

Neural Networks

CMSC 422

Slides adapted from Prof. CARPUAT

Neural Networks

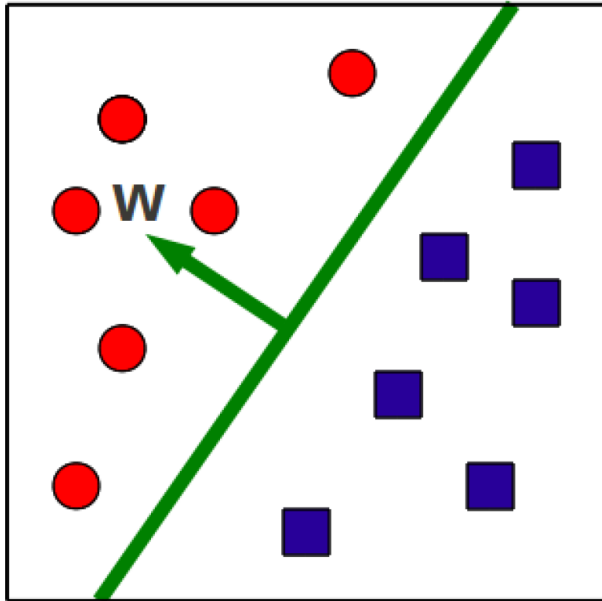
- Today
 - What are Neural Networks?
 - How to make a prediction given an input?
 - Why are neural networks powerful?
- Thursday
 - how to train them?

A warm-up example

sentiment analysis for movie review

- the movie was horrible +1
- the actors are excellent -1
- the movie was not horrible -1
- he is usually an excellent actor, but not in this movie +1

Binary classification via hyperplanes



- At test time, we check on what side of the hyperplane examples fall

$$\hat{y} = \text{sign}(w^T x + b)$$

Function Approximation with Perceptron

Problem setting

- Set of possible instances X
 - Each instance $x \in X$ is a feature vector $x = [x_1, \dots, x_D]$
- Unknown target function $f: X \rightarrow Y$
 - Y is binary valued $\{-1; +1\}$
- Set of function hypotheses $H = \{h \mid h: X \rightarrow Y\}$
 - Each hypothesis h is a hyperplane in D -dimensional space

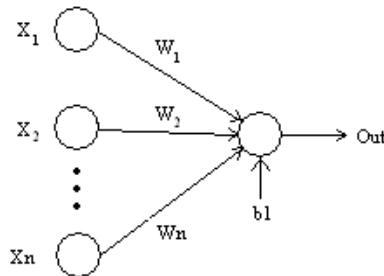
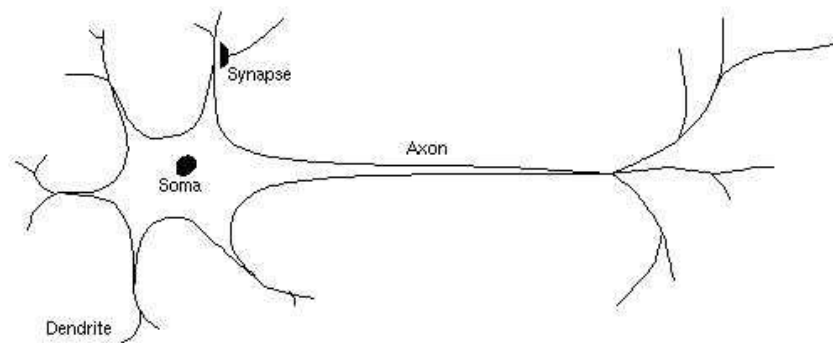
Input

- Training examples $\{(x^{(1)}, y^{(1)}), \dots, (x^{(N)}, y^{(N)})\}$ of unknown target function f

Output

- Hypothesis $h \in H$ that best approximates target function f

Aside: biological inspiration



Analogy: the
perceptron
as a neuron

Neural Networks

- We can think of neural networks as combination of multiple perceptrons
 - Multilayer perceptron
- Why would we want to do that?
 - Discover more complex decision boundaries
 - Learn combinations of features

What does a 2-layer perceptron look like?

(illustration on board)

- Key concepts:
 - Input dimensionality
 - Hidden units
 - Hidden layer
 - Output layer
 - Activation functions

Activation functions

- Activation functions are **non-linear** functions
 - sign function as in the perceptron
 - hyperbolic tangent and other sigmoid functions that approximate sign but are differentiable
- What happens if the hidden units use the identity function as an activation function?

Matrix of hidden layer
parameters

Vector of output layer
parameters

Algorithm 24 `TWO_LAYER_NETWORK_PREDICT`(\mathbf{W} , v , \hat{x})

1: **for** $i = 1$ **to** *number of hidden units* **do**

2: $h_i \leftarrow \tanh(w_i \cdot \hat{x})$

// compute activation of hidden unit i

3: **end for**

4: **return** $v \cdot h$

// compute output unit

What functions can we approximate with a 2 layer perceptron?

Problem setting

- Set of possible instances X
 - Each instance $x \in X$ is a feature vector $x = [x_1, \dots, x_D]$
- Unknown target function $f: X \rightarrow Y$
 - Y is binary valued $\{-1; +1\}$
- **Set of function hypotheses** $H = \{h \mid h: X \rightarrow Y\}$

Input

- Training examples $\{(x^{(1)}, y^{(1)}), \dots, (x^{(N)}, y^{(N)})\}$ of unknown target function f

Output

- Hypothesis $h \in H$ that best approximates target function f

Two-Layer Networks are Universal Function Approximators

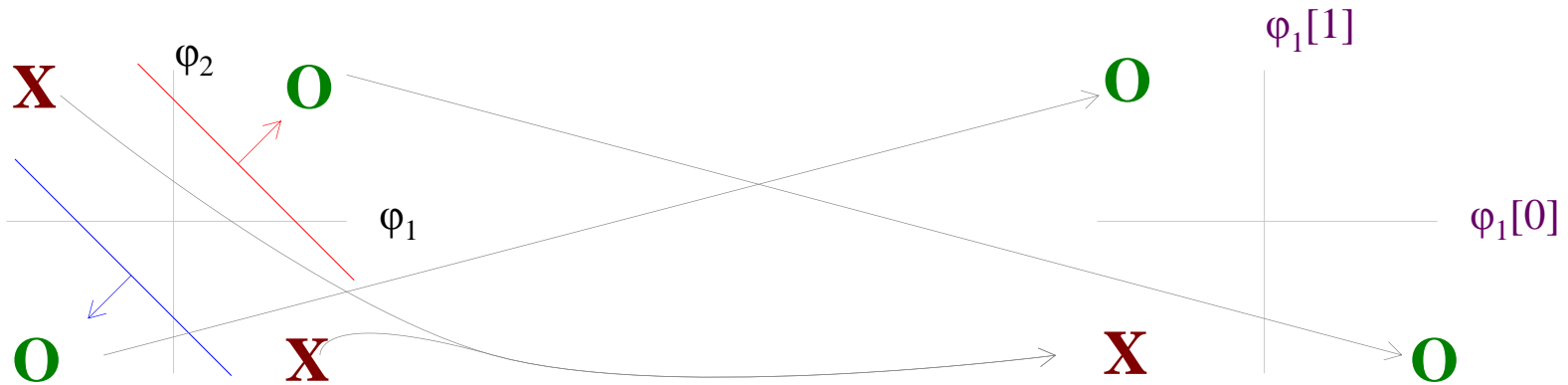
- Theorem (Th 9 in CIML):

Let F be a continuous function on a bounded subset of D -dimensional space. Then there exists a two-layer neural network \hat{F} with a finite number of hidden units that approximates F arbitrarily well. Namely, for all x in the domain of F , $|F(x) - \hat{F}(x)| < \epsilon$.

Example: a neural network to solve the XOR problem

$$\varphi_0(\mathbf{x}_1) = \{-1, 1\} \quad \varphi_0(\mathbf{x}_2) = \{1, 1\}$$

$$\varphi_1(\mathbf{x}_3) = \{-1, 1\}$$



$$\varphi_0(\mathbf{x}_3) = \{-1, -1\} \quad \varphi_0(\mathbf{x}_4) = \{1, -1\}$$

$$\varphi_1(\mathbf{x}_1) = \{-1, -1\}$$

$$\varphi_1(\mathbf{x}_2) = \{1, -1\}$$

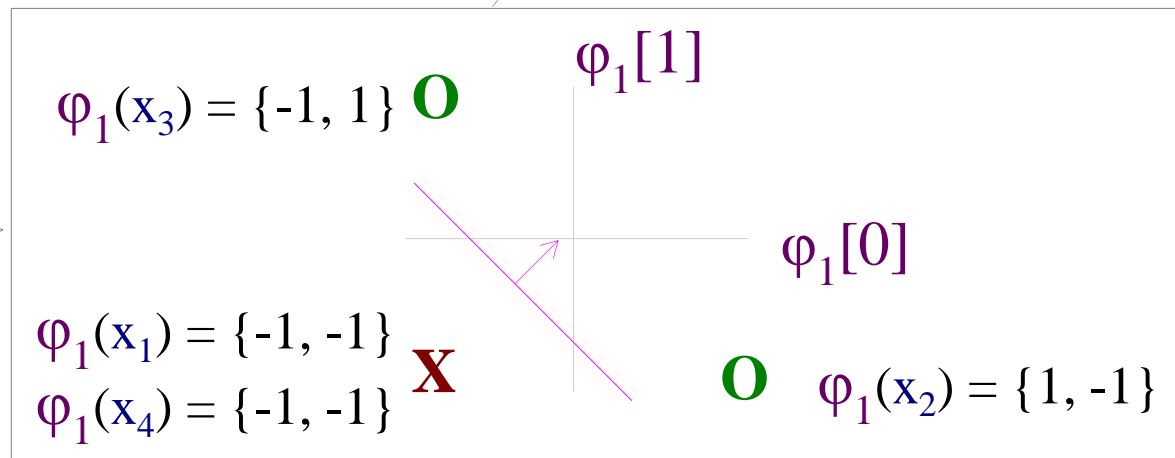
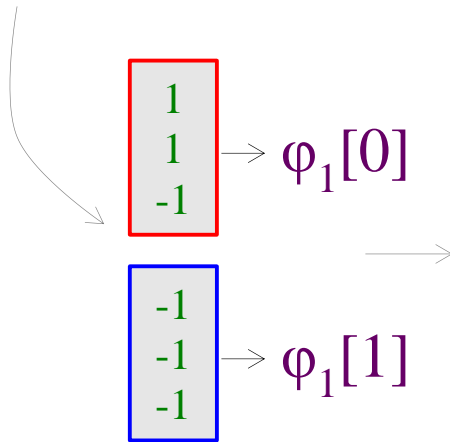
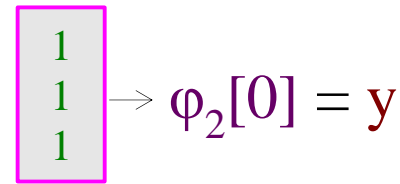
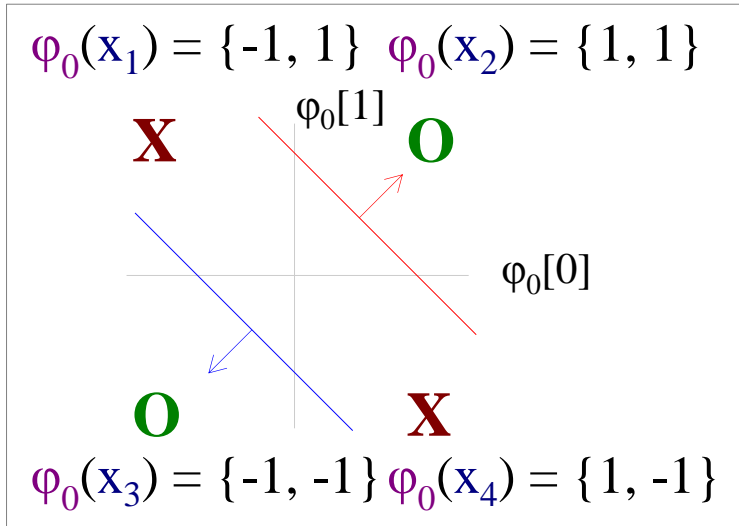
$$\varphi_1(\mathbf{x}_4) = \{-1, -1\}$$

$$\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \rightarrow \varphi_1[0]$$

$$\begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \rightarrow \varphi_1[1]$$

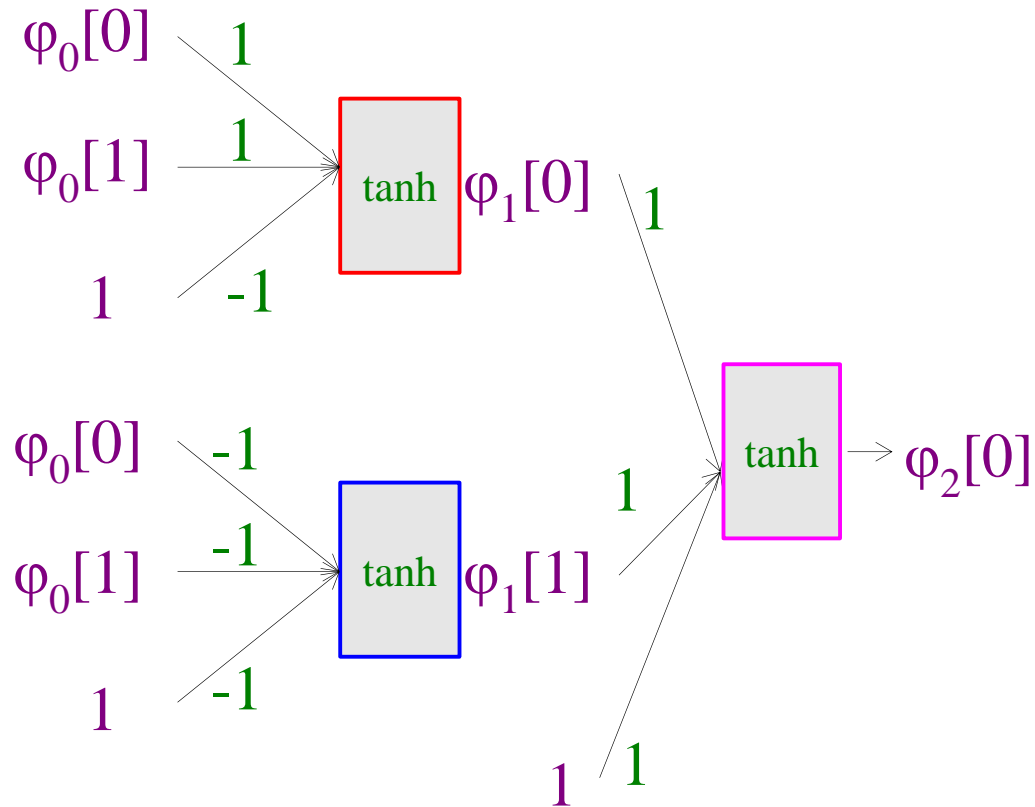
Example

- In new space, the examples are linearly separable!



Example

- The final net



Discussion

- 2-layer perceptron lets us
 - Discover more complex decision boundaries than perceptron
 - Learn combinations of features that are useful for classification
- Key design question
 - How many hidden units?
 - More hidden units yield more complex functions
 - Fewer hidden units requires fewer examples to train

Neural Networks

- Today
 - What are Neural Networks?
 - Multilayer perceptron
 - How to make a prediction given an input?
 - Simple matrix operations + non-linearities
 - Why are neural networks powerful?
 - Universal function approximators!
- Thursday
 - how to train them?