Artificial Neural Network

by Dane Brown

ML like the Human Brain

Perceptrons

- **Single Perceptron**
  - solves linear problems with a decision boundary function
  - Can attain similar accuracy as linear SVM if you fidget with parameters and get lucky
- **Multilayer Perceptron (MLP)**
  - hidden layers allow for a complex decision boundary
  - solves linear/non-linear problems

Perceptrons

Check it out -> CV_ML

Remember linear classifiers prefer sharper features like digits

Perceptrons

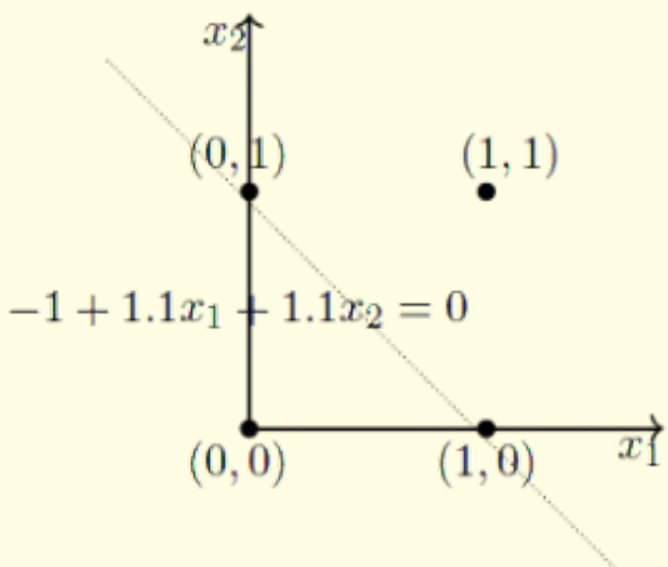
Can we plot an OR function ‘linearly’?

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$
$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 > -w_0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 > -w_0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 \geq 0 \implies w_1 + w_2 > -w_0$$

One possible solution is

$$w_0 = -1,\ w_1 = 1.1,\ w_2 = 1.1$$

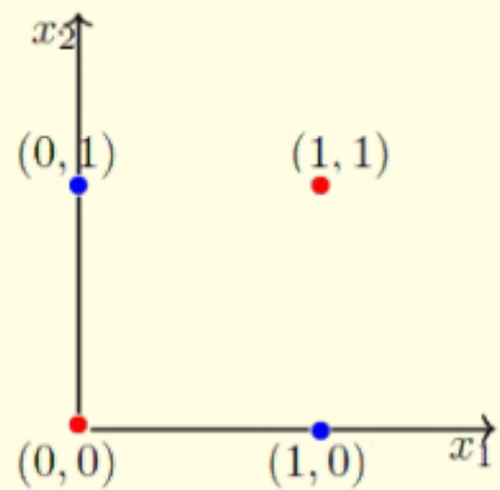


Perceptrons

- Can we plot an XOR function ‘linearly’?
- No, cannot separate red from blue points

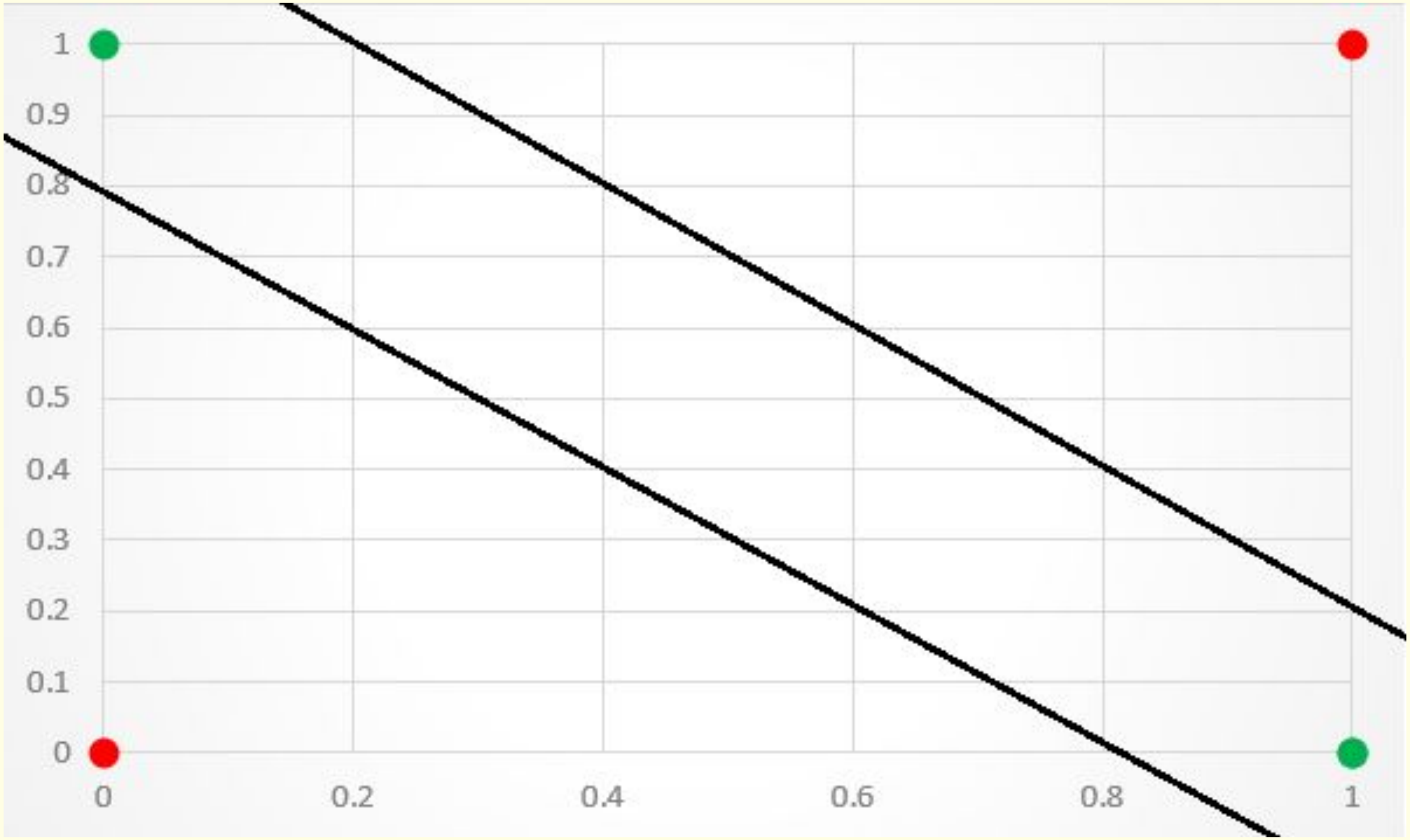
x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$
$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 > -w_0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 > -w_0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 \geq 0 \implies w_1 + w_2 < -w_0$$



Check it out -> CV_ML

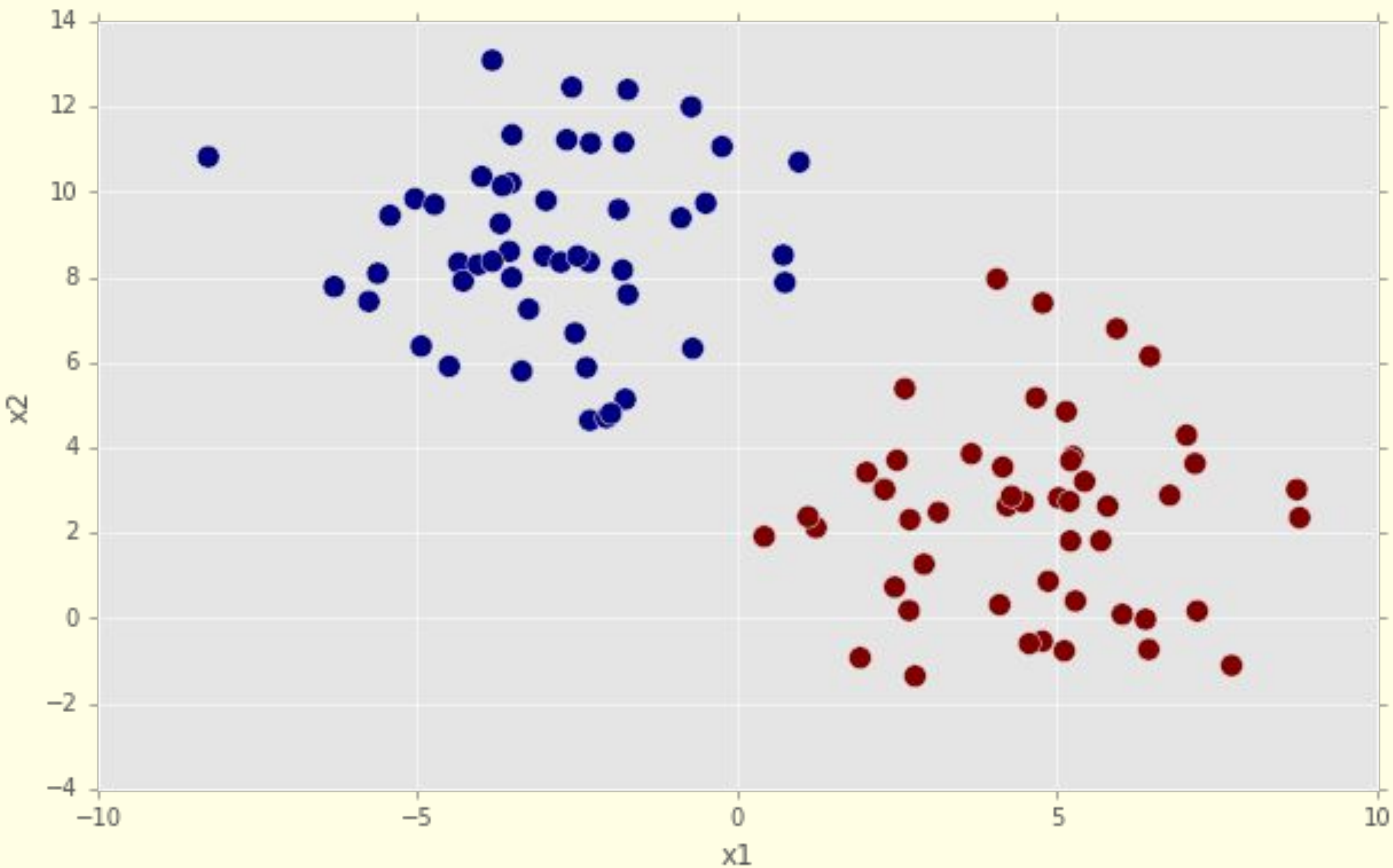
But two perceptrons can



Perceptrons

Check it out -> CV_ML

- Train a perceptron and show the resulting decision boundary for this easy blobs data (again)



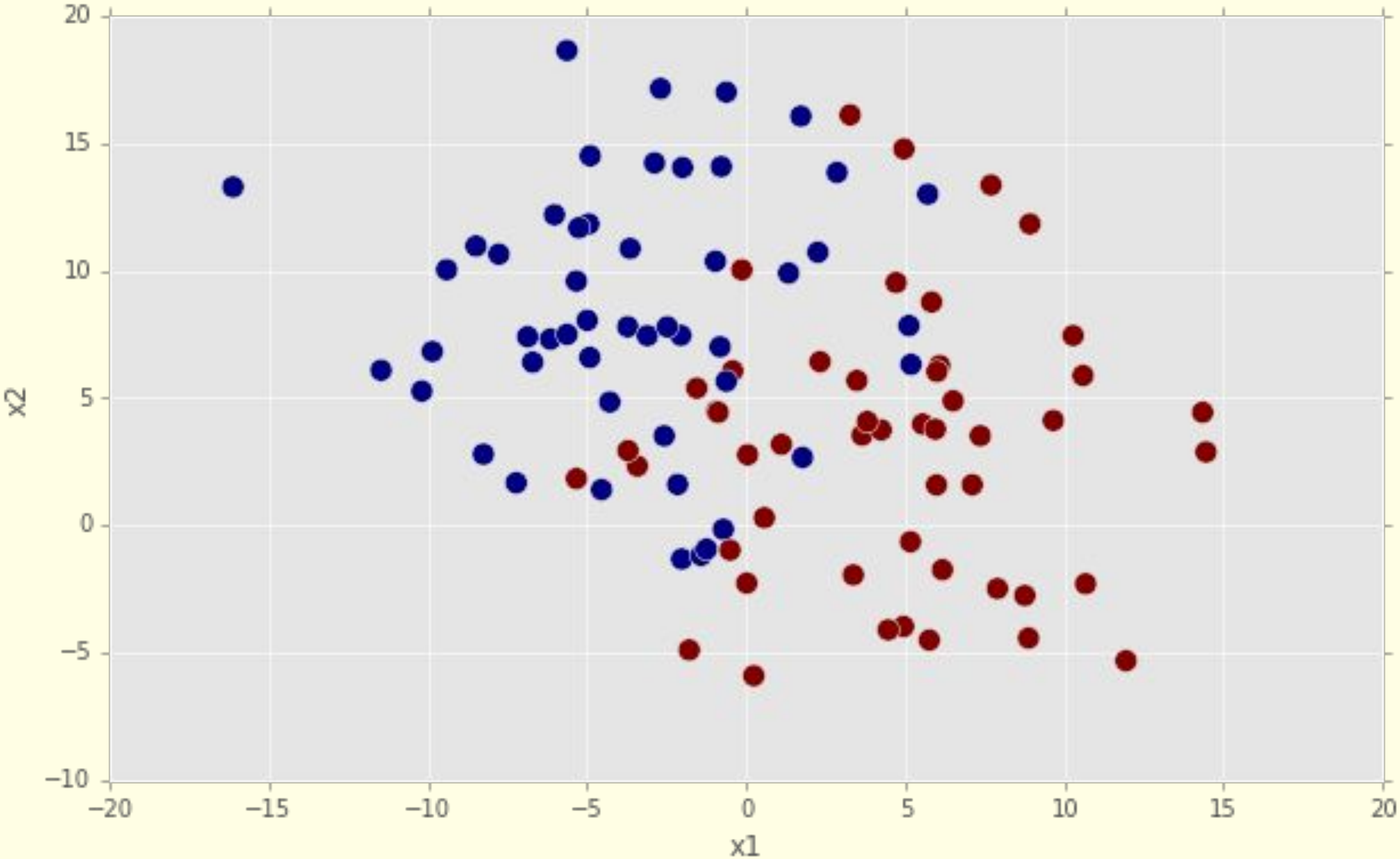
Perceptrons

- The perceptron learnt the decision boundary of the form: $\mathbf{12}x_1 - \mathbf{4.1}x_2 + \mathbf{3.0} >= \mathbf{0}$

Check it out -> CV_ML

Perceptrons

- Decision boundary difficult blobs data



MLPs

- How to make the perceptron more powerful and create nonlinear decision boundaries?
  - add layers
  - tune more

- MLPs combine multiple perceptrons to form a larger neural network that represents a more complex decision boundary. They have:
  - input layer
  - hidden layer(s)
  - output layer (as labels for classification)

MLP in OpenCV

Check it out -> CV_ML

- Let’s find that decision boundary in OpenCV!

MLP in OpenCV

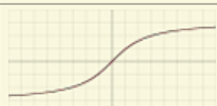
- You expected a large performance increase?
- Try adding more neurons to the hidden layer

Artificial Neural Network (ANN): What works best?

- **Activation functions** (AF) are important for ANNs
  - Allow complex and non-linear functional mappings between the inputs and response variable
  - Convert a input signal of a node in an ANN to an output signal
  - That output signal is used as input in the next layer in the stack

Artificial Neural Network (ANN)

AF:

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x \neq 0 \\ ? & \text{for} \quad x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

MLP vs. CNN

- Shallow MLP vs. Shallow CNN in Keras
- CNN specifically effective for image processing

MLP

- MLP with **one hidden** layer
- same number of neurons as there are inputs (**784**)
- **relu** activation is used for the hidden layer
- **Softmax** activation on output layer
- **Logarithmic** loss function
- **ADAM** gradient descent algorithm is fast and is used to learn the weights.
- **Fit** and **evaluate** the model over **10 epochs** with updates every 200 images as **batches**

`Check it out -> CV_ML`

Convolution Neural Network (CNN): State-of-the-Art for CV

- **Convolutional** layer with 32 5×5 feature maps and **relu** activation.
- **Pooling** layer 2x2
- Regularization layer using **Dropout**.
- Randomly exclude 20% neurons, avoid overfitting
- **Flatten** the layers as one fully connected layer
- Fully connected layer with 128 neurons and **relu**
- Ditto: Softmax, Log, ADAM, fit in batches

`Check it out -> CV_ML`