



Machine Learning: Neural Networks

CSE 415: Introduction to Artificial Intelligence
University of Washington
Spring, 2019

© S. Tanimoto and University of Washington, 2019.



Outline

- Introduction to Learning
- Applications of Neural Networks
- The Perceptron
- Training algorithm
- Fundamental training theorem
- Two-level perceptrons
- Feedforward neural nets
- Backpropagation and the Delta rule
- History of neural networks research

Univ. of Wash.

Learning: Neural Nets

2



Applications of Neural Networks

These networks are very popular as trainable classifiers for a wide variety of pattern data.

Examples:

- Speech recognition and synthesis
- Visual texture classification
- Optical character recognition
- Control systems for robot actuators

Univ. of Wash.

Learning: Neural Nets

3



AlphaGo and AlphaStar



Adversarial Search —→ Reinforcement Learning
—→ Deep Reinforcement Learning

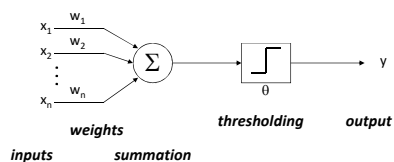
Univ. of Wash.

Learning: Neural Nets

4



The Perceptron



$$y = \begin{cases} 1 & \text{if } \sum w_i x_i \geq \theta; \\ 0, & \text{otherwise.} \end{cases}$$

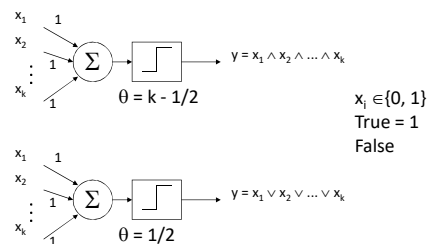
Univ. of Wash.

Learning: Neural Nets

5



Perceptron Examples: Boolean AND and OR.



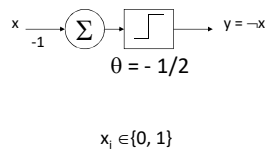
Univ. of Wash.

Learning: Neural Nets

6



Perceptron Examples: Boolean NOT



Univ. of Wash.

Learning: Neural Nets

7



Perceptron Example: Template Matching

-1	-1	1	-1	-1
-1	1	-1	1	-1
1	1	1	1	1
1	-1	-1	-1	1
1	-1	-1	-1	1

$$x_i \in \{-1, 1\}$$

$$\theta = 25 - \varepsilon$$

Recognizes the letter A provided the exact pattern is present.

weights w_1 through w_{25}

Univ. of Wash.

Learning: Neural Nets

8



Perceptron Training Sets

Let $X = X^* \cup X^-$ be the set of training examples.

$S_X = \langle X_1, X_2, \dots, X_k, \dots \rangle$ is a *training sequence* on X , provided:

- (1) Each X_k is a member of X , and
- (2) Each element of X occurs infinitely often in S_X .

This is basically saying: you should have a mix of each training example in your training sequence.

Notice, it is mathematically incorrect to have a sequence with infinite amount of A followed by an infinite amount of B, so this prevents the algorithm from never seeing a training example.

Univ. of Wash.

Learning: Neural Nets

9



Perceptron Training Algorithm

Let $X = X^* \cup X^-$ be the set of training examples. and let $S_X = \langle X_1, X_2, \dots, X_k, \dots \rangle$ be a training sequence on X .

Let w_k be the weight vector at step k .

Choose w_0 arbitrarily. For example. $w_0 = (0, 0, \dots, 0)$.

Each each step k , $k = 0, 1, 2, \dots$

Classify X_k using w_k .

What about θ ?

If X_k is correctly classified, take $w_{k+1} = w_k$.

If X_k is in X^- but misclassified, take $w_{k+1} = w_k - c_k X_k$.

If X_k is in X^* but misclassified, take $w_{k+1} = w_k + c_k X_k$.

The sequence c_k should be chosen according to the data. Overly large constant values can lead to oscillation during training. Values that are too small will increase training time. However, $c_k = c_0/k$ will work for any positive c_0 .

Univ. of Wash.

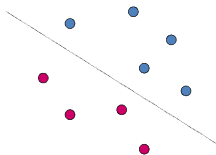
Learning: Neural Nets

10



Convergence

Perceptron training always converges if the training data X^* and X^- are linearly separable sets.



Univ. of Wash.

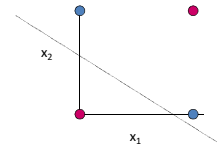
Learning: Neural Nets

11



Perceptron Limitations

The boolean function XOR (exclusive or) is not linearly separable. (Its positive and negative instances cannot be separated by a line or hyperplane.) It cannot be computed by a single-layer perceptron. It cannot be learned by a single-layer perceptron.



$$X^+ = \{(0, 1), (1, 0)\}$$

$$X^- = \{(0, 0), (1, 1)\}$$

$$X = X^+ \cup X^-$$

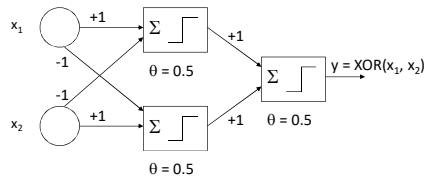
Univ. of Wash.

Learning: Neural Nets

12



Two-Layer Perceptrons



Univ. of Wash.

Learning: Neural Nets

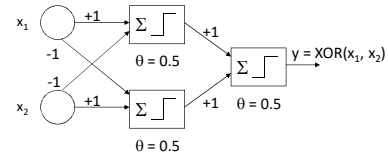
13



Two-Layer Perceptrons (cont.)

Two-Layer perceptrons are computationally powerful.

However: they are not trainable with a method such as the perceptron training algorithm, because we don't know the correct value of the middle layer nodes; there is no way to know what the correct updates to first-layer weights should be.



Univ. of Wash.

Learning: Neural Nets

14



Two-Layer Feedforward Networks with Sigmoid Activation Functions

We get: the power of 2-layer perceptrons, and the trainability of 1-layer perceptrons (well, sort of).

These are sometimes called

- multi-layer perceptrons
- feedforward neural networks
- backpropagation networks (because the training method is called backpropagation)

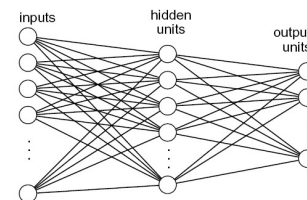
Univ. of Wash.

Learning: Neural Nets

15



Structure of a Backprop. Network



Univ. of Wash.

Learning: Neural Nets

16



Hidden Node Input Activation

As with perceptrons, a weighted sum is computed of values from the previous level:

$$h_j = \sum_i w_{ij} x_i$$

However the hidden node does not apply a threshold, but a sigmoid function ...

Univ. of Wash.

Learning: Neural Nets

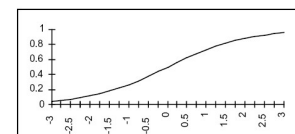
17



Sigmoid Activation Functions

Instead of using threshold functions, which are neither continuous nor differentiable, we use a *sigmoid* function, which is a sort of smoothed threshold function.

$$g_1(h) = 1/(1 + e^{-h})$$



Univ. of Wash.

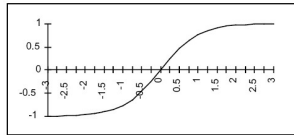
Learning: Neural Nets

18



An Alternative Sigmoid Func.

$$g_2(h) = \tanh(h) = (e^h - e^{-h}) / (e^h + e^{-h})$$



Univ. of Wash.

Learning: Neural Nets

19



Sigmoid Function Properties

Both g_1 and g_2 are continuous and differentiable.

$$g_1(h) = 1 / (1 + e^{-h})$$

$$g_1'(h) = g_1(h) (1 - g_1(h))$$

$$g_2(h) = \tanh(h) = (e^h - e^{-h}) / (e^h + e^{-h})$$

$$g_2'(h) = 1 - g_2(h)^2$$

Univ. of Wash.

Learning: Neural Nets

20



Rectified Linear Unit (ReLU)

- Continuous and sub-differentiable
- $g_3(h) = \max(0, h)$

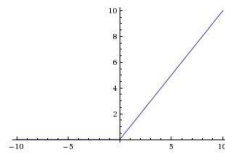
- Why is this preferable?

- $g_3'(h) = 0$ if $h < 0$

$$1 \text{ if } h > 0$$

$$[0, 1] \text{ if } h = 0$$

* Usually 0 or 1



Univ. of Wash.

Learning: Neural Nets

21



Training Algorithm

Each *training example* has the form $\langle X_i, T_i \rangle$, where X_i is the vector of inputs, and T_i is the desired corresponding output vector.

An *epoch* is one pass through the training set, with an adjustment to the networks weights for each training example. (Use the "delta rule" for each example.)

Perform as many epochs of training as needed to reduce the classification error to the required level.

If there are not enough hidden nodes, then training might not converge.

Univ. of Wash.

Learning: Neural Nets

22



Delta Rule

For each training example $\langle X_i, T_i \rangle$, Compute $F(X_i)$, the outputs based on the current weights.

To update a weight w_{ij} , add ∇w_{ij} to it, where

$$\nabla w_{ij} = \eta \delta_j F_j \quad (\eta \text{ is the training rate.})$$

If w_{ij} leads to an output node, then use

$$\delta_j = (t_j - F_j) g'_j(h_j)$$

If w_{ij} leads to a hidden node, then use "backpropagation":

$$\delta_j = g'_j(h_j) \sum_k \delta_k w_{kj}$$

The δ_k in this last formula comes from the output level, as computed above.

Univ. of Wash.

Learning: Neural Nets

23



Performance of Backpropagation

Backpropagation typically takes no longer than 5x of the time for forward inference.

Multilayer perceptrons are slower than 1-layer perceptrons, but are more expressive.

The training rate η can be set large near the beginning and made smaller in later epochs.

In principle, backpropagation can be applied to networks with more than one layer of hidden nodes, but this slows the algorithm much more.

In computer vision, since the nearby pixels in an image have a much stronger correlation than pixels far away, we can just connect the pixels with stronger correlation for the first several layers. Such systems are called *Deep Convolutional Neural Nets*.

Univ. of Wash.

Learning: Neural Nets

24

Le
Learning

Setting the Number of Hidden Nodes

The number of nodes in the hidden layer affects generality and convergence.

If too few hidden nodes: convergence may fail.

Few but not too few nodes: possibly slow convergence but good generalization

Too many hidden nodes: Rapid convergence, but “overfitting” happens.

Overfitting: the learned network handles the training set, but fails to generalize effectively to similar examples not in the training set.

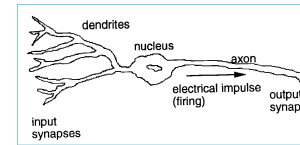
Univ. of Wash.

Learning: Neural Nets

25

Le
Learning

The Biological Neuron



The human brain contains approximately 10^{11} neurons.

Activation process:

Inputs are transmitted electrochemically across the input synapses

Input potentials are summed.

If the sum reaches a threshold, a pulse moves down the axon. (The neuron has “fired”.)

The pulse is distributed at the axonal arborization to the input synapses of other neurons.

After firing, there is a refractory period of inactivity.

Univ. of Wash.

Learning: Neural Nets

26

Le
Learning

History of Neural Networks Research

1943 McCulloch & Pitts model of neuron.

$$n_i(t+1) = \Theta(\sum_j w_{ij} n_j(t) - \mu_i), \quad \Theta(x) = \begin{cases} 1 & \text{if } x \geq 0; \\ 0, & \text{otherwise.} \end{cases}$$

1962 Frank Rosenblatt’s book gives a training algorithm for finding the weights w_{ij} from examples.

Principles of neurodynamics: Perceptrons and the theory of brain mechanisms

1969 Marvin Minsky and Seymour Papert publish *Perceptrons*, and prove that 1-layer perceptrons are incapable of computing image connectedness.

1974-89, 1982: Associated content-addressable memory.

Backpropagation: Werbos 1974, Parker 1985, Rumelhart, Hinton, & Williams 1986.

Univ. of Wash.

Learning: Neural Nets

27