CMSC 373 Artificial Intelligence Fall 2023 12-Perceptrons

Deepak Kumar
Bryn Mawr College

1

Review

- “Anarchy of Methods”
- “Easy Things Are Hard”
- Connectionism
- “Bad at Logic, Good at Frisbee”
- GOFAI

2

2

Review

- What is the McCulloch Pitts Neuron?
- What is a Perceptron?
- How does a Perceptron improve on the McCulloch-Pitts Neuron?

3

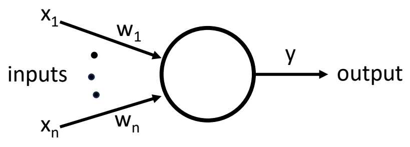
3

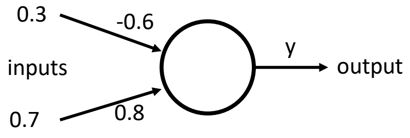
The Perceptron

- A single unit.
- Transfer Function (assume $T = 0$)

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

$$y = \begin{cases} +1, \text{if } I \geq T \\ -1, \text{if } I < T \end{cases}$$





$I = w_1x_1 + w_2x_2$
$I = -0.6 * 0.3 + 0.8 * 0.7$
$I = -0.18 + 0.56 = 0.38$
Since I = 0.38 > (T=0)
$y = +1$

4

4

Perceptron Learning Rule

- Changes the weights

$\overline{\boldsymbol{w}} = [w_1, w_2]$	weight vector
$\overline{\boldsymbol{x}} = [x_1, x_2]$	input vector
$\overline{\boldsymbol{w_{new}}} = \overline{\boldsymbol{w_{old}}} - y^* \, \overline{\boldsymbol{x}}$	Training Rule

5

5

Vocabulary - Perceptron

- **Labelled training Dataset**
  N samples/patterns/input vector with desired outputs (targets/labels)
- **Output Error (Loss)**
  y is the perceptron's answer/output

$$\boldsymbol{\beta} = \begin{cases} +\mathbf{1}, \text{if perceptron's answer is correct} \\ -\mathbf{1}, \text{if perceptron's answer is wrong} \end{cases}$$

- **Learning Rule**
  Specifies change in the weights using the Error
  Perceptron Learning Rule: $\overline{\boldsymbol{w_{new}}} = \overline{\boldsymbol{w_{old}}} + \beta y^* \, \overline{\boldsymbol{x}}$
- **Prediction/Forward Pass**
  Application of a pattern to produce output
- **Epoch**
  1 pass through the training dataset

6

6

Perceptron Training Algorithm

```
Initialize all weights to random values
    #In what range? Typically [-1.0..1.0]
Set #Epochs to some N
    // How to decide what N should be?
Do N times or until all outputs are correct
    Do for each pattern in the training set
        apply the pattern to the perceptron
        change the weight vector as defined
```

7

7

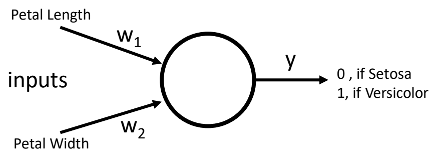
Example – Iris Dataset

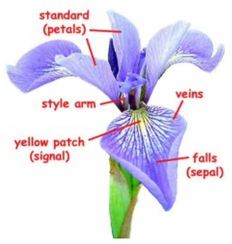
- 150 Samples, 50 of each variety

Sepal Length & Width

```
[5.1, 3.5, 1.4, 0.2],
[4.9, 3.0, 1.4, 0.2],
[4.7, 3.2, 1.3, 0.2],
[4.6, 3.1, 1.5, 0.2],
...
```

Petal Length & Width





From: https://www.fs.usda.gov/wildflowers/beauty/iris/flower.shtml

https://peaceadegbite1.medium.com/iris-flower-classification-60790e9718a1

8

8

Introducing Google Colab

- Live demo…

- Writing a Perceptron from scratch…

9

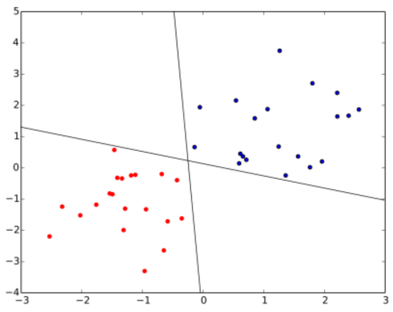
What does a Perceptron Learn?

- A Perceptron is a **binary classifier**

- Given a set of data samples/observations
  e.g., a two-dimensional dataset is shown
  The perceptron decides whether a given point is in one class/category or another
  (red, or blue in the example).

- Geometrically, it tries to find a linear boundary that separates the two classes of data. That is, the **dataset must be linearly separable**.

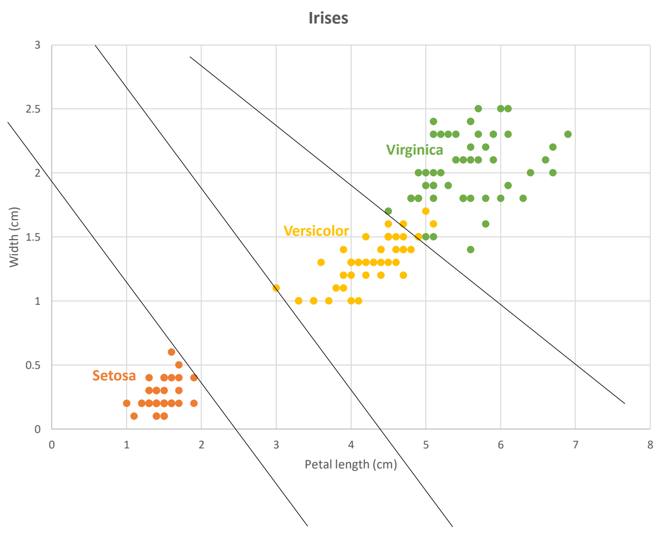
- If the dataset is linearly separable, the Perceptron training algorithm, together with the learning rule, is guaranteed to find a line separating the two classes. There are an infinite number of lines!



From: https://en.wikipedia.org/wiki/Perceptron

10

Classifying Irises



11

11

More Vocabulary

- **Parameters**

  Total number of weights in a network are called parameters

- **Hyperparameters**

  (External) variables used to manage the training/learning

- **Training & Testing Dataset**

  Training Dataset: Dataset the network is trained on
  Testing Dataset: Dataset used to test how well the network has learned

12

12

Another Example – MNIST Dataset

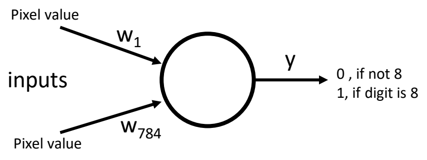
- 70,000 images of handwritten digits
- Each image is 28x28 pixels
- Each pixel is in the range [0..255]
  0 = white, 255 = black. Greys in between.
- Training set: 60,000 images
- Testing set: 10,000 images
- Task: Given an image, classify it as [0,1,…,9]

13

13

Another Example – MNIST Dataset

- Using the Perceptron (an 8 detector)
- Can only use it as a detector for two digits
  Or simply it is a specific digit (say 8) or not.
- Typically, scale the pixel values to the range [0.0..1.0]
- There will be 28x28 = 784 inputs



14

14

Limitations of Perceptrons

- Limited to binary classification tasks only
- *Perceptrons*, by Minsky & Papert, 1969

  Types of problems Perceptrons could solve were limited to linearly separable problems. Real world problems are not linearly separable.

  Perceptron Learning Algorithm would not scale up to tasks requiring large number of weights and thresholds.

  For networks with three or more layers there is no obvious way of knowing what the desired output of hidden layers should be.

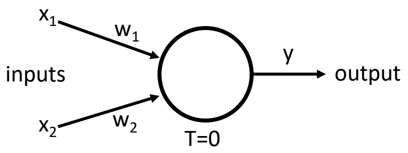
  There is no training procedure possible for networks of three or more layers.

  Led to drying up of funding in neural network research in the 1970s.

Happy Ending

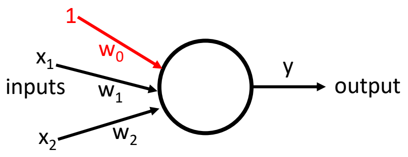
- Minsky & Papert were wrong.
- Researchers chipped away at the linear/non-linear problem. And, developed a learning algorithm for multi-layer networks!
- The result: **Backpropagation**
- Forms the core of nearly all Deep Learning models and their success.

Introducing Bias



- Instead of using an arbitrary Threshold value, we can turn it into an input (=1)
- The weight on the bias, w_0 can then be learned using the same algorithm.

$$\overline{\boldsymbol{w}} = [w_0, w_1, w_2]$$
$$\overline{\boldsymbol{x}} = [x_0, x_1, x_2]$$



- More often, in other networks, the net input is determined using the following (and no bias is used for output layer):

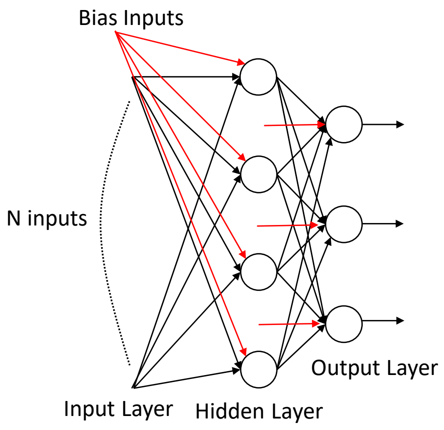
$$I = \sum_{i=1}^{i=n} w_i x_i + \overline{\boldsymbol{b}}$$

17

17

Multi-Layer Perceptron Network

- A 3-layer MLP, 3-output network with 4 hidden units
- Each unit shown is a TLU, with bias inputs
- All N inputs are connected to all 4 hidden units
- All hidden units are connected to all 3 output units.
- **Example:** This could be a network that can recognize all three categories of irises from the Iris dataset.



18

18

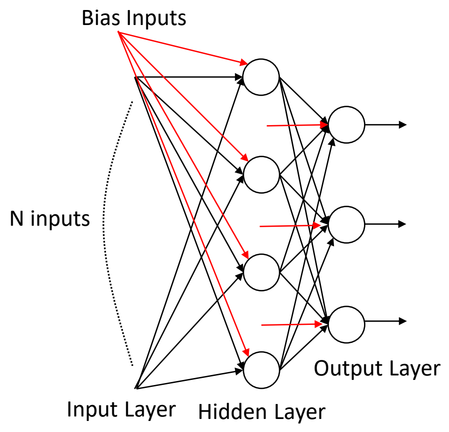
Multi-Layer Perceptron Network

- **Example:** This could be a network that can recognize all three categories of irises from the Iris dataset.

  4 inputs, 3 outputs (Hyperparameters)
  4x4 (input to hidden) + 4x3 (hidden to output) weights + 7 bias inputs

  #Parameters = 16+12+7 = 35

- Since all units are linear TLUs this network can only learn linear functions.
- We need to make each unit non-linear.

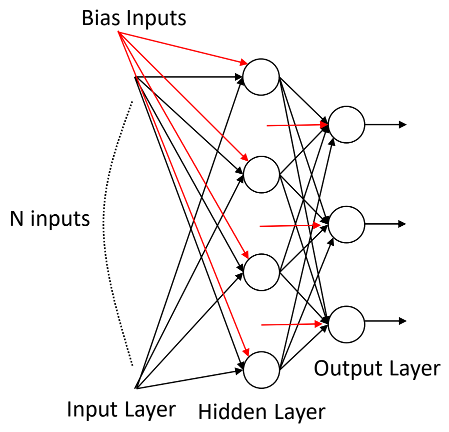


19

19

Multi-Layer Perceptron Network

- Since all units are linear TLUs this network can only learn linear functions.
- We need to make each unit non-linear. This is done by using a different **Activation Function**.
- Need to define a **Learning Rule** that can update weights on hidden layer units.



20

20

References

- M. Caudill and C. Butler: Understanding Neural Networks, Volume 1, MIT Press, 1993.
- F. Chollet: *Deep Learning with Python, Second Edition*, Manning2021.
- A Geron: *Hands-on Machine Learning with SciKit-Learn, Keras and TensorFlow*, Oreilly, 2019.
- M. Mitchell: *Artificial Intelligence: A Guide For Thinking Humans*, Farrar, Strouss, Giroux, 2019.
- M. Wooldridge: *A Brief History of Artificial Intelligence*. Flatiron Books, 2020.