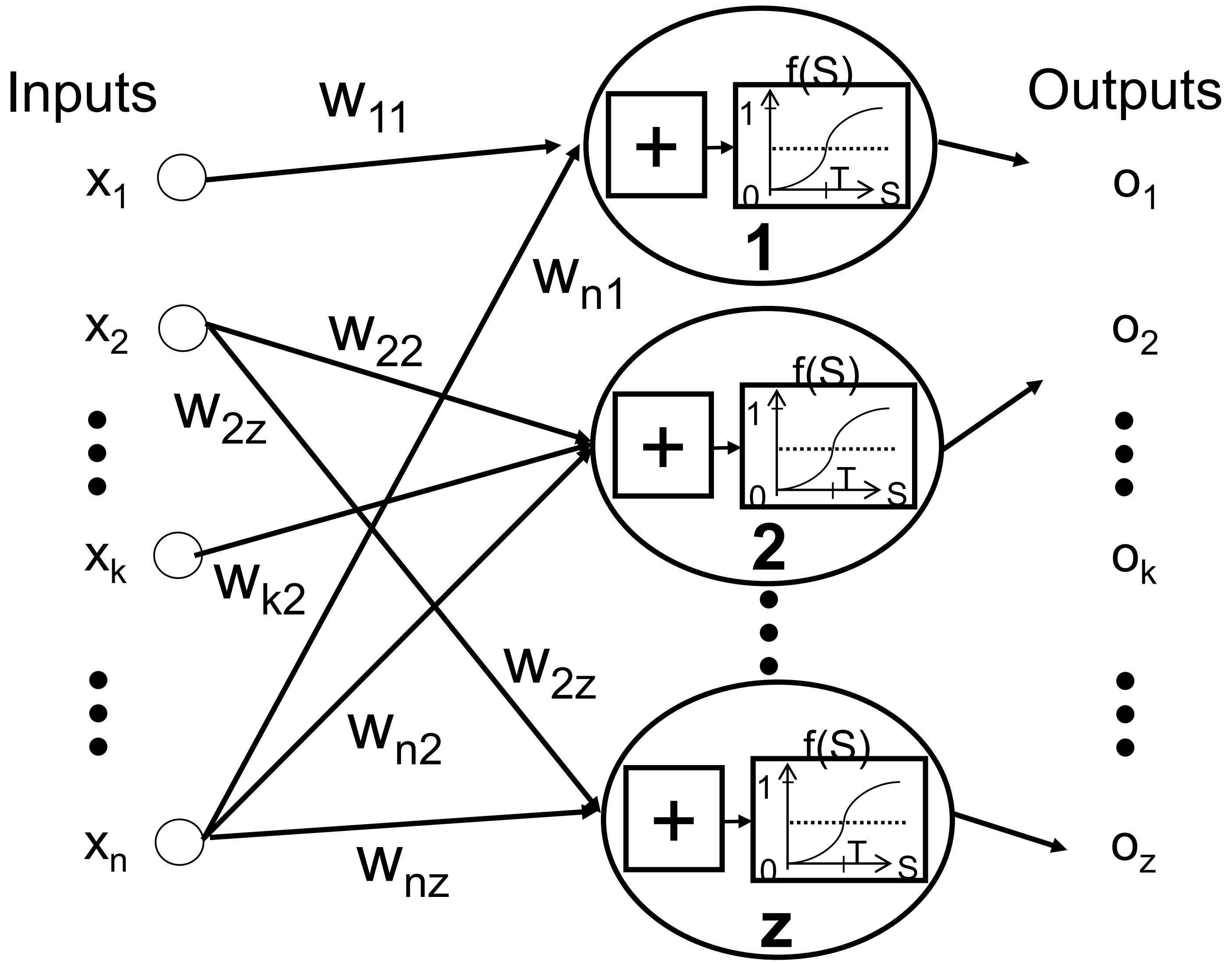
$$S = \sum_{i=1}^{n} x_i w_i$$

$$o_k = f(S)$$

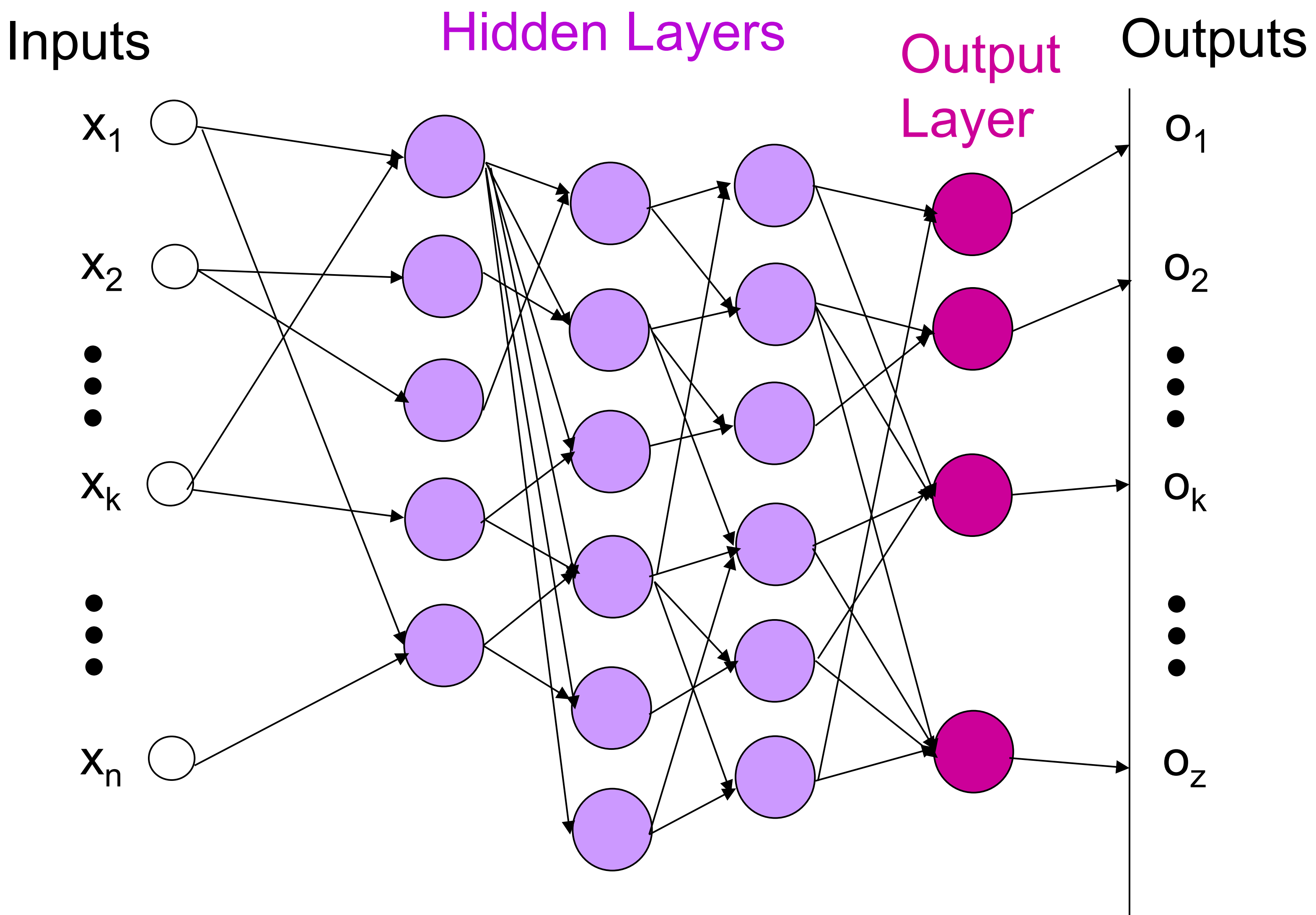
if $f(S) >= T$

$o_k >= 0$

2-Layer Neural Networks



Multilayer Neural Networks



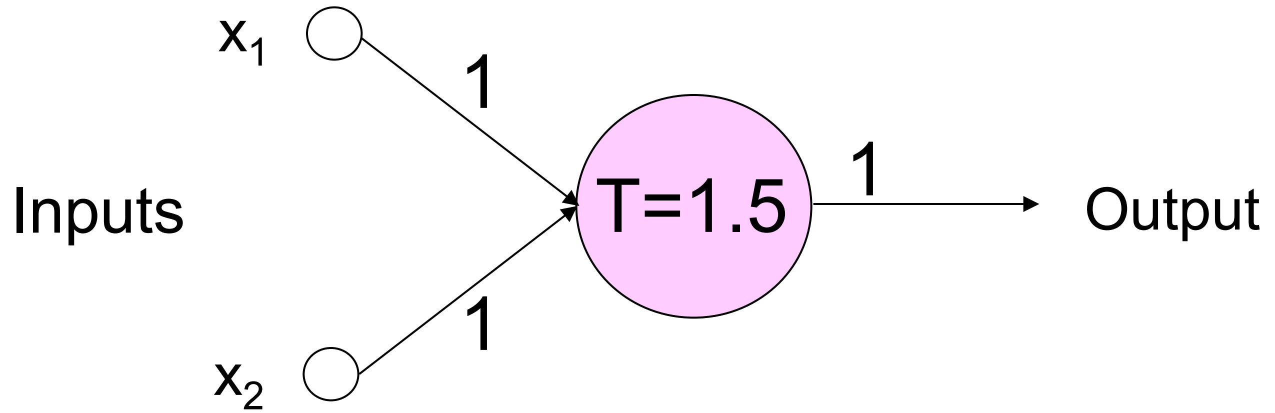
Tasks for Neural Networks

- Evaluation Problem
- Training Problem

Single Node Neural Net

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

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



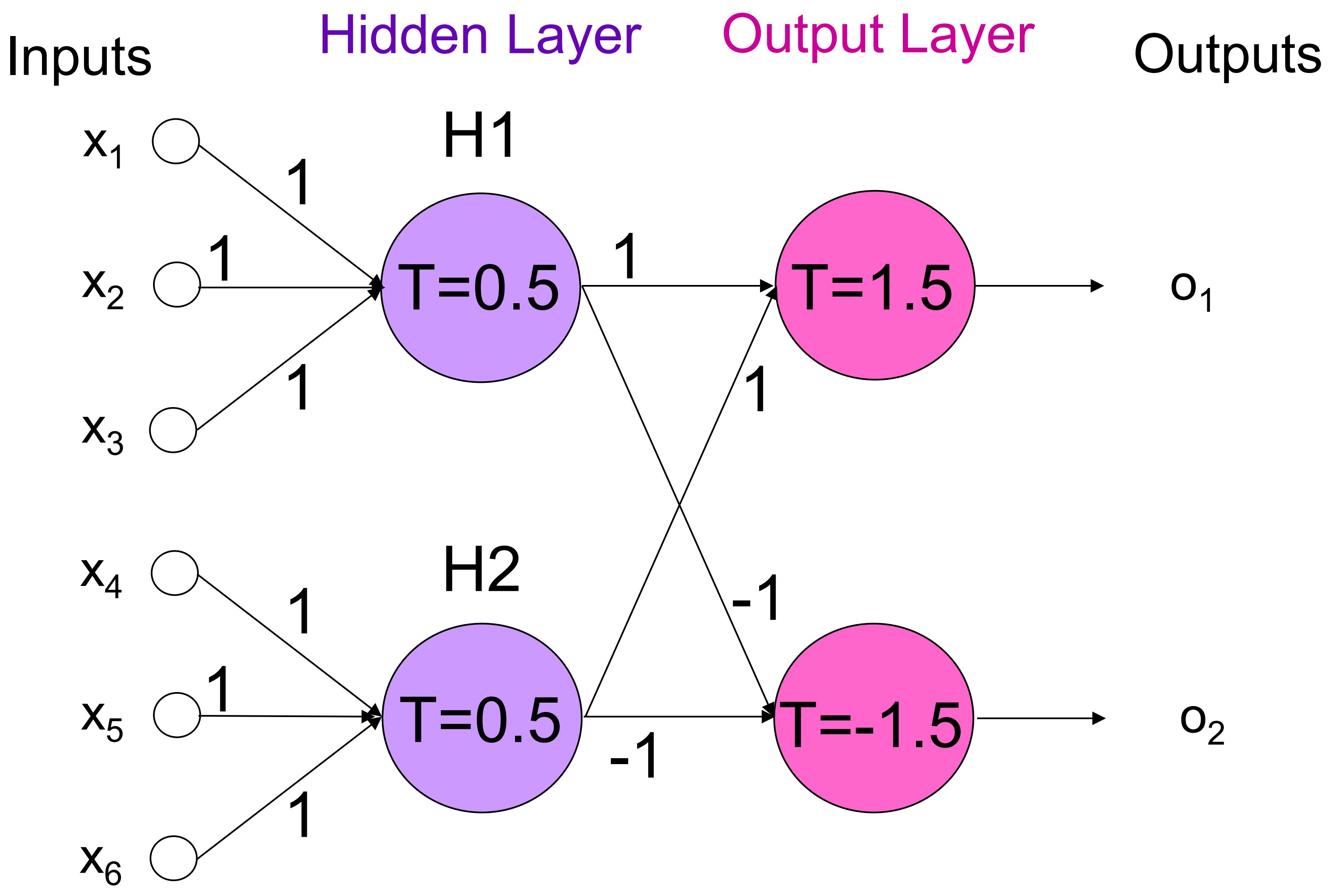
x_1	x_2	Computation	Output
0	0	0(1)+0(1)=0<1.5	0
0	1	0(1)+1(1)=1<1.5	0
1	0	1(1)+0(1)=1<1.5	0
1	1	1(1)+1(1)=2>1.5	1

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

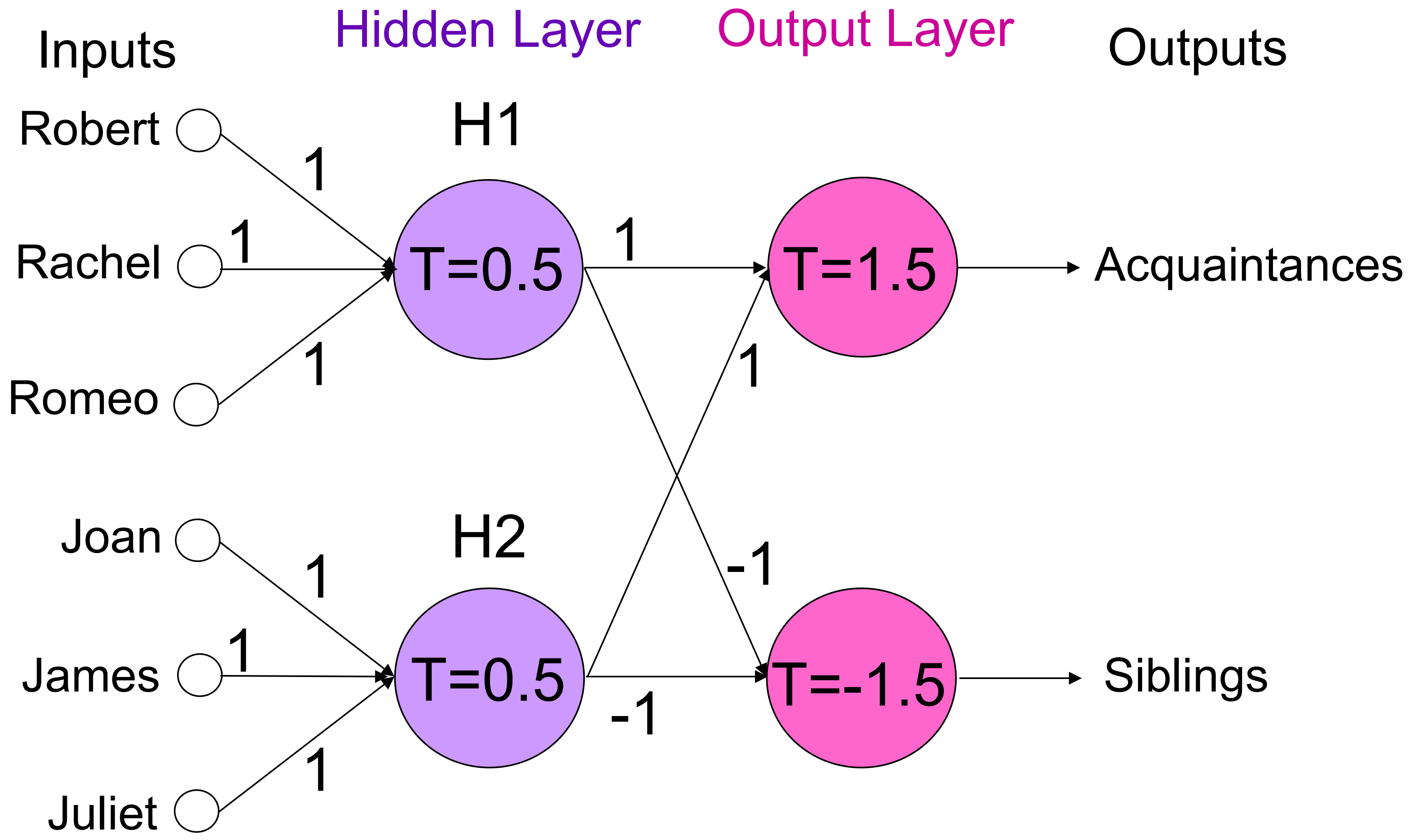
x_1	x_2	Computation	Output
0	0	0(1)+0(1)=0 < T	0
0	1	0(1)+1(1)=1 > T	1
1	0	1(1)+0(1)=1 > T	1
1	1	1(1)+1(1)=2 > T	1

T ?

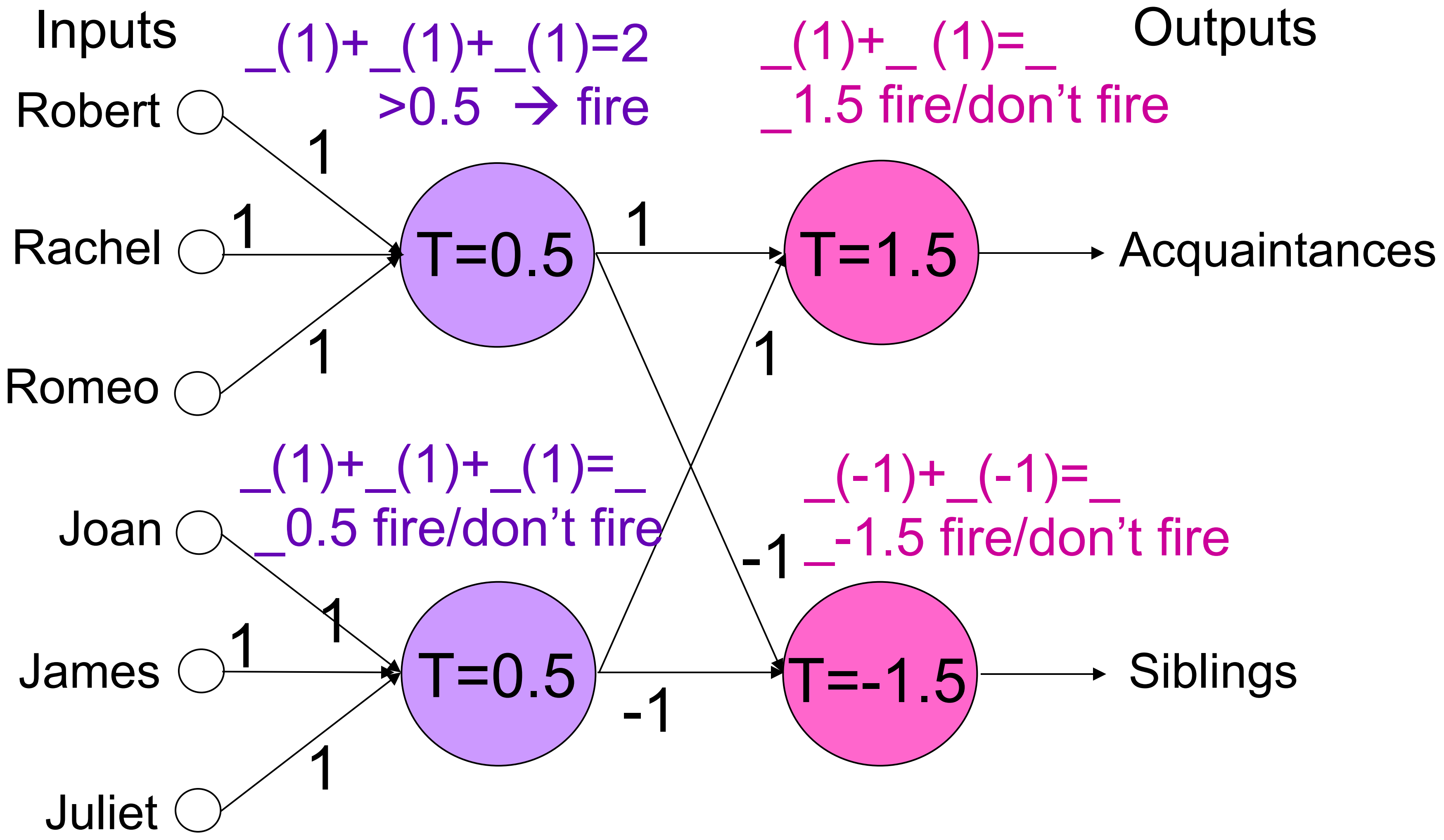
Example of a 3-layer Net



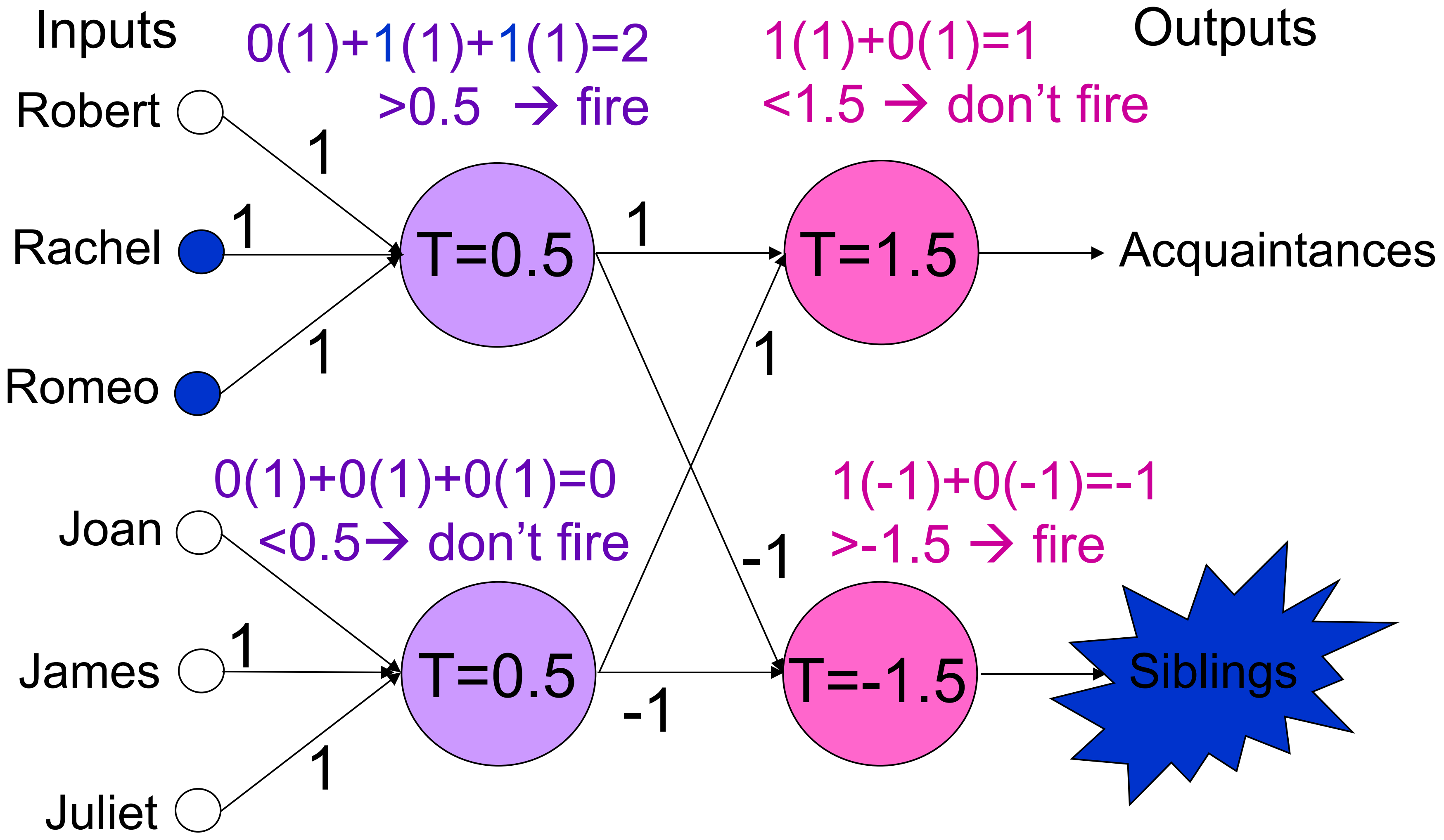
Acquaintance or Sibling Net



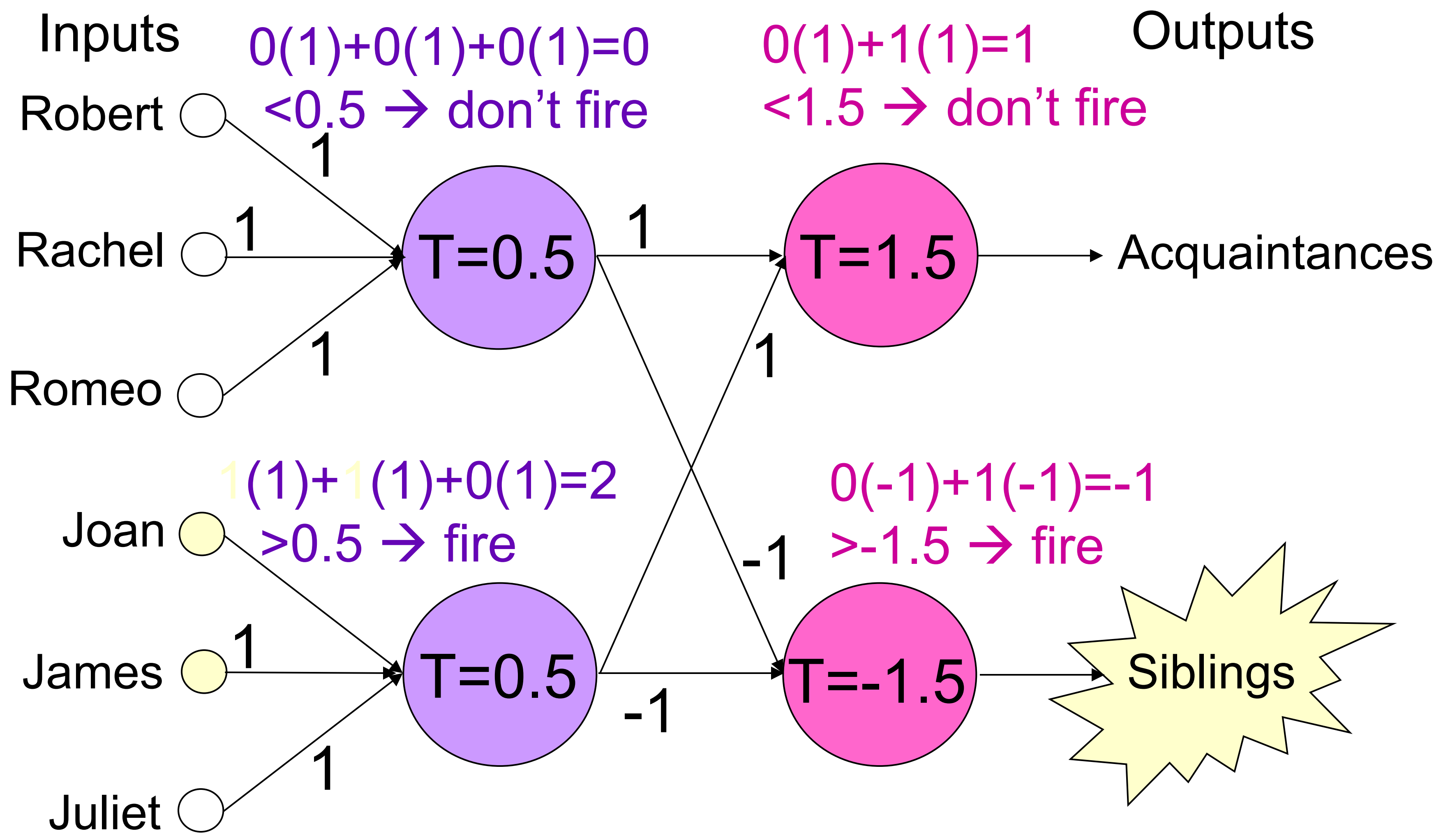
Acquaintances or Siblings?



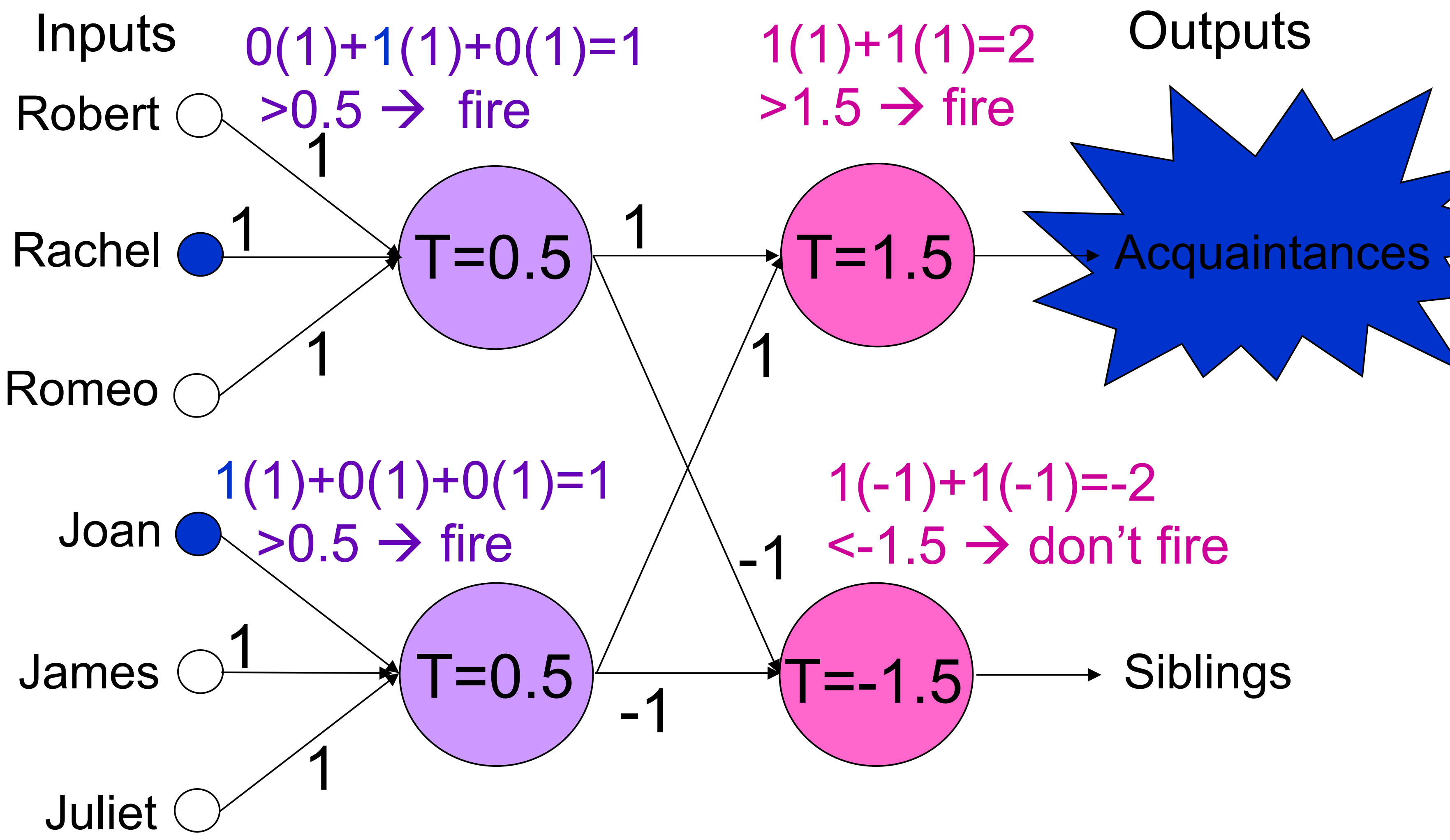
Acquaintances or Siblings?



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 $X_{training}$

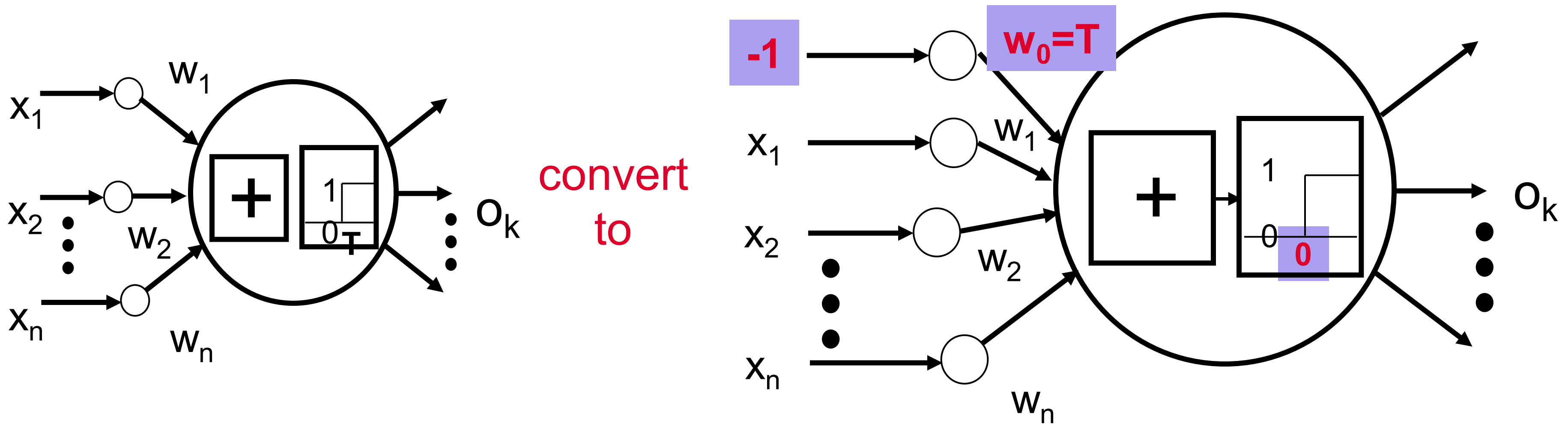
Network Architecture

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

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

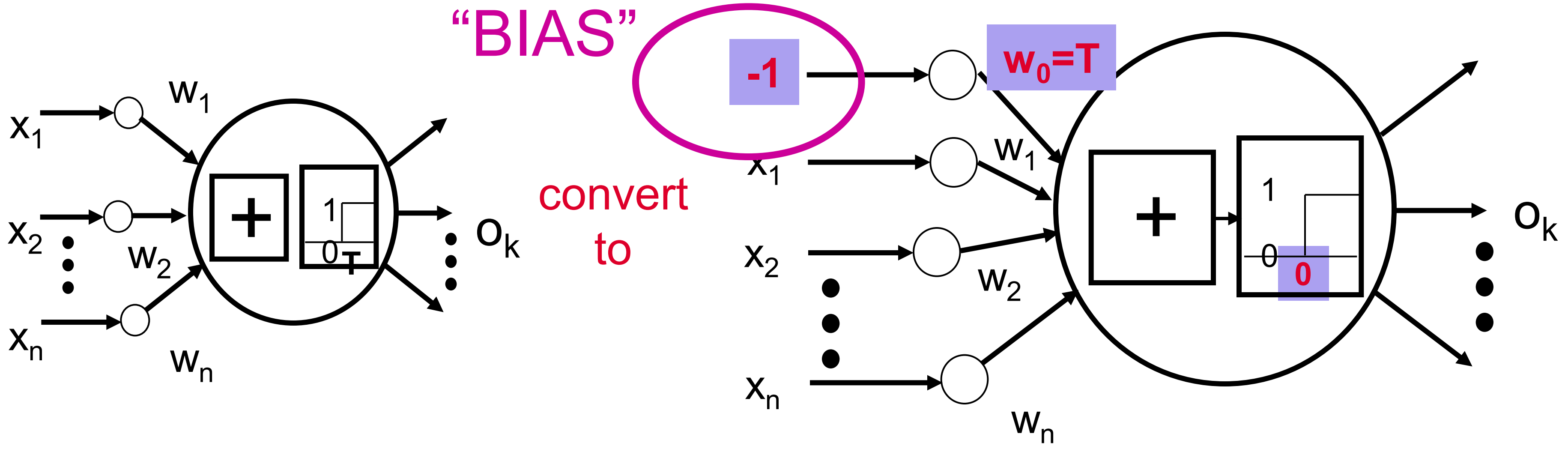
Training Problem – Version 1

- 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:



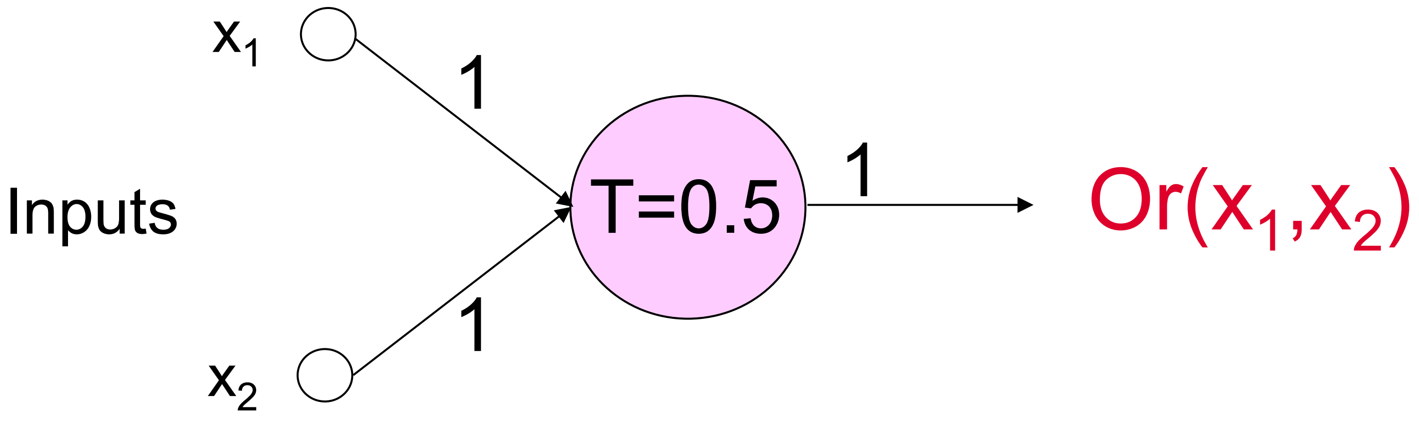
Training Problem – Version 1

- 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

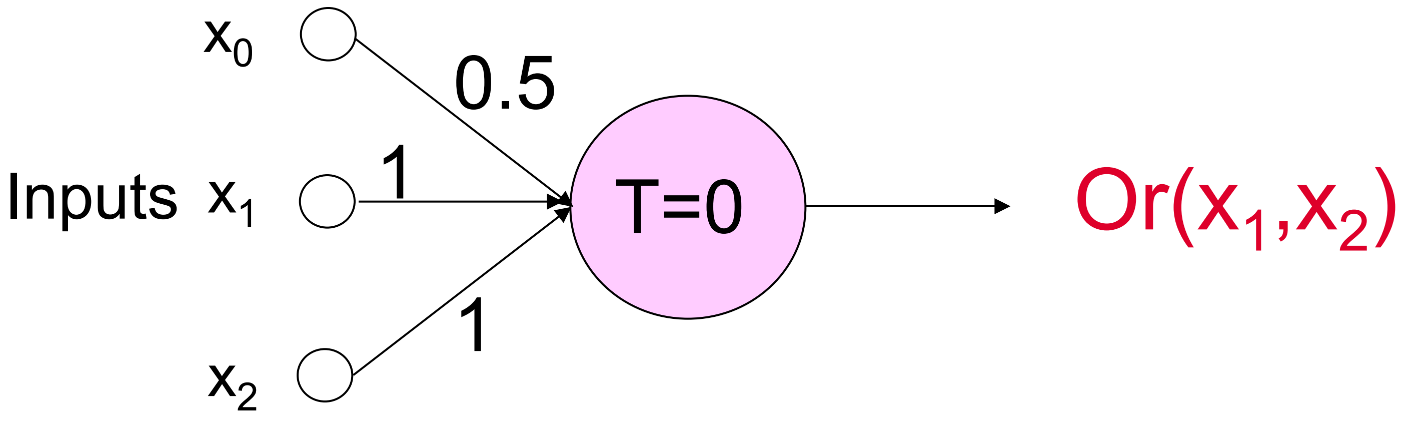
x_1	x_2	Computation	Output
0	0	$0(1)+0(1)=0 < 0.5$	0
0	1	$0(1)+1(1)=1 > 0.5$	1
1	0	$1(1)+0(1)=1 > 0.5$	1
1	1	$1(1)+1(1)=2 > 0.5$	1



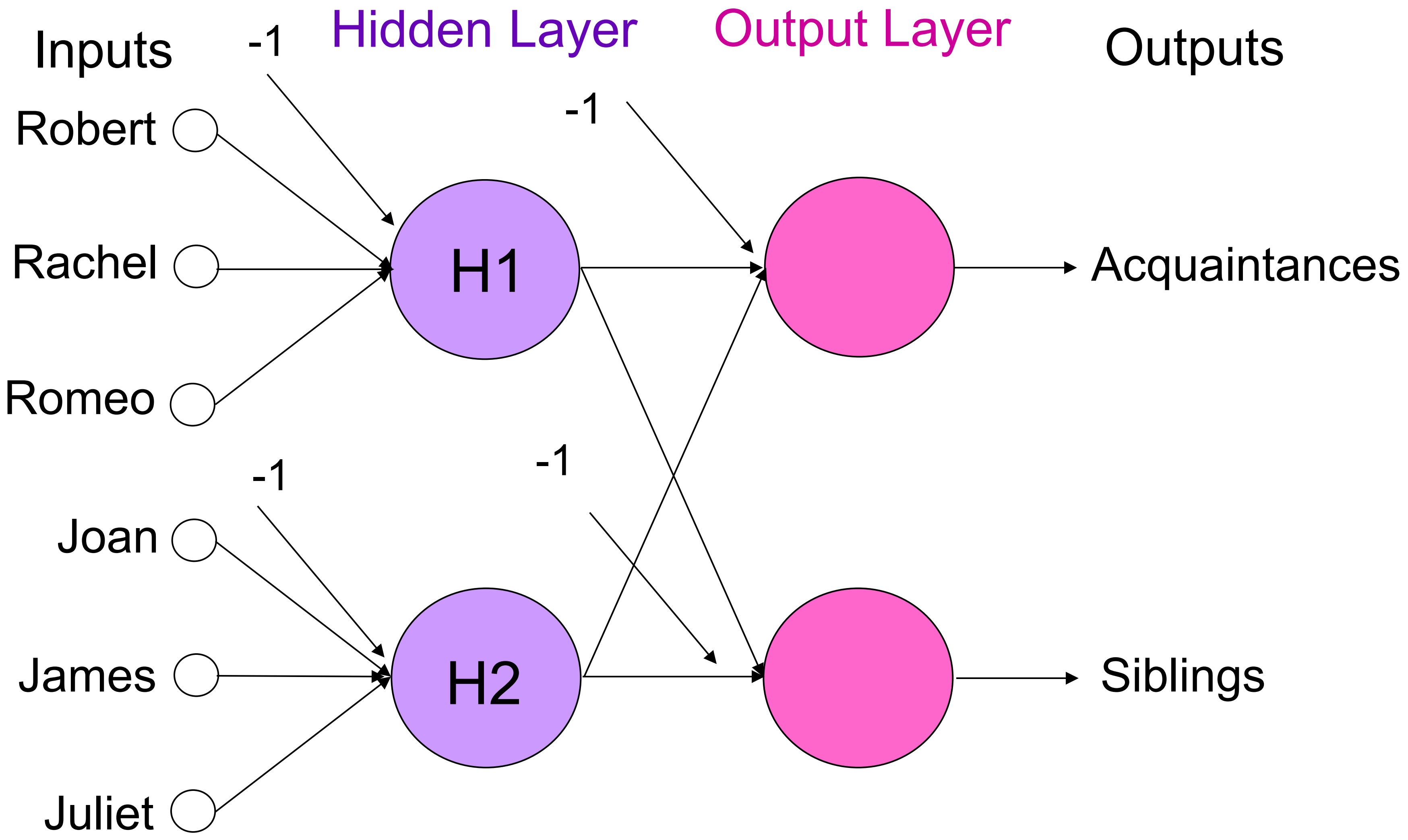
Example:
Converting Threshold to Weight

"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

CAS CS 640
Artificial Intelligence

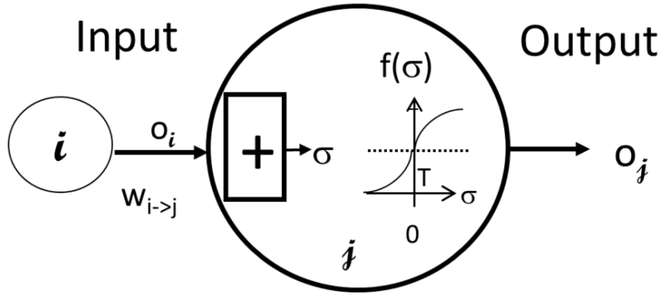
Learning by Training Neural Nets, Part 2

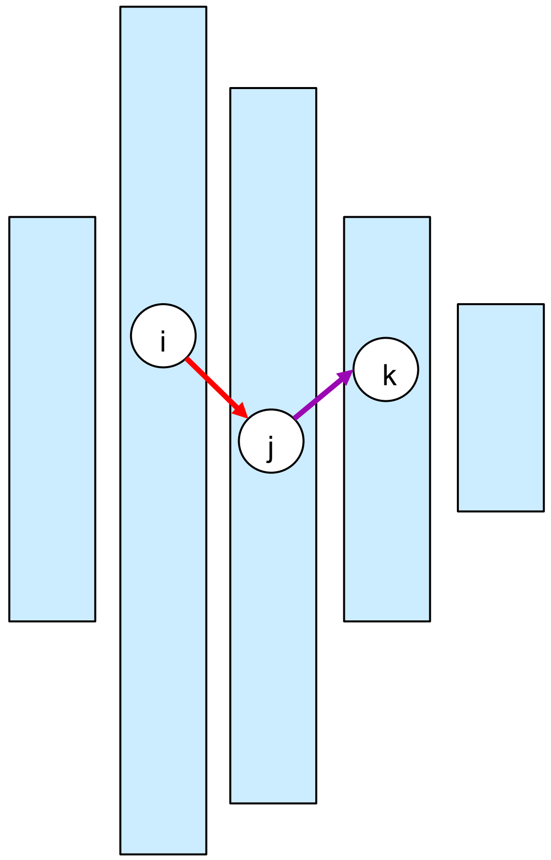
Adopted from P. Winston, Artificial Intelligence, 1992

1. Understanding Backpropagation requires an Understanding of Multivariate Functions, Derivates, and the General Chain Rule
2. Backprop Procedure on the Board (not all derivation details on the board can be found in Winston's book but main equations and variable names match)

Backpropagation Neural Net Training Algorithm

```
Input: NN structure (# nodes, # layers, activation function, here $\frac{1}{1+e^{-\sigma}}$),
       Labeled training data = input/output pairs $\{\vec{x}, \vec{d}\}$

1) Choose weights randomly (or some other way)

2) Compute performance $P$ on training data

3) WHILE performance $P$ not satisfactory:

    { FOR EACH input vector $\vec{x}$:
        {
            Compute the outputs of the last layer $\vec{o}_{last}$,
            (= evaluate NN on $\vec{x}$ = forward pass through network)

            Compute benefits $\beta_z$'s for all nodes in the last layer:

                $\beta_z = d_z - o_z$

            Compute benefits $\beta_j$'s using the recursive formula

                $\beta_j = \sum_{k=1}^{K} \beta_k \cdot \frac{\partial o_k}{\partial o_j} = \sum_{k=1}^{K} \beta_k \, o_k (1 - o_k) \, w_{j \to k}$

            for each layer from back to front of the NN, where $K$ is the
            number of nodes of the layer (layer index is omitted here).

            Compute and store weight changes: $\Delta w_{i \to j} = r \, o_i \, o_j (1 - o_j) \, \beta_j$
        }

    Add all stored $\Delta w$'s computed for *all* input vectors and update weights:

                $w_{i \to j,new} = w_{i \to j,old} + \Delta w_{i \to j}$

    Compute P using updated weights.

    }

Output: Weights (= trained neural net)
        Performance $P$ on trained net
```





Learning Objectives: Be able to

- Implement the Backpropagation Procedure
- Explain how to initialize a neural net
- Explain how to compute performance (orloss) of a network
- Explain the weight update equation
- Explain the recursive step
- Derive the weight update equation for an activation function that is not the sigmoid as in our example

CAS CS 640
Artificial Intelligence

Lectures by Margrit Betke

Learning by Training Neural Nets, Part 3

Adopted from P. Winston, Artificial Intelligence, 1992

Training Problem – Version 1

Example:

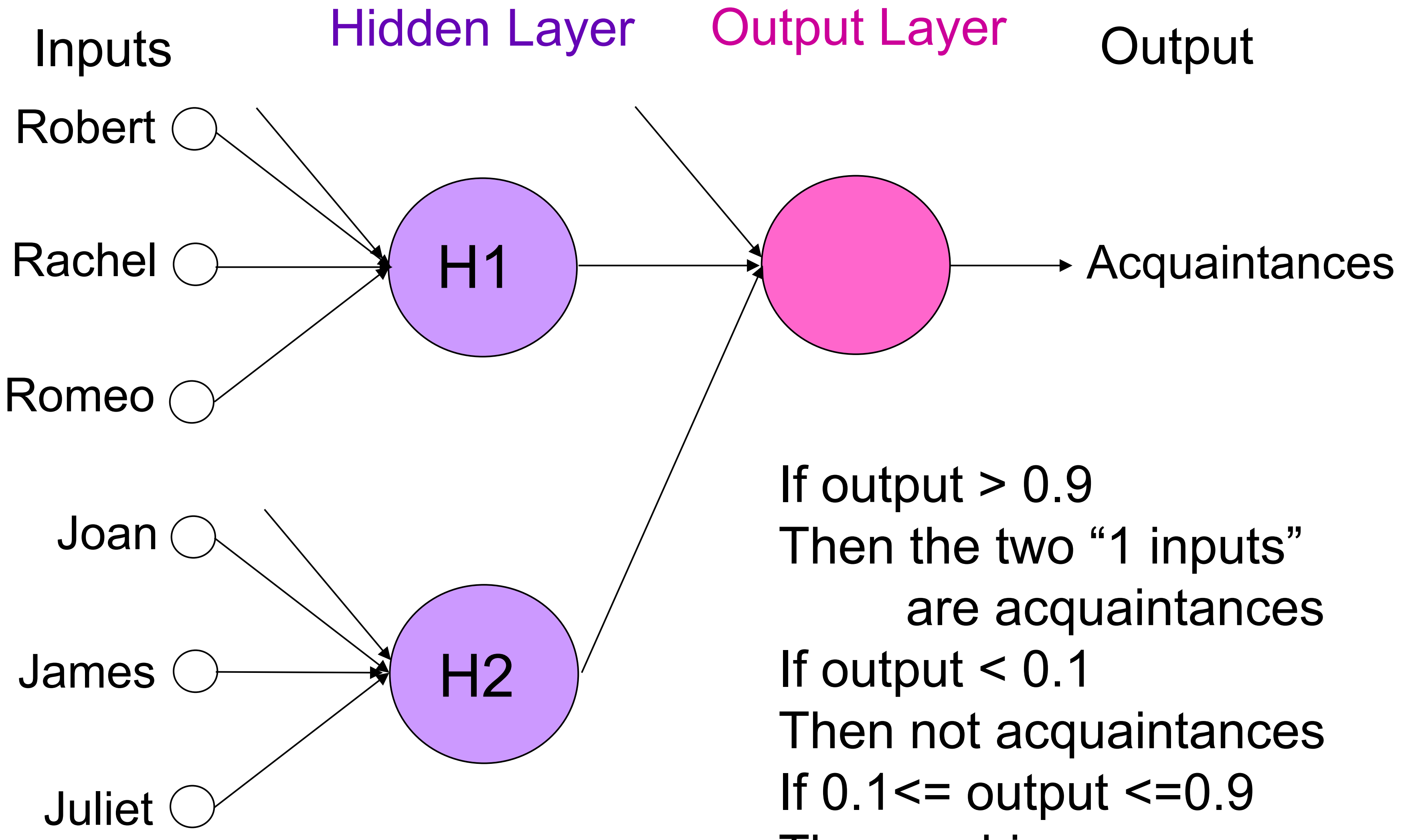
Acquaintance/Sibling Network

First:

Simplified Network:

Acquaintance Net

Acquaintance Net



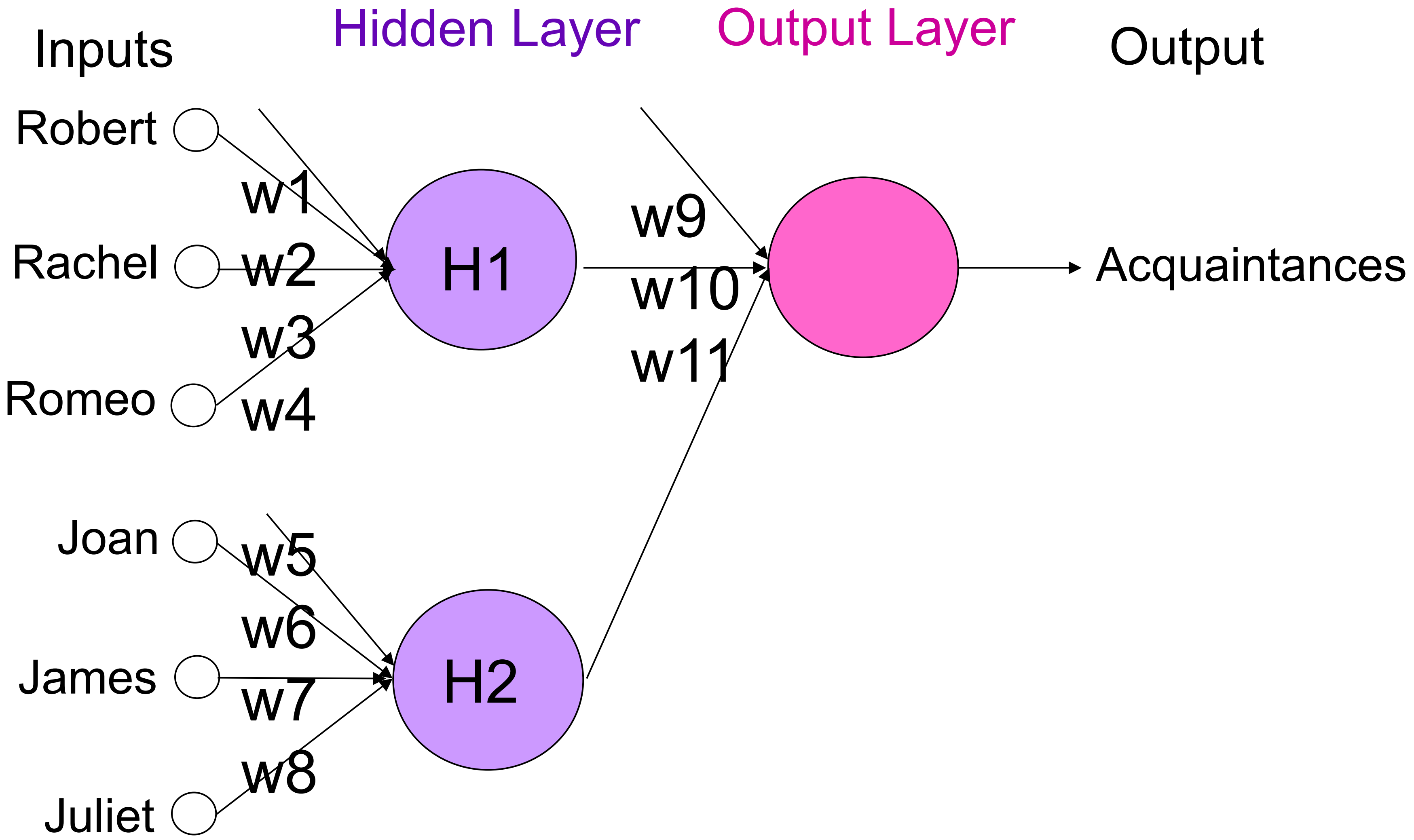
If output > 0.9
Then the two "1 inputs"
are acquaintances
If output < 0.1
Then not acquaintances
If 0.1<= output <=0.9
Then ambiguous

Only two inputs may be 1

Labeled Training Data: 15

Robert	Raquel	Romeo	Joan	James	Juliet	Acquaintan
1	1	0	0	0	0	0
1	0	1	0	0	0	0
1	0	0	1	0	0	1
1	0	0	0	1	0	1
1	0	0	0	0	1	1
0	1	1	0	0	0	0
0	1	0	1	0	0	1
0	1	0	0	1	0	1
0	1	0	0	0	1	1
0	0	1	1	0	0	1
0	0	1	0	1	0	1
0	0	1	0	0	1	1
0	0	0	1	1	0	0
0	0	0	1	0	1	0
0	0	0	0	1	1	0

Acquaintance Net



Backprop for the Acquaintance Net Initial Values of 11 Weights:

Weight	Initial Value	Value after Backprop
w1	0.1	1.99
w2	0.2	4.65
w3	0.3	4.65
w4	0.4	4.65
w5	0.5	2.28
w6	0.6	5.28
w7	0.7	5.28
w8	0.8	5.28
w9	0.9	9.07
w10	1	6.27
w11	1.1	6.12

RMS error during Training of Acquaintance Net

RMS error
0.5
0.4
0.3
0.2
0.1
0
0
100
200
300
400
500
Weight change cycles

Satisfactory performance after 225 weight changes

225 x 15 = 3,375 inputs processed

Learning rate = 1

Characteristics of Back-Propagation

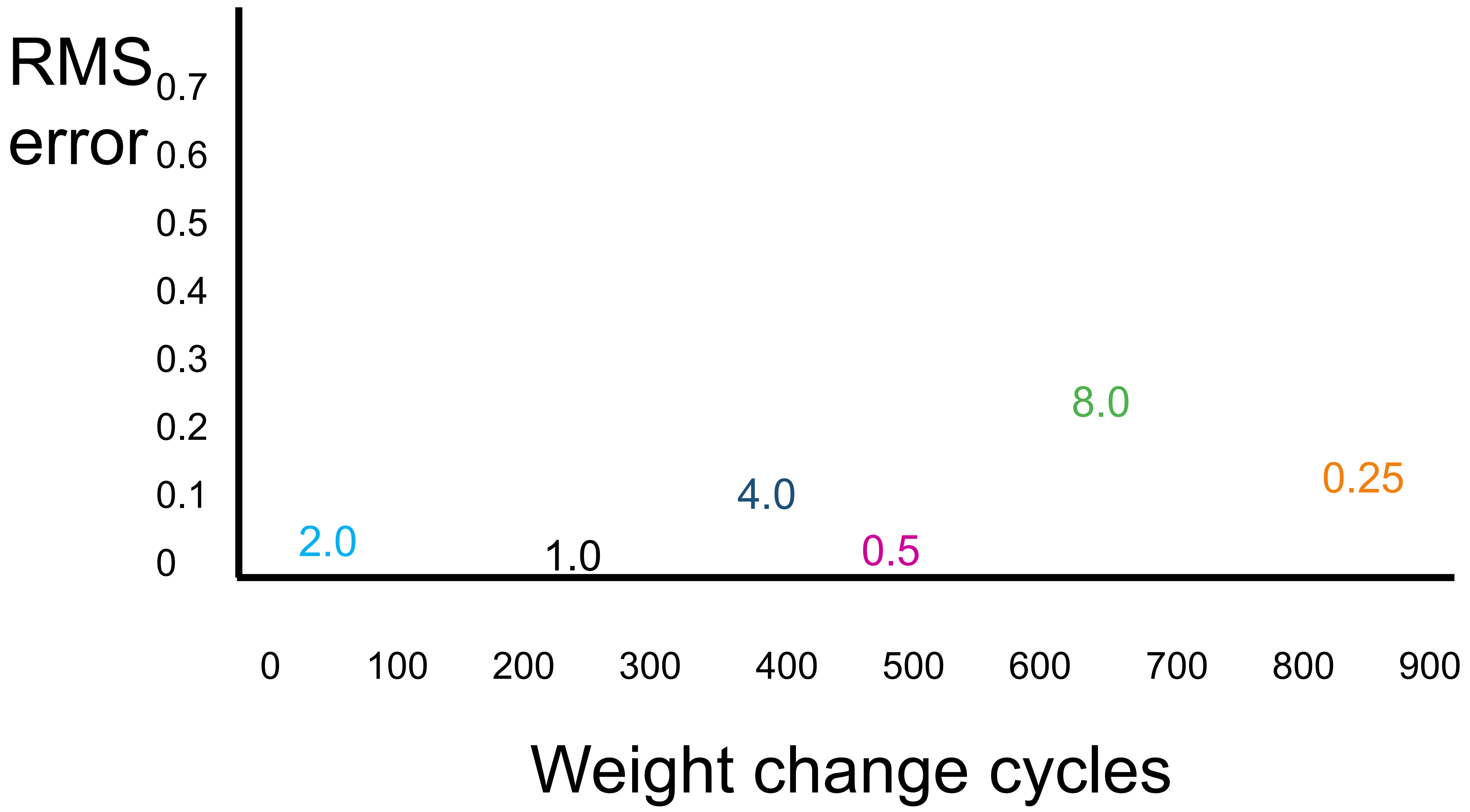
- Training a small network may require hundreds of backpropagations, a larger network maybe thousands

Characteristics of Back-Propagation

- Training can get stuck or become unstable:

r = 1.0	225 weight changes
r = 2.0	150 weight changes
r = 0.25	900 weight changes
r = 0.5	425 weight changes
r = 4.0	serious instability
r = 8.0	serious instability

Backprop can get stuck or become unstable



Characteristics of Back-Propagation

- Training may require thousands of backpropagations
- Training can get stuck or become unstable
  - No general learning rate rule
  - Rate selection is problem dependent

If learning rate too low: slow training

If learning rate too high: instability

Characteristics of Back-Propagation

- Training may require thousands of backpropagations
- Training can get stuck or become unstable
- Training can be done in stages:

Later stages refine training of network in earlier stages

Characteristics of Back-Propagation

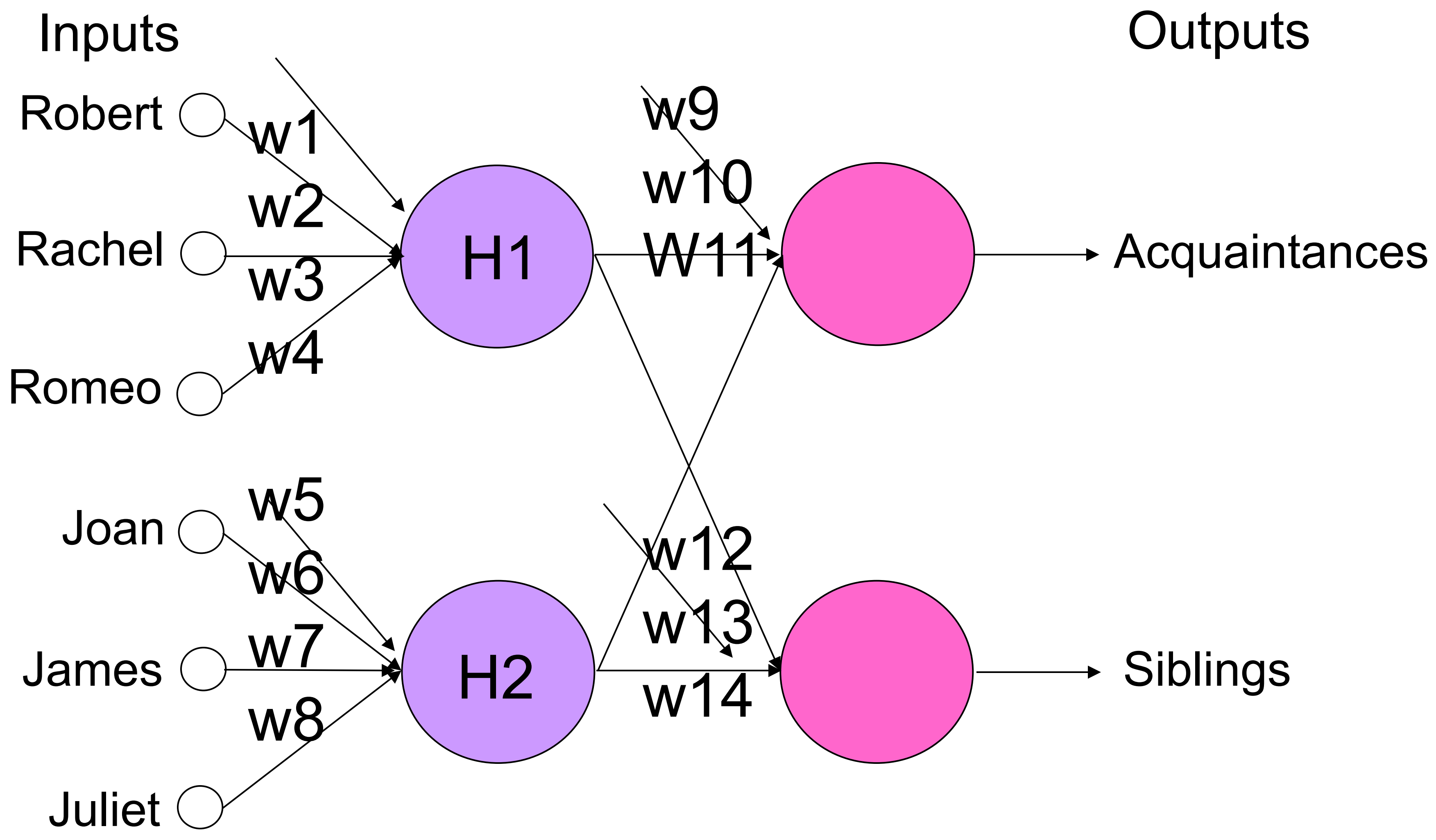
- Training may require thousands of backpropagations
- Training can get stuck or become unstable
- Training can be done in stages:

Later stages refine training of network in earlier stages

Example:

To train Acquaintance or Sibling Net, use the trained Acquaintance Net as the pre-trained model and extent the model by one output node

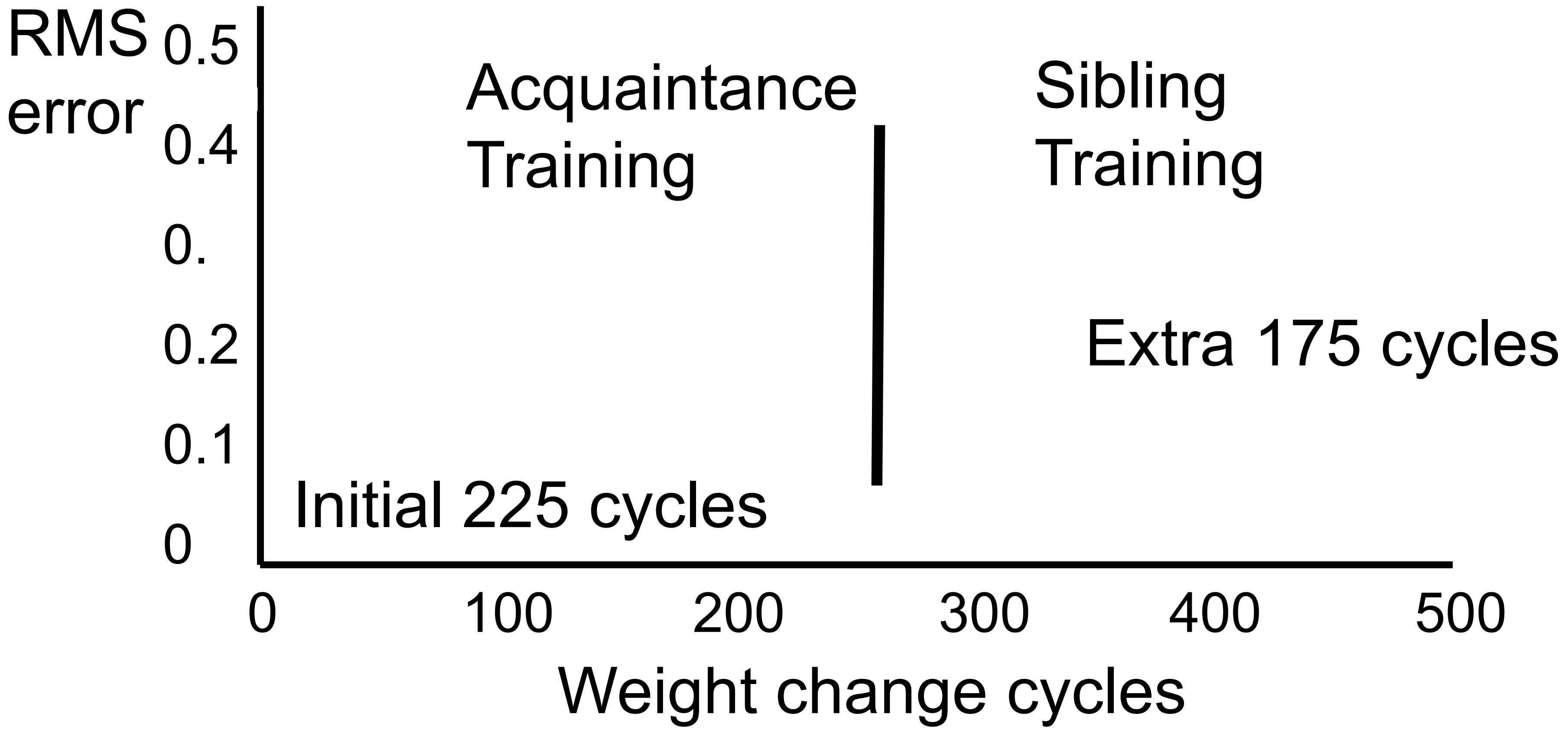
Acquaintance or Sibling Net



2-Stage Training of Ac/Sib Net

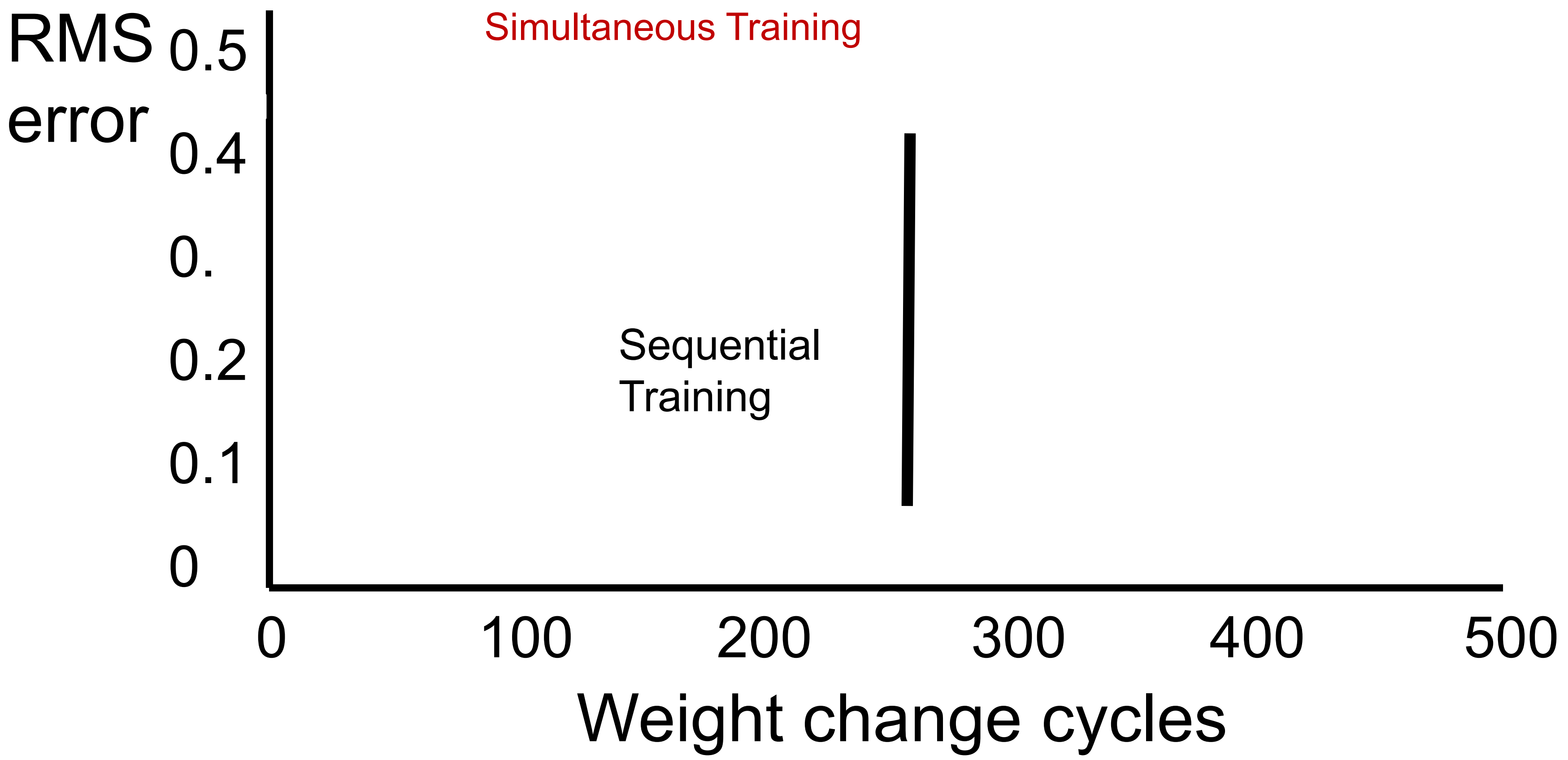
	Weight	Initial Value	Value after Pretraining	Value after Sibling Training
	w1	0.1	1.99	2.71
	w2	0.2	4.65	6.02
	w3	0.3	4.65	6.02
Stage 1	w4	0.4	4.65	6.02
	w5	0.5	2.28	2.89
	w6	0.6	5.28	6.37
	w7	0.7	5.28	6.37
	w8	0.8	5.28	6.37
	w9	0.9	9.07	10.29
	w10	1	6.27	7.04
	w11	1.1	6.12	6.97
	w12	1.2		-8.32
Stage 2	w13	1.3		-5.72
	w14	1.4		-5.68

RMS Error during Two-State Training



400 cycles total

Simultaneous Training of 14 Weights of Full Acquaintance/Sibling Net



Characteristics of Back-Propagation

- Training may require thousands of backpropagations
- Training can get stuck or become unstable
- Training can be done in stages
- Trained neural nets can make predictions

Labeled Dataset: 15 Samples

Robert	Raquel	Romeo	Joan	James	Juliet	Acquaintance	Sibling
1	1	0	0	0	0	0	1
1	0	1	0	0	0	0	1
1	0	0	1	0	0	1	0
1	0	0	0	1	0	1	0
1	0	0	0	0	1	1	0
0	1	1	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	0	0	1	0	1	0
0	1	0	0	0	1	1	0
0	0	1	1	0	0	1	0
0	0	1	0	1	0	1	0
0	0	1	0	0	1	1	0
0	0	0	1	1	0	0	1
0	0	0	1	0	1	0	1
0	0	0	0	1	1	0	1

We used all 15 samples for training! Nothing left for testing…

Robert	Raquel	Romeo	Joan	James	Juliet	Acquaintance	Sibling
1	1	0	0	0	0	0	1
1	0	1	0	0	0	0	1
1	0	0	1	0	0	1	0
1	0	0	0	1	0	1	0
1	0	0	0	0	1	1	0
0	1	1	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	0	0	1	0	1	0
0	1	0	0	0	1	1	0
0	0	1	1	0	0	1	0
0	0	1	0	1	0	1	0
0	0	1	0	0	1	1	0
0	0	0	1	1	0	0	1
0	0	0	1	0	1	0	1
0	0	0	0	1	1	0	1

Use 3 Samples for Testing, Train on the Remaining 12 Samples

Robert	Raquel	Romeo	Joan	James	Juliet	Acquaintance	Sibling
1	1	0	0	0	0	0	1
1	0	1	0	0	0	0	1
1	0	0	1	0	0	1	0
1	0	0	0	1	0	1	0
1	0	0	0	0	1	1	0
0	1	1	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	0	0	1	0	1	0
0	1	0	0	0	1	1	0
0	0	1	1	0	0	1	0
0	0	1	0	1	0	1	0
0	0	1	0	0	1	1	0
0	0	0	1	1	0	0	1
0	0	0	1	0	1	0	1
0	0	0	0	1	1	0	1

Testing Result:

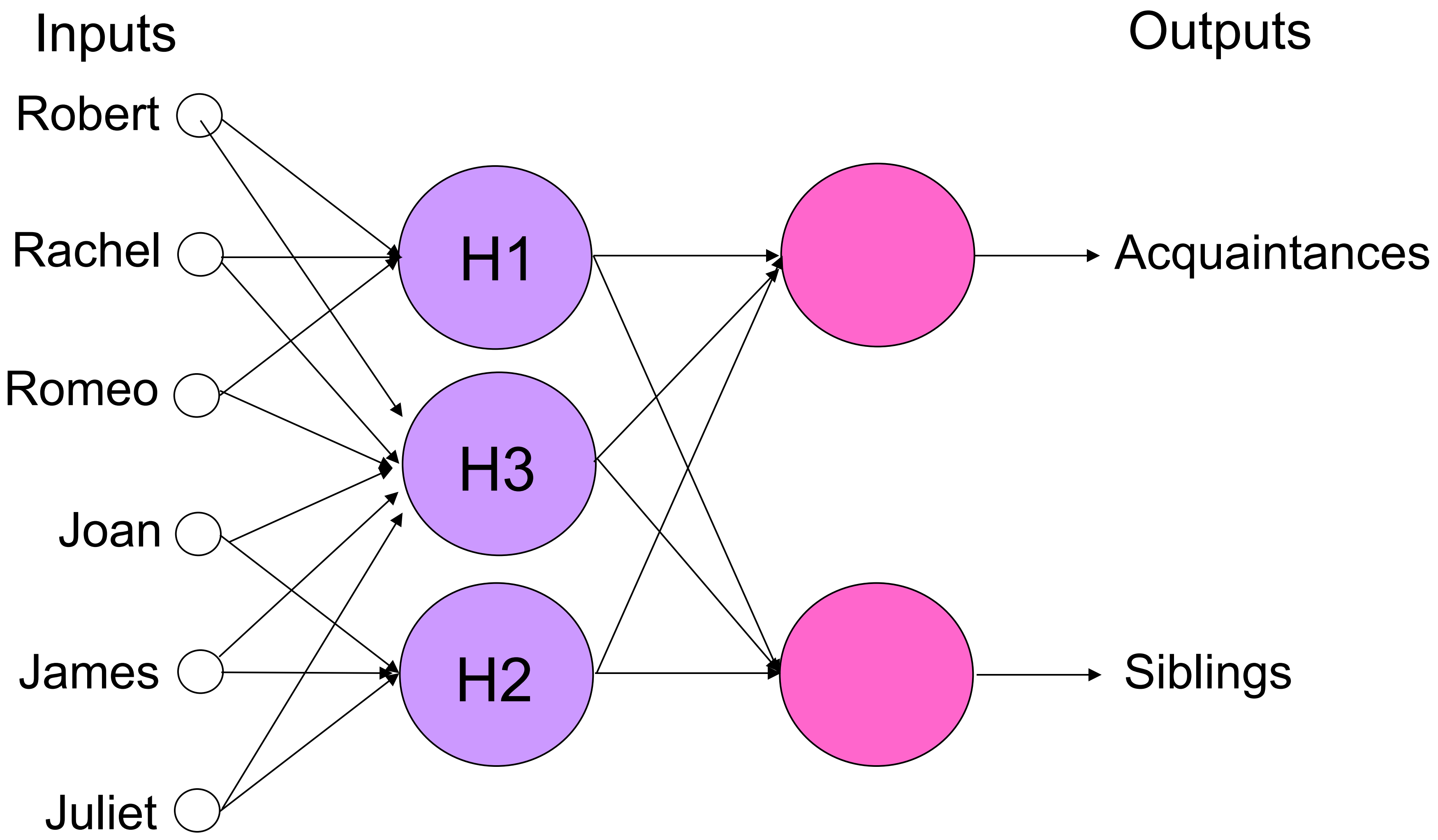
	Acquaintance		Sibling	
	Desired	Computed	Desired	Computed
Robert/Juliet	1	0.92	0	0.06
Romeo/Joan	1	0.92	0	0.06
James/Juliet	0	0.09	1	0.91

Interpretation:
Trained Acq/Sib net deals successfully with previously unseen data = it can predict!

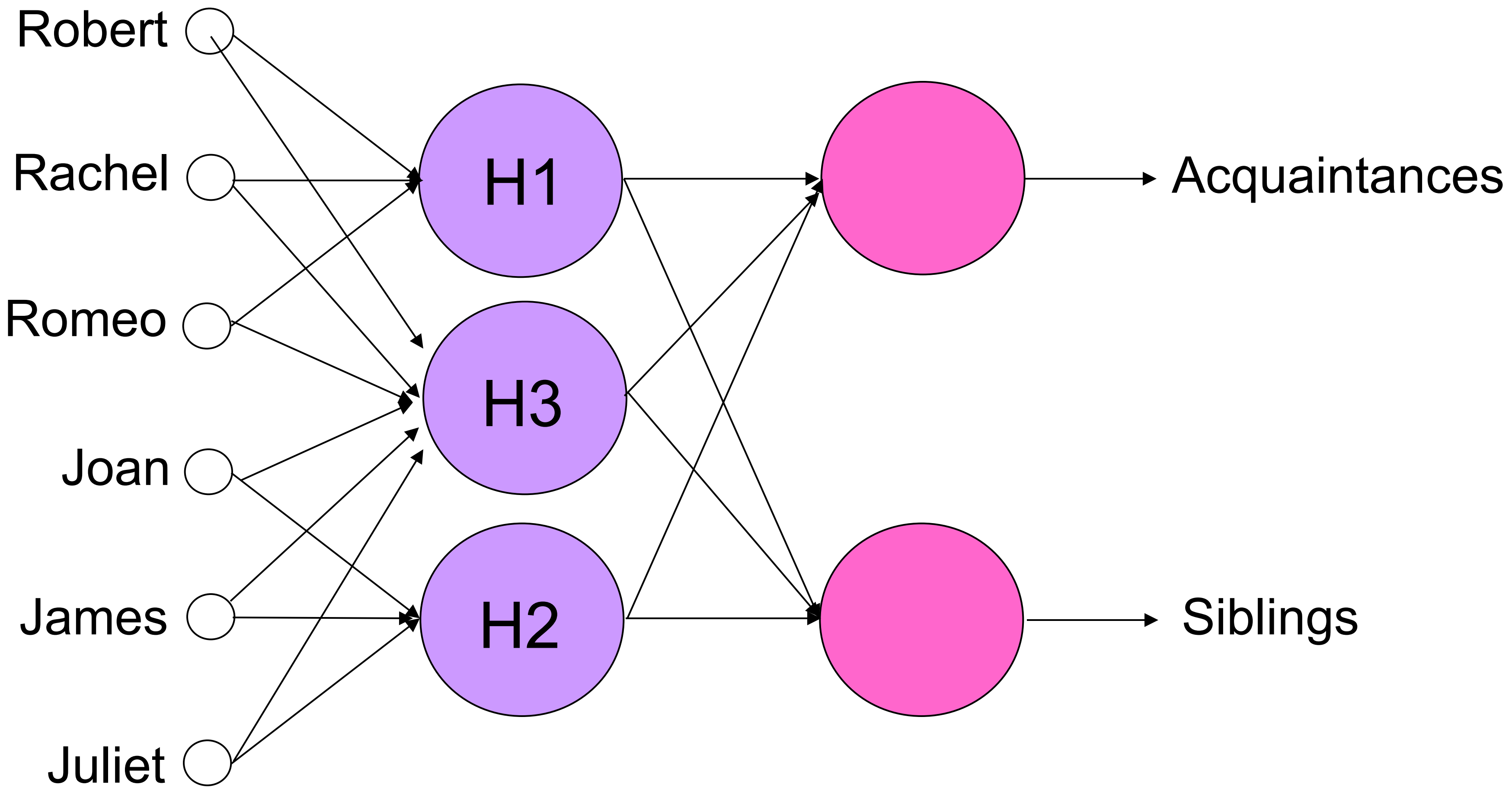
Characteristics of Back-Propagation

- Training may require thousands of backpropagations
- Training can get stuck or become unstable
- Training can be done in stages
- Trained neural nets can make predictions
- Excess weights lead to overfitting

Would a net with more trainable weights do better?



Training only takes 300 cycles – the extra weights make it too easy to deal with the training set



Testing Result:

	Acquaintance		Sibling	
	Desired	Computed	Desired	Computed
Robert/Juliet	1	0.99	0	0
Romeo/Joan	1	0.06	0	0.94
James/Juliet	0	0.97	1	0.01

Interpretation: Overfitting Occurred

Trained Acq/Sib net does not deal successfully with previously unseen data.

It cannot predict two of the three test cases correctly

Heuristic to Avoid Overfitting

Number of trainable weights influencing a particular output should be less than the number of training samples

- In Acquaintance/Sibling Network with two hidden nodes: 11 trainable weights & 12 input-output samples: a dangerously small margin!
- In Acquaintance/Sibling Network with three hidden nodes: 19 trainable weights & 12 input-output samples: 7 (>50%) more weights than i/o samples – overfitting is inevitable!

Characteristics of Back-Propagation

shown for Sibling/Acquaintance Neural Net

generalizes to all neural networks

- Training may require thousands of backpropagations
- Training can get stuck or become unstable
- Training can be done in stages
- Trained neural nets can make predictions
- Excess weights lead to overfitting

Patrick Winston (1943–2019)

"Neural-net experts are artists; they are not mere handbook users."

Occham's Razor

= Law of succinctness

Which hypothesis among h_1, h_2, h_3 ... should the AI system choose?

Choose the simplest hypothesis consistent with the data.

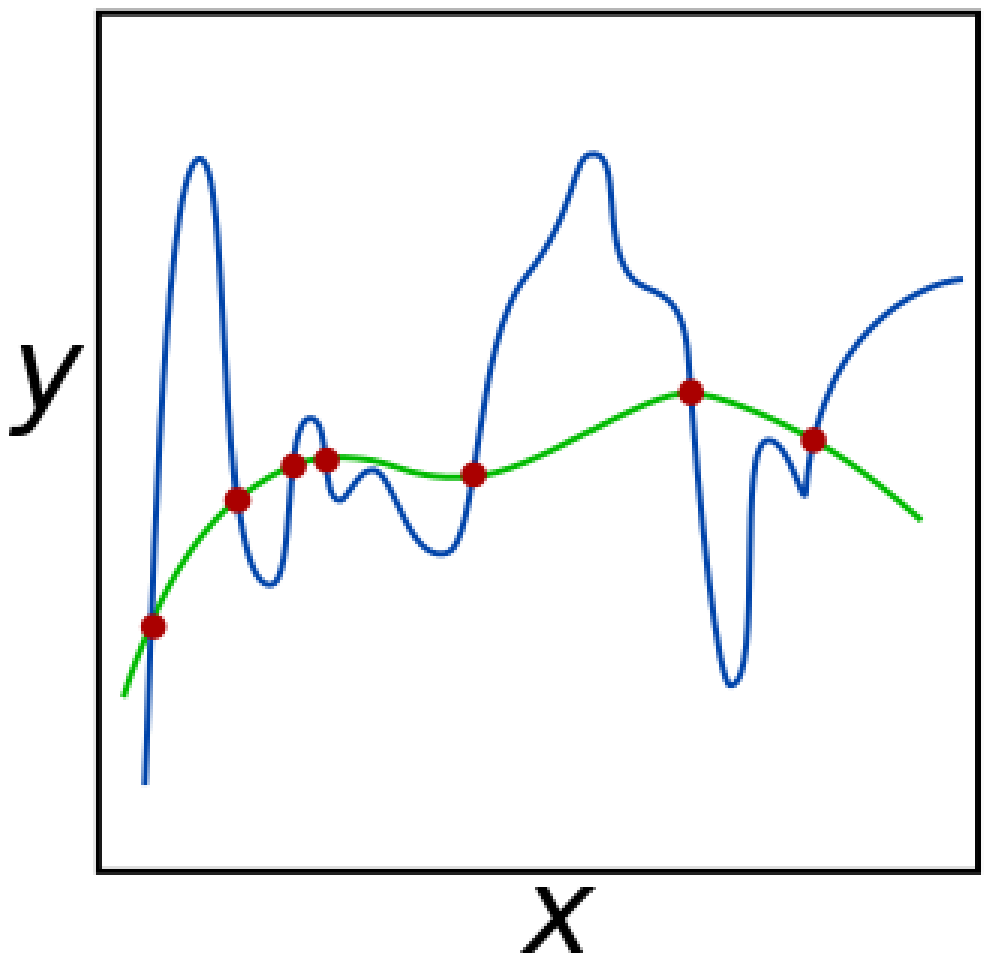
The simplest explanation will be the most plausible until evidence is presented to prove it false.

Example: Prefer a degree-1 polynomial (line) over a degree-7 polynomial

Trade-off between complex hypothesis that fit training data well and simpler hypotheses that may generalize better (and can typically be computed faster)

Russell and Norvig

Occam’s Razor: Choose green over blue model for h



Source: Wikipedia

Overfitting

- Avoid choosing an excessively complex learning system= model= hypothesis=neural net h.
- h is too complex if it has too many parameters relative to the number of observations.
- A model which has been overfitted will generally have poor predictive performance, as it can exaggerate minor fluctuations in the data.
- Higher-degree polynomials or complicated neural nets with many hidden layers and nodes fit the data better but may lead to overfitting.

Russell and Norvig

Overfitting

- Avoid choosing an excessively complex learning system= model= hypothesis=neural net h.
- h is too complex if it has too many parameters relative to the number of observations.
- A model which has been overfit will generally have poor predictive performance, as it can exaggerate minor fluctuations in the data.
- Higher-degree polynomials or complicated neural nets with many hidden layers and nodes fit the data better but may lead to overfitting.

Solutions:

1. Use “wrapper” to enumerate models h according to model size (e.g., number of nodes or layers in neural net). Select model with smallest error.
2. Feature selection: Simplify model by discarding irrelevant attributes (dimensionality reduction). See below.
3. Minimum description length: Select model with smallest number of bits required to encode program and data.

Russell and Norvig

Part 3 Learning Objectives: Understand that

- The AI network designer must monitor RMS error (or some other metric) during network training
- The AI designer can train in stages by simplifying a network, training that network, and then generalizing to the larger network
- Training may require thousands of epochs
- Training may get stuck
- Training may become unstable
- Too many weights lead to overfitting
- Test and training data must be different
- Overfitting can be avoided by (1) starting simple and enumerating models with a wrapper or (2) reducing features

CAS CS 640

Artificial Intelligence

Lectures 5, 6, and 7, 2024

Margrit Betke

Machine Learning

Cross Validation

Cross-Validation

Holdout cross-validation =
Randomly split available (input,output) pairs into a single training set to learn *h* and a single test set to test the learned *h*.

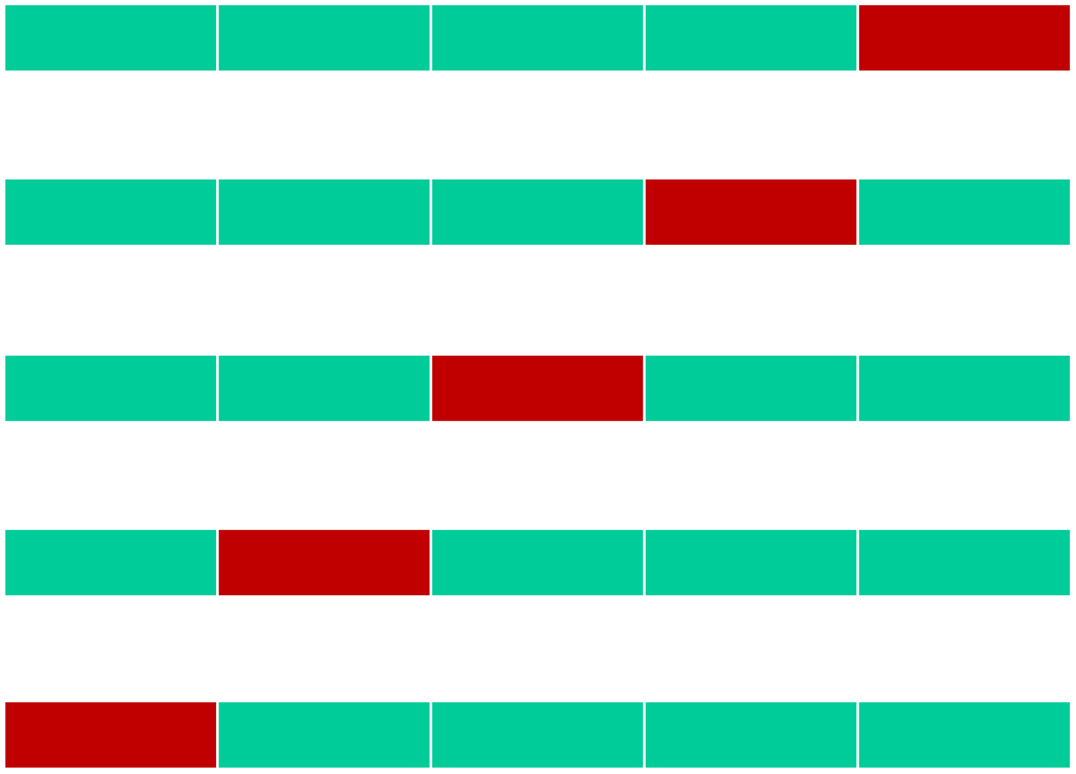
k-fold cross-validation =

- Split data into *k* equal subsets.
- Perform *k* rounds of learning. Each round leaves *1/k* examples out of the training set that can then be used as the test set.
- The average test set score should be a better estimate than a single score (need to keep *k h*'s around for prediction). Typically, *k*=5 or 10.

Leave-one-out cross validation: *k=N*.

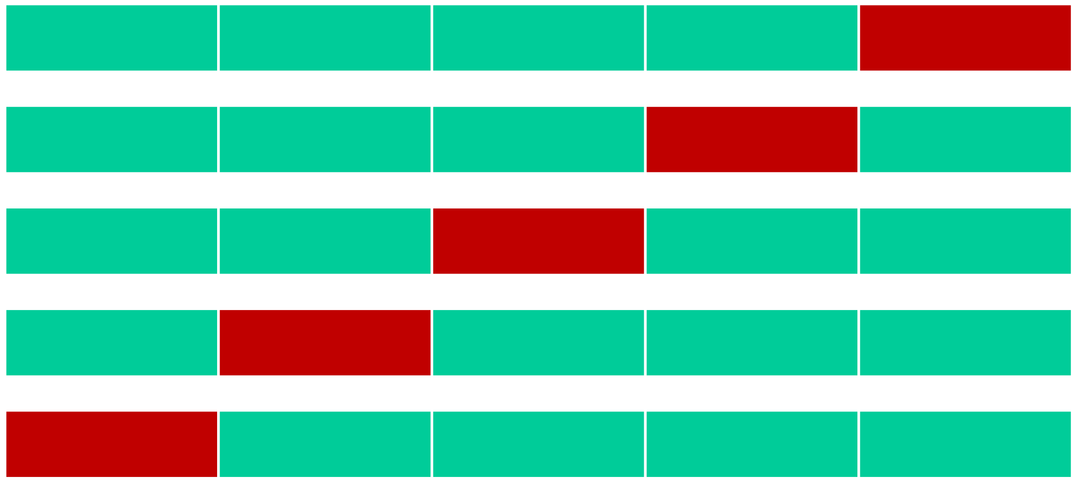
Simple 5-Fold Cross-Validation

5 Folds of Labeled Dataset: **Training** & **Testing**



Simple 5-Fold Cross-Validation

1. Create 5 folds of the labeled dataset for **training** & **testing**

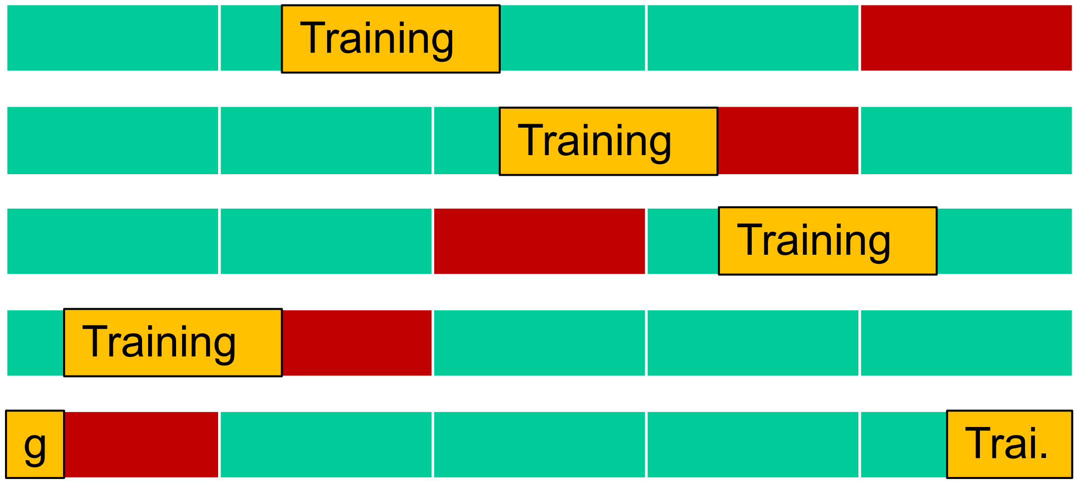


2. Train 5 AI models
3. Conduct ROC analysis for each model
4. Report average performance (accuracy, etc.)
5. Use the best of 5 models in your application (external data)

5-Fold Cross-Validation with Random Selection of Folds

What if you were simply lucky in your **training** & **testing** folds?

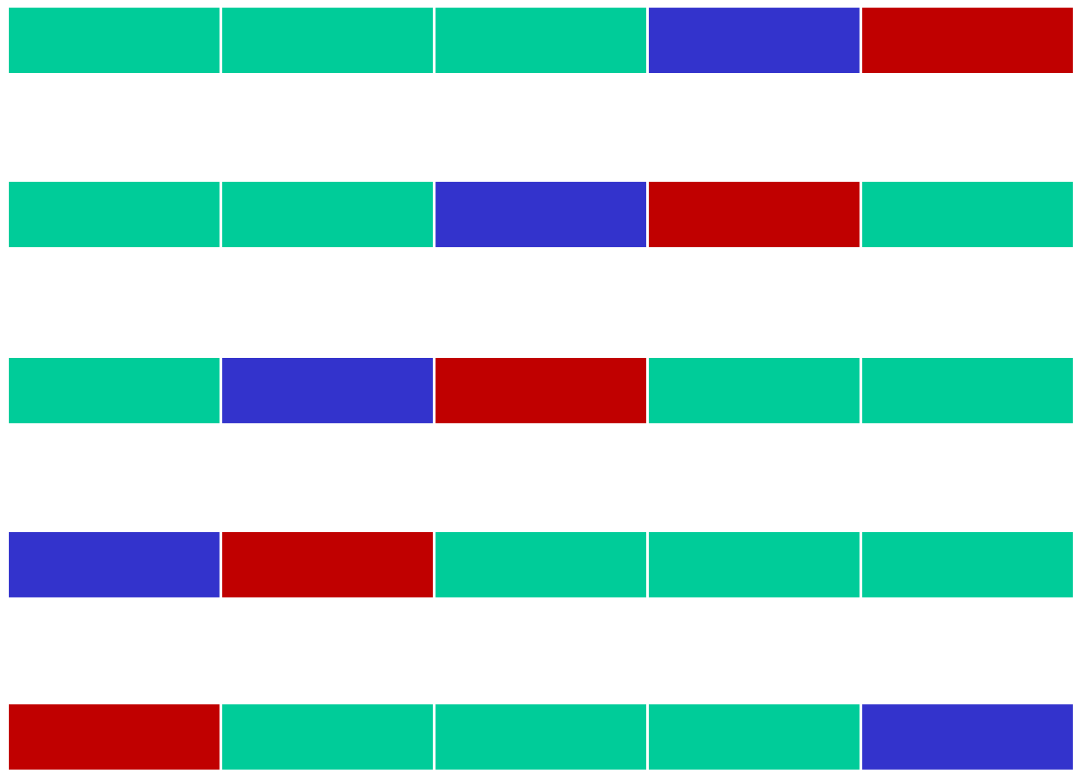
1. Randomly create a different set of 5 folds & report average performance:



2. Do this *n* times & report *1/n* performance sum, i.e., average performance

Cross-Validation with Train/Validation/Test Sets

Training, Validation, & **Testing**



Why Validation?

With the validation data, we tune the hyperparameters of an AI model.

The validation process tells us whether training is moving in the right direction. It can be performed after each training epoch.

The validation process helps prevent our model from overfitting to the training data by challenging it on the unseen validation data.

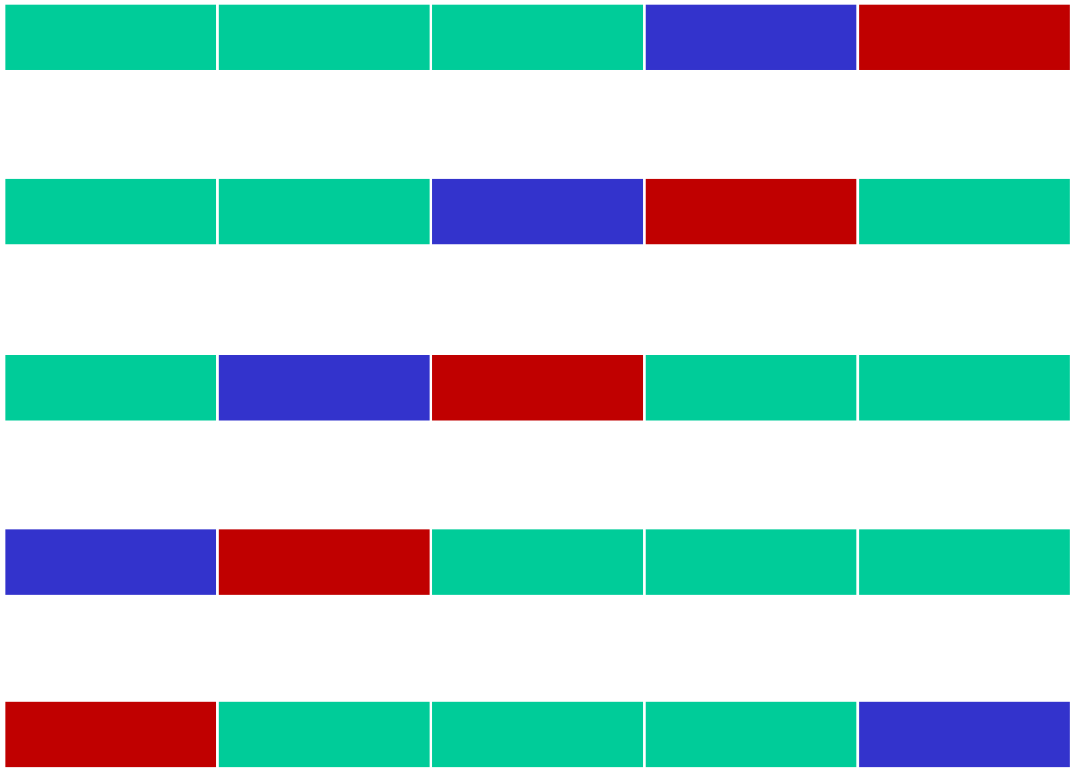
The validation data must therefore be separate from the training data (and of course from the test data).

What are common Train/Validation/Test Splits?

Training, Validation, & **Testing**

60%, 20%, 20% **or** **80%, 10%, 10%**

5 folds 10 folds



How to Train/Test when feature reduction is used to avoid overfitting

Feature selection: Simplify model by discarding irrelevant attributes (reduction to *k* features).

Use the normalized correlation coefficient r for feature reduction.

How to compare features *x* and desired regression values *y*

Normalized correlation coefficient:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where

x_i = value of ith feature, $\bar{x}$ mean feature value

y_i = value of ith desired regression value (ground truth),
$\bar{y}$ mean desired regression value

Properties of r: $-1 <= r <= 1$

E[r]=0 in Gaussian case

Normalized Correlation Coefficient

The normalized correlation coefficient was discovered by Auguste Bravais in 1844.

It was named after Karl Pearson (1857-1936), a proponent of eugenics and scientific racism.

Please avoid using Pearson’s name to describe the normalized correlation coefficient.

Source: Wikipedia link

Comparing Features and Labels: Example involving Stroke Survivors

Features x

Age

Months since Stroke

Spared Grey Matter in Region 1

Spared Grey Matter in Region 2

etc.

Spared Grey Matter in Region 50

Spared White Matter in Region 1’

Spared White Matter in Region 2’

etc.

Label y

Language Test Score [0..100]

Comparing Features and Labels: Example involving Stroke Survivors

<u>Feature</u> Age x	<u>Label</u> y
Age[patient 1]	Language Test Score [patient 1]
Age[patient 2]	Language Test Score [patient 2]
…	
Age[patient n]	Language Test Score [patient n]

Compute $r_{age}[x,y] = r[(x1, ..., xn)^T,(y1,…,yn)^T]$

=> This determines how strong the correlation between the feature *age* and the *language test score* of a stroke survivor is.

Comparing Features and Labels: Example involving Stroke Survivors

Feature x=Months-since-stroke (MSS)	Label y
MSS[patient 1]	Language Test Score [patient 1]
MSS[patient 2]	Language Test Score [patient 2]
…	
MSS[patient n]	Language Test Score [patient n]

Compute $r_{MSS}[x,y] = r[(x1, ..., xn)^T,(y1,…,yn)^T]$

=> This determines how strong the correlation between the feature *MSS* and the *language test score* of a stroke survivor is.

Feature Reduction: Example involving Stroke Survivors

r_{Age}

r_{MSS}

$r_{\text{Spared Grey Matter in Region 1}}$

…

$r_{\text{Spared Grey Matter in Region n}}$

$r_{\text{Spared White Matter in Region 1'}}$

…

$r_{\text{Spared White Matter in Region n'}}$

$\xrightarrow{\text{Sort with largest r on top}}$

r_{Age}

$r_{\text{Spared Grey Matter in Region 5}}$

…

$r_{\text{Spared Grey Matter in Region 12}}$

r_{MSS}

$r_{\text{Spared White Matter in Region 29'}}$

…

$r_{\text{Spared White Matter in Region 33'}}$

Select top k features (with k largest r's)

How to Train/Test when feature reduction is used to avoid overfitting

Feature selection: Simplify model by discarding irrelevant attributes (reduction to *k* features).

Common questionable practice for regressors:

1) For each feature, compare the vector of feature values for all samples in the dataset with the vector of desired output values.
2) Sort features by result of this comparison
3) Use highest ranked *k* features (rank computed with NCC) in subsequent usual cross-validation procedure

How to Train/Test when feature reduction is used to avoid overfitting

Feature selection: Simplify model by discarding irrelevant attributes (reduction to *k* features).

Common better practice for regressors:

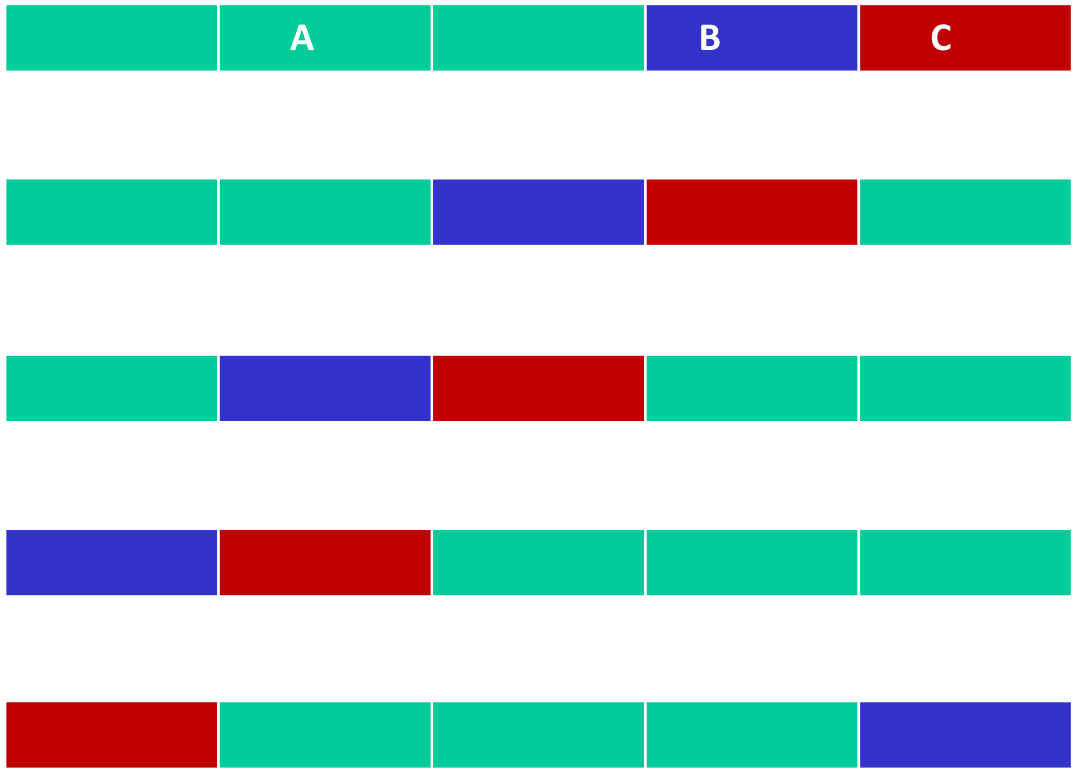
1) For each feature, compare the vector of feature values for samples in the training dataset with the vector of desired output values.
2) Sort features by result of this comparison
3) Use highest ranked *k* features (rank computed with NCC) in subsequent nested cross-validation procedure

Nested Cross Validation

Outer loop: 5 experiments

Inter loop: Not just [A|B]C but also [4 versions of A,B]C

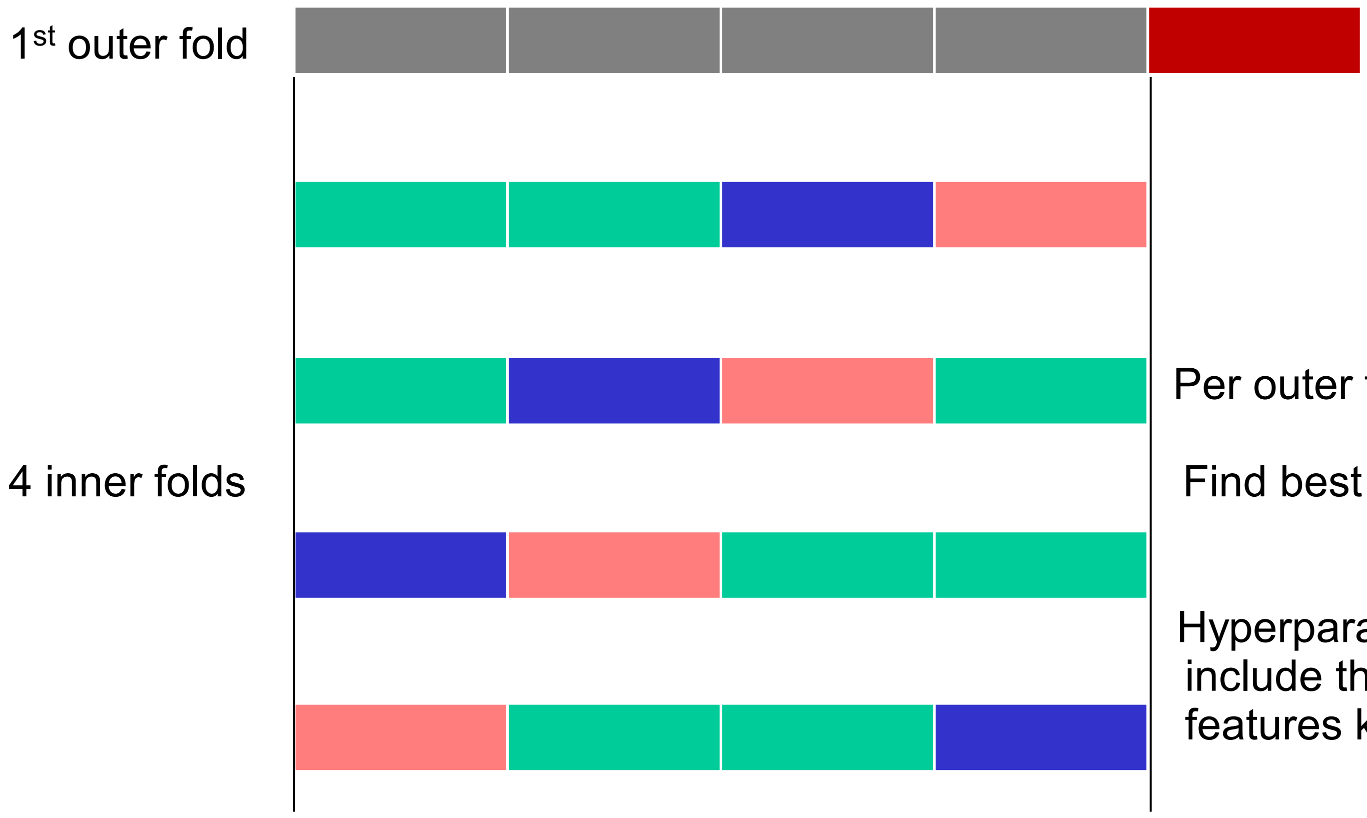
Training, Validation, & Testing



Nested Cross Validation

Here: 5 outer folds, 4 inner folds, report performance of 5 models

Training, Validation, & Testing



Per outer fold:

Find best model among 4

Hyperparameters could even include the number of features k

Research Paper with Nested Cross Validation and Feature Selection:

Saurav Chennuri, Sha Lai, Anne Billot, Maria Varkanitsa, Emily J. Braun, Swathi Kiran, Archana Venkataraman, Janusz Konrad, Prakash Ishwar, and Margrit Betke. Fusion Approaches to Predict Post-stroke Aphasia Severity from Multimodal Neuroimaging Data. Accepted at the International Conference on Computer Vision Workshop on Computer vision for Automated Medical Diagnosis (ICCV CVAMD 2023). Paris, France, October 2, 2023. 10 pages.

Abstract:

This paper explores feature selection and fusion methods for predicting the clinical outcome of post-stroke aphasia from medical imaging data. Utilizing a multimodal neuroimaging dataset derived from 55 individuals with chronic aphasia resulting from left-hemisphere lesions following a stroke, two distinct approaches, namely Early Fusion and Late Fusion, were developed using Support Vector Regression or Random Forest regression models for prognosticating patients' functional communication skills measured by Western Aphasia Battery (WAB) test scores. A supervised learning method is proposed to reduce the number of features derived from each imaging modality. …

Saurav was a BU MS AI student who graduated in May 2023. Link to paper

Part 4 Learning Objectives: Be able to

- Define holdout cross validation, k-fold cross validation and leave-one-out cross validation
- Explain the difference between a [train/test] and [train/validate/test] cross-validation experiment
- Give typical split values
- Explain how to report the results of a cross-validation experiment
- Define the normalized correlation coefficient
- Explain how to perform feature selection (reduction) based on the normalized correlation coefficient
- Explain nested cross validation