# *Neural Networks – Part II*

## *PHYS 250 (Autumn 2024) – Lecture 15*

David Miller

Department of Physics and the Enrico Fermi Institute
University of Chicago

November 21, 2024

# *Outline*

# *Reminders from last time*

We embarked on a whirlwind introduction to neural networks.

**Neural networks and machine learning**

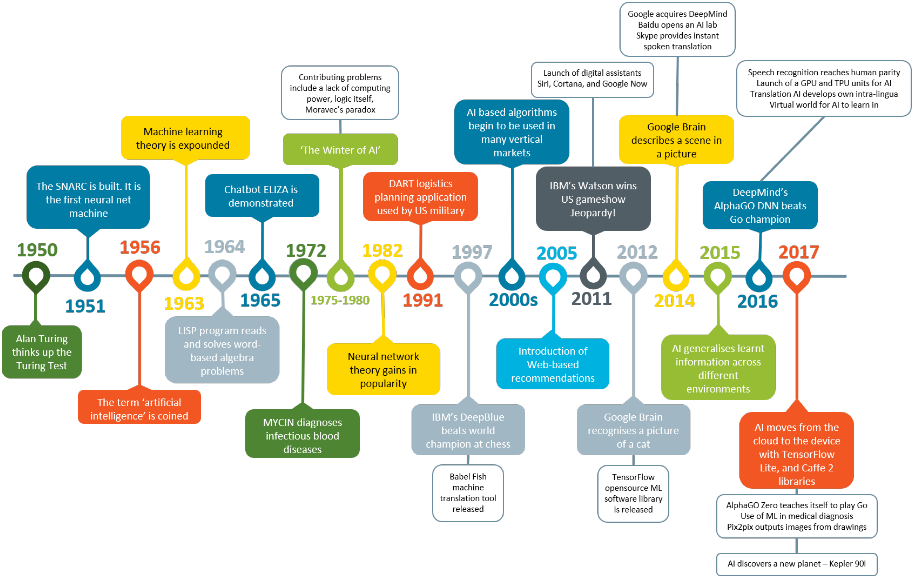
- **Context and perspective**
  - We discussed the general issue of training computers to **discover, identify, and analyze patterns** of interest in datasets
  - Categorized tasks that make use of this idea: **classification, regression, generation, clustering, anomaly detection**
- **Neural networks as a tool**
  - Introduced both the **modeling** perspective as well as the **biological** perspective on what a neural network achieves
  - Described the **structure and function** of a neuron
  - Began discussing the **mathematical properties** of a neural network

Today we will build our own networks! But first, I just wanted to follow-up on some points and questions from last time.

# *Outline*

# *Brief history of machine learning*

Taken from Harry Ide on InnovationLaboratory.com (18 May 2018):



# *Brief history of machine learning*

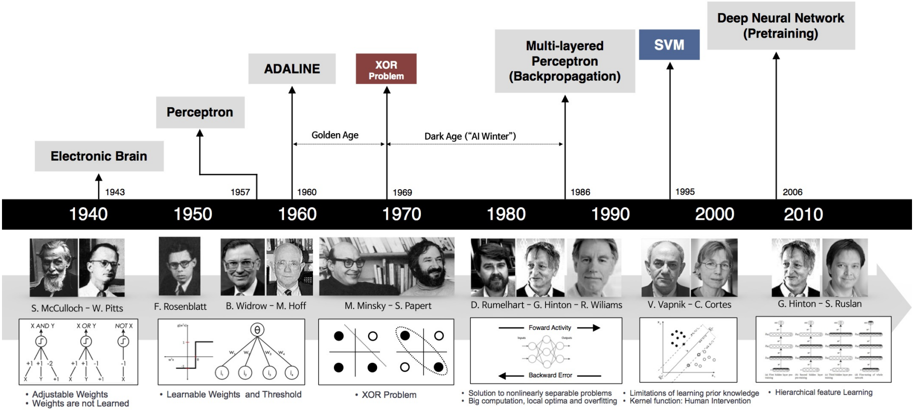
Taken from Harry Ide on InnovationLaboratory.com (18 May 2018):


## Quote from Harry Ide on "AI"

*"The cutting edge might not be as sharp as many believe. AI has a nearly 70-year long history with some surprising back-tracks and re-emergences among it."*

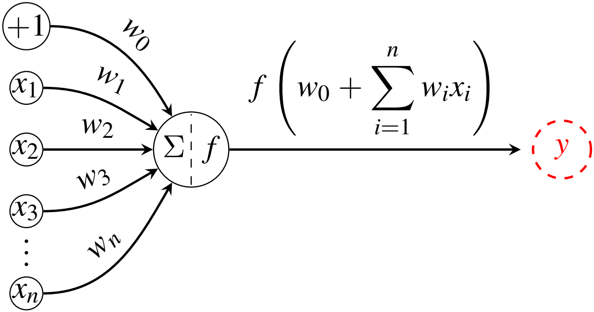
# *Brief history of neural networks*

Taken from this talk on SlideShare:



# *Outline*

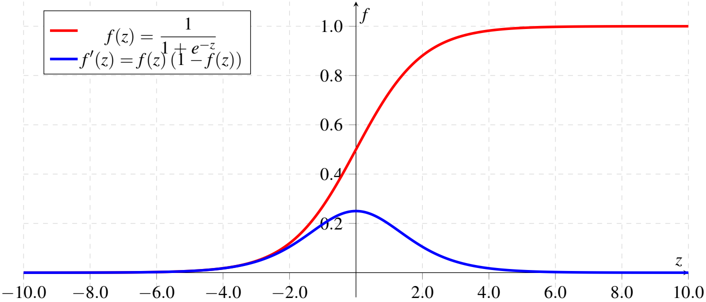
# *Single layer perceptron*




- $\vec{x} = (x_1, x_2, \cdots, x_n)$ **is an input feature vector of length** $n$
  i.e. the attributes of the data, e.g. voltages
- $\vec{w} = (w_1, w_2, \cdots, w_n)$ **is the weight vector with** $w_0$ **reserved as a bias**
  - becomes a matrix for multiple layers
- $\Sigma$ **indicates summation (or matrix mult.):** $z = \sum w_i x_i \ (x_0 = 1)$
- $f$ **is the activation function, or non-linearity:** $f(z)$
- $y = f(z)$ **is the output**

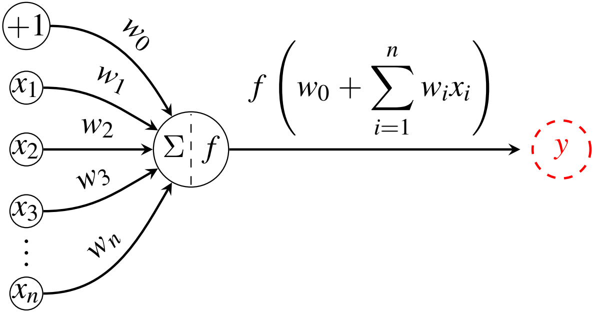
# *Sigmoid as activation function*

As we discussed, a typical function for a **single layer perceptron** is the **sigmoid**.




Here, we plot both the function itself, as well as its derivative, since that will be important when evaluating the **backpropagation** of weights in order to update the neural network.

# *Training a single layer perceptron*

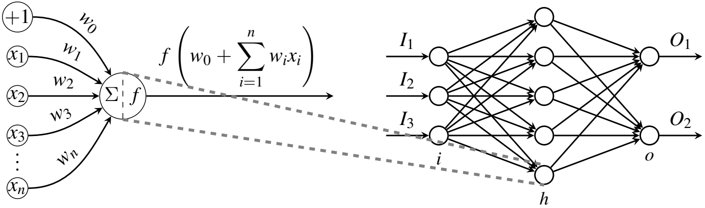



Given $j$ objects $\vec{x}_j$ in dataset, each with **known values of $f$**, $d_j$

- **Calculate the output:** $y_j = f(\vec{w} \cdot \vec{x}_j)$
- **Determine the error:** $\epsilon_j = d_j - y_j$
- **Update the weights:** $w_i^{\text{new}} = w_i + r(\epsilon_j \cdot \vec{x}_j)_i$

Choosing the learning rate $r$ is where the derivative is used. It's not important for the single-layer perceptron, but is **essential** for a network.

# *Multi-layer perceptron (MLP)*

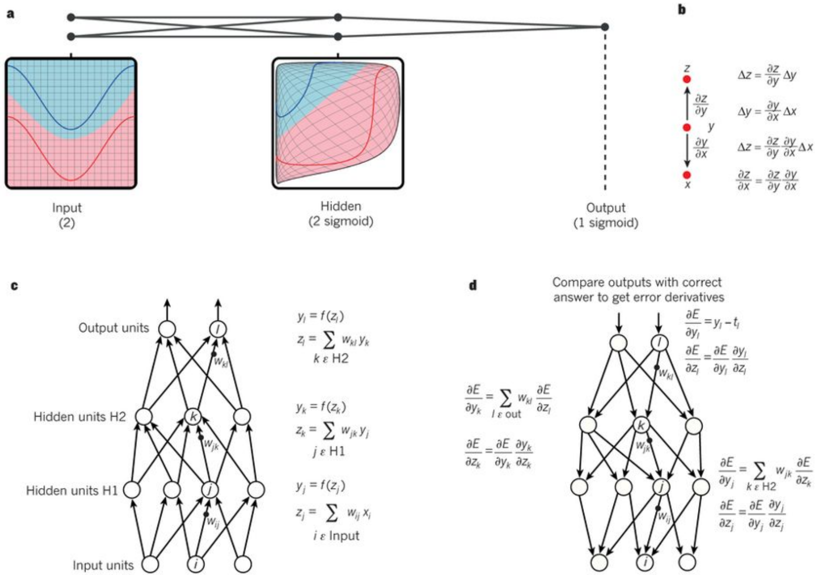



Given $j$ objects $\vec{I}_j$ in dataset, each with features $\vec{I} = (I_1, I_2, \cdots, I_n)$ and **known outputs $\vec{d}_j$ at each output node $o$**, $\vec{d} = (d_1, d_2, \cdots, d_o)$

- **Calculate the $h$ outputs of hidden layer:** $v_h = f(\sum_i w_{ih} I_i)$
- **Calculate the $o$ outputs of output layer:** $y_o = f(\sum_h w_{ho} v_h)$
- **Determine the error at output each node $o$:** $\epsilon_o = d_o - y_o$
- **Determine the total error for data object $j$:** $\mathcal{E}_j = \frac{1}{2} \sum_o \epsilon_o^2$
- **Determine change in weights for output neuron $y_o$:** $\Delta w_{oh} = -\eta \frac{\partial \mathcal{E}}{\partial z_o} v_h = \eta \epsilon_o f'(z_o)$

# *LeCun, Bengio, Hinton, "Deep learning"*

*Nature volume 521, pages 436-444 (28 May 2015)*



# *Outline*

# *What is classification?*

*Slides stolen from colleague Ben Nachmann*

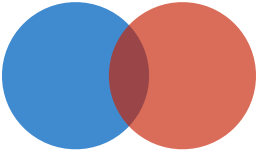
## Classification

Goal: Given a *feature vector*, return an integer indexed by the set of possible *classes.*

In most cases, we care about *binary* classification in which there are only two classes (signal versus background)

There are some cases where we care about *multi-class classification*

Feature vector can be many-dimensional



Harder = more overlap between for **S** and **B**

# *What is classification?*

## Classification

Goal: Given a *feature vector*, return an integer indexed by the set of possible *classes.*

In practice, we don't just want one classifier, but an entire set of classifiers indexed by:

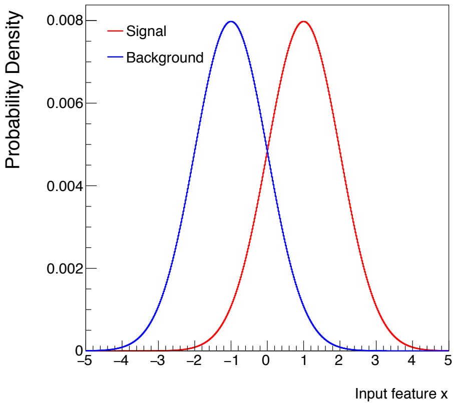
**True Positive Rate** = signal efficiency = Pr(label signal | signal) = sensitivity

**True Negative Rate** = 1 - background efficiency = rejection = Pr(label background | background) = specificity

*For a given TPR, we want the lowest possible TNR!*

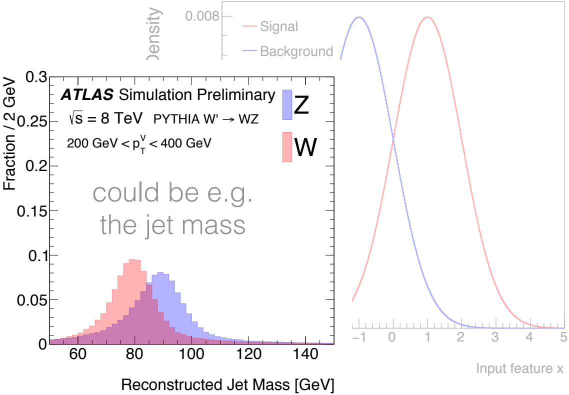
# *Binary classification (I)*

Let's consider an important special case:
binary classification in 1D



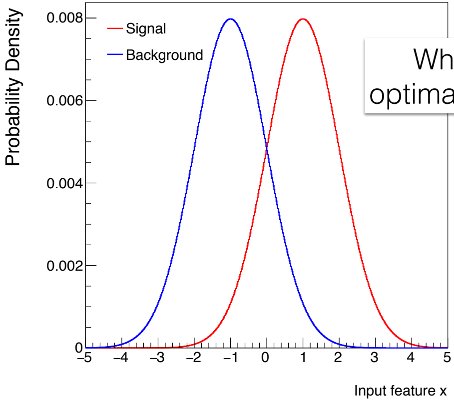
# *Binary classification (II)*

Let's consider an important special case:
binary classification in 1D



# *Binary classification (III)*

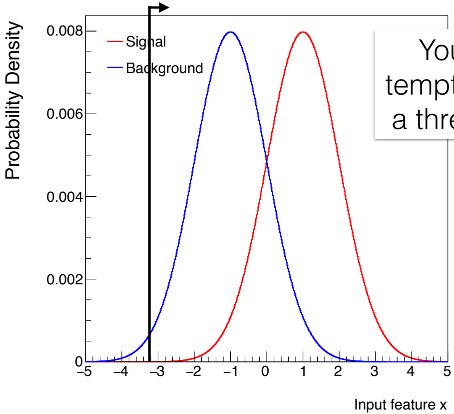
Let's consider an important special case:
binary classification in 1D




What is the optimal classifier?

# *Binary classification (IV)*

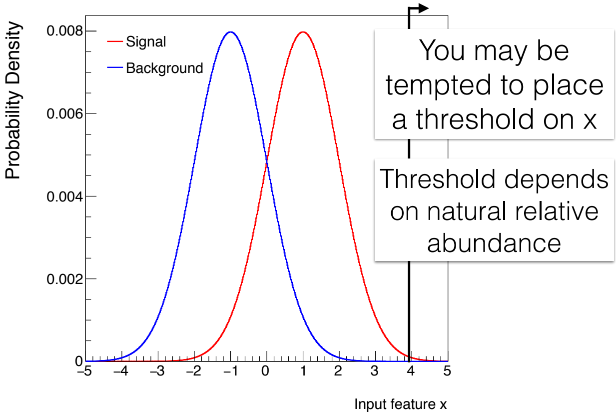
Let's consider an important special case:
binary classification in 1D



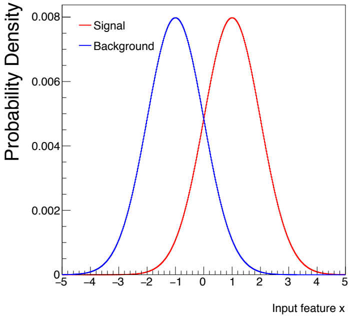

You may be tempted to place a threshold on x

# *Binary classification (V)*

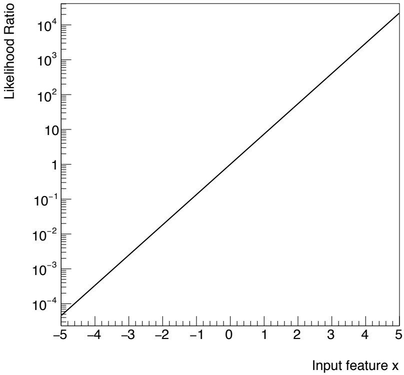
Let's consider an important special case:
binary classification in 1D



# *Binary classification (VI)*



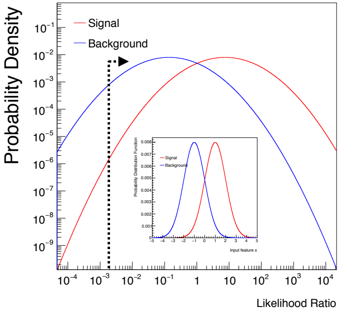

Is the simple threshold cut optimal?



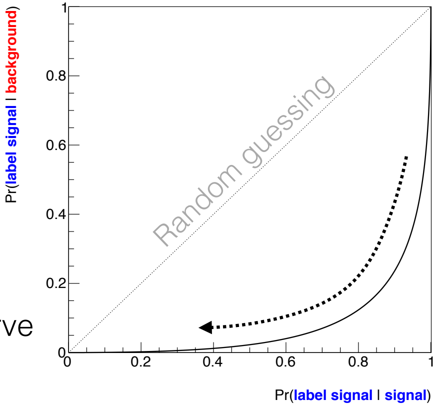

In this simple case, the log LL is proportional to x: **no need for non-linearities**!

*Threshold cut is optimal*

# *Binary classification (VII)*




The optimal procedure is a threshold on the LL

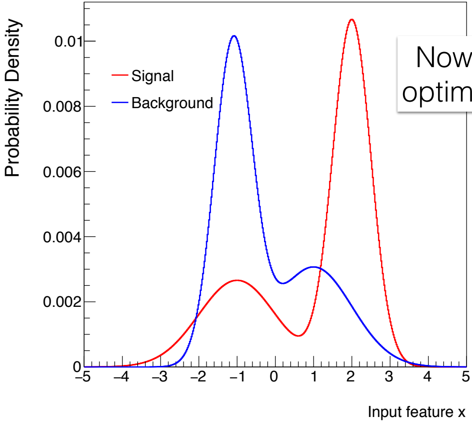



"Receiver Operating Characteristic" (**ROC**) Curve

# *"Realistic" classification (I)*

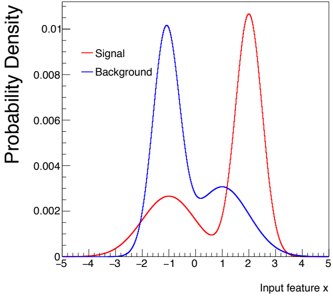
What if the distribution of x is complicated?

Real life is complicated!

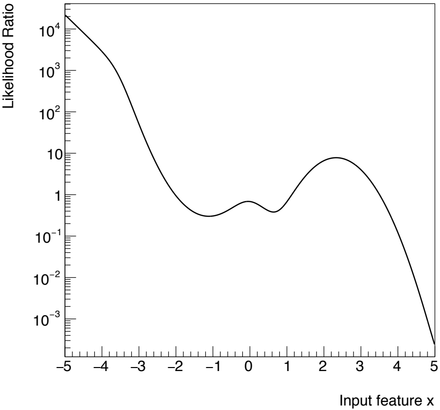



Now what is the optimal classifier?

# “Realistic” classification (II)

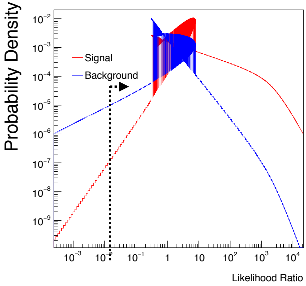


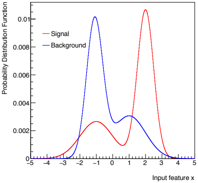

In this case, LL is highly non-linear (**non-monotonic**) function of x




A threshold on x would be sub-optimal

# “Realistic” classification (III)




ROC is worse than the Gaussians, but that is expected since the overlap in their PDFs is higher.