Machine Learning CS 4641



## Neural Networks Introduction

Nakul Gopalan Georgia Tech

These slides are from Vivek Srikumar, Mahdi Roozbahani and Chao Zhang.

#### Outline

Perceptron



- Stacking Linear Threshold Units
- Neural Networks
- Expressivity of Neural Networks
- Predicting with Neural Networks \_\_\_\_\_\_ Forward Propagation
- Backpropogation

#### Linear Classifiers

- Input is a n dimensional vector **x**
- Output is a label  $y \in \{-1, 1\}$
- Linear Threshold Units (LTUs) classify an example x using the following classification rule

- Output = sgn( $\mathbf{w}^T \mathbf{x} + \mathbf{b}$ ) = sgn( $\mathbf{b} + \sum w_i x_i$ )

$$- \mathbf{w}^{\mathsf{T}}\mathbf{x} + \mathbf{b} \ge \mathbf{0} \rightarrow \text{Predict } \mathbf{y} = \mathbf{1}$$

$$- \mathbf{w}^{\mathsf{T}}\mathbf{x} + \mathbf{b} < \mathbf{0} \rightarrow \text{Predict } \mathbf{y} = -1$$

b is called the bias term

#### Linear Classifiers

42MX +3 0

5 MMX + C

y= mx -1

1=0 .0000

- Input is a n dimensional vector x
- Output is a label  $y \in \{-1, 1\}$
- Linear Threshold Units (LTUs) classify an example x using the following classification rule



#### Perceptron

- Rosenblatt 1958
- The goal is to find a separating hyperplane
  - For separable data, guaranteed to find one
- An online algorithm
   Processes one example at a time
- Several variants exist (will discuss briefly at towards the end)

These slides are from Vivek Srikumar

# Perceptron Algorithm

Input: A sequence of training examples  $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \cdots$   $(\mathbf{y}_1, \mathbf{y}_2)$ where all  $x_i \in \Re^n$ ,  $y_i \in \{-1, 1\}$ 

- Initialize  $\mathbf{w}_0 = \mathbf{0} \in \Re^n$
- For each training example (**x**<sub>i</sub>, y<sub>i</sub>):

  - Predict  $y' = sgn(\mathbf{w}_t^T \mathbf{x}_i)$  If  $y_i \neq y'$ : Update  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + r(y_i \mathbf{x}_i)$
- Return final weight vector

$$5ign/5gn(n) - 5 + 1 : 250$$
  
-1: X40

These slides are from Vivek Srikumar

Mistake on positive:  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{r} \mathbf{x}_i$ Mistake on negative:  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{r} \mathbf{x}_i$ 

r is the learning rate, a small positive number less than 1

Update only on error. A mistakedriven algorithm

This is the simplest version. We will see more robust versions at the end

Mistake can be written as  $y_i \mathbf{w}_t^T \mathbf{x}_i \leq 0$ 

Mistake on positive:  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + r \mathbf{x}_i$ Mistake on negative:  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - r \mathbf{x}_i$ 

#### Geometry of the perceptron update



# For a mistake on a positive example

These slides are from Vivek Srikumar

#### Poll

- Perceptron as described:
  - For solving linear boundaries ---> 66 1
  - Can solve non-linear boundaries -7 17.
  - Can solve non-linear boundaries with non-linear features  $\rightarrow 17 1$

Y-OR

#### Outline

- Perceptron
- Stacking Linear Threshold Units
- Neural Networks
- Expressivity of Neural Networks
- Predicting with Neural Networks
- Backpropogation

#### Linear Threshold Unit



Prediction

 $sgn(\mathbf{w}^T\mathbf{x} + b) = sgn(\sum w_i x_i + b)$ 

#### Learning

various algorithms perceptron, SVM, logistic regression,...

in general, minimize loss

features

But where do these input features come from?

What if the features were outputs of another classifier?

#### Features for Linear Threshold Unit



Each of these connections have their own weights as well





This is a two layer feed forward neural network





Five neurons in this picture (four in hidden layer and one output)



The input layer

What if the inputs were the outputs of a classifier?

We can make a **three** layer network.... And so on.

#### Outline

- Perceptron
- Stacking Linear Threshold Units
- Neural Networks



- Expressivity of Neural Networks
- Predicting with Neural Networks
- Backpropogation



- A robust approach for approximating real-valued, discrete-valued or vector valued functions
- Among the most effective general purpose supervised learning methods currently known
  - Especially for complex and hard to interpret data such as realworld sensory data
- The Backpropagation algorithm for neural networks has been shown successful in many practical problems
  - handwritten character recognition, speech recognition, object recognition, some NLP problems

#### Inspiration from Biological Neurons



The first drawing of a brain cells by Santiago Ramón y Cajal in 1899 Neurons: core components of brain and the nervous system consisting of

- 1. Dendrites that collect information from other neurons
- 2. An axon that generates outgoing spikes

### **Artificial Neurons**

Functions that very loosely mimic a biological neuron

A neuron accepts a collection of inputs (a vector **x**) and produce: an output by:

- 1. Applying a dot product with weights w and adding a bias b
- 2. Applying a (possibly non-linear) transformation called an *activation*

 $output = activation(\mathbf{w}^T \mathbf{x} + b)$ 



#### **Activation Functions**

 $output = activation(\mathbf{w}^T \mathbf{x} + b)$ 

Y1

<b>Activation function:</b> <i>activation</i> ( <i>z</i> )
Z
sgn(z)
$\frac{1}{1 + \exp(-z)}$
max (0, <i>z</i> )
tanh (z)

Y

#### **Neural Network**

A function that converts inputs to outputs defined by a directed acyclic graph

- Nodes organized in layers, correspond to neurons
- Edges carry output of one neuron to another, associated with weights
- To define a neural network, we need to specify:
  - The structure of the graph
    - How many nodes, the connectivity
  - The activation function on each node
  - The edge weights



#### Neural Network

A function that converts inputs to outputs defined by a directed acyclic graph

- Nodes organized in layers, correspond to neurons
- Edges carry output of one neuron to another, associated with weights
- To define a neural network, we need to specify:
  - The structure of the graph
    - How many nodes, the connectivity
  - The activation function on each node

The edge weights



Called the <u>architecture</u> of the network Typically predefined, part of the design of the classifier

Learned from data

#### A Brief History of Neural Network

- 1943: McCullough and Pitts showed how linear threshold units can compute logical functions
- 1949: Hebb suggested a learning rule that has some physiological plausibility
- 1950s: Rosenblatt, the Peceptron algorithm for a single threshold neuron
- 1969: Minsky and Papert studied the neuron from a geometrical perspective
- 1980s: Convolutional neural networks (Fukushima, LeCun), the backpropagation algorithm (various)
- 2003-today: More compute, more data, deeper networks

#### Outline

- Perceptron
- Stacking Linear Threshold Units
- Neural Networks
- Expressivity of Neural Networks
- Predicting with Neural Networks
- Backpropogation

#### A Single Neuron with Threshold Activation

Prediction =  $sgn(b + w_1 x_1 + w_2 x_2)$ 



#### Two Layers with Threshold Activation



Figure from [Shai Shalev-Shwartz and Shai Ben-David, 2014]

#### Three Layers with Threshold Activation



In general, unions of convex polygons

20

Figure from [Shai Shalev-Shwartz and Shai Ben-David, 2014]

#### NNs are Universal Function Approximators

- Any continuous function can be approximated to arbitrary accuracy using one hidden layer of sigmoid units [Cybenko 1989]
- Approximation error is insensitive to the choice of activation functions [DasGupta et al 1993]

#### Outline

- Perceptron
- Stacking Linear Threshold Units
- Neural Networks



 $\hat{y}!=y=\omega^{\dagger n}=\omega^{\dagger}+v(y,x)$ 

- Expressivity of Neural Networks
- Predicting with Neural Networks
- Backpropogation

#### Predicting with Neural Networks



#### Predicting with Neural Networks



#### Predicting with Neural Networks



#### Predicting with Neural Nets: The Forward Pass

Given an input **x**, how is the output predicted



#### The Forward Pass

Given an input x, how is the output predicted



$$z_1 = \sigma(w_{01}^{h_1} + w_{11}^{h_2} x_1 + w_{21}^{h_2} x_2)$$

#### The Forward Pass

Given an input x, how is the output predicted



$$z_{2} = \sigma(w_{02}^{h} + w_{12}^{h}x_{1} + w_{22}^{h}x_{2})$$
$$z_{1} = \sigma(w_{01}^{h} + w_{11}^{h}x_{1} + w_{21}^{h}x_{2})$$

#### **The Forward Pass**



#### Outline

- Perceptron
- Stacking Linear Threshold Units
- Neural Networks
- Expressivity of Neural Networks
- Predicting with Neural Networks
- Backpropogation



#### Backpropogation

- Smarter chain rule for derivatives
- What is chain rule:
   Univariate case:

$$\frac{\mathrm{d}}{\mathrm{d}t}f(x(t)) = \frac{\mathrm{d}f}{\mathrm{d}x} \cdot \frac{\mathrm{d}x}{\mathrm{d}t}$$

Multivariate case (Partial derivatives):

$$\frac{\mathrm{d}}{\mathrm{d}t}f(x(t),y(t)) = \frac{\partial f}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}t} + \frac{\partial f}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t}$$

$$\int (g,y) = e^{x} + e^{y}$$

$$\chi = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} + \frac{\partial f}{\partial$$



These slides are from Roger Grosse

$$Sin(H^{2})$$

$$Sin(\pi) = \mathcal{A}zt^{2}$$

$$\frac{dSin(\pi)}{dt} = \frac{dSinGn}{d\pi} \cdot \frac{dx}{dT}$$

$$z \quad Cos(\pi) \cdot 2 \cdot f$$

#### Backpropogation



These slides are from Matt Gromley



These slides are from Matt Gromley

#### Backpropogtion



These slides are from Matt Gromley

#### Backpropagation

- Backprop is used to train the overwhelming majority of neural nets today.
  - Even optimization algorithms much fancier than gradient descent (e.g. second-order methods) use backprop to compute the gradients.
- Despite its practical success, backprop is believed to be neurally implausible. No evidence for biological signals analogous to error derivatives.
  - All the biologically plausible alternatives we know about learn much more slowly (on computers). So how on earth does the brain learn?

#### Take-Home Messages

- Stacking Linear Threshold Units
- Neural Networks
- Expressivity of Neural Networks
- Predicting with Neural Networks (Sound > 10).
- Backprop is chain rule with some book-keeping