Queen Mary
University of London

DR ADRIAN BEVAN

PRACTICAL MACHINE LEARNING

INTRODUCTORY NEURAL NETWORKS (NNs)

LECTURE PLAN

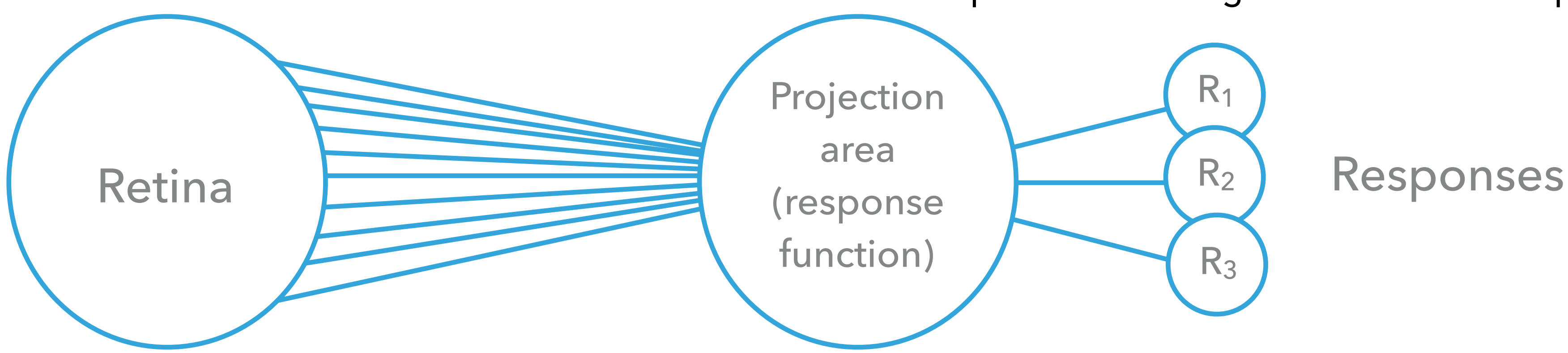
- Perceptrons
- Activation functions
- Artificial Neural Network
- Multilayer Perceptrons
- Training
- Summary

QMUL Summer School: https://www.qmul.ac.uk/summer-school/
Practical Machine Learning QMplus Page: https://qmplus.qmul.ac.uk/course/view.php?id=10006

PERCEPTRONS

- Rosenblatt[1] coined the concept of a perceptron as a probabilistic model for information storage and organisation in the brain.
  - Origins in trying to understand how information from the retina is processed.

Simplified view of Fig 1 from Rosenblatt's paper.



    - Start with inputs from different cells.
    - Process those data: "if the sum of excitatory or inhibitory impulse intensities is either equal to or greater than the threshold (θ) ... then the A unit fires".
  - This is an all or nothing response-based system.

[1] F. Rosenblatt, Psych. Rev. **65** p386-408, 1958.

PERCEPTRONS

- This picture can be generalised as follows:
  - Take some number, n, of input features
  - Compute the sum of each of the features multiplied by some factor assigned to it to indicate the importance of that information.
  - Compare the sum against some reference threshold.
  - Give a positive output above some threshold.

[1] F. Rosenblatt, Psych. Rev. **65** p386-408, 1958.

PERCEPTRONS

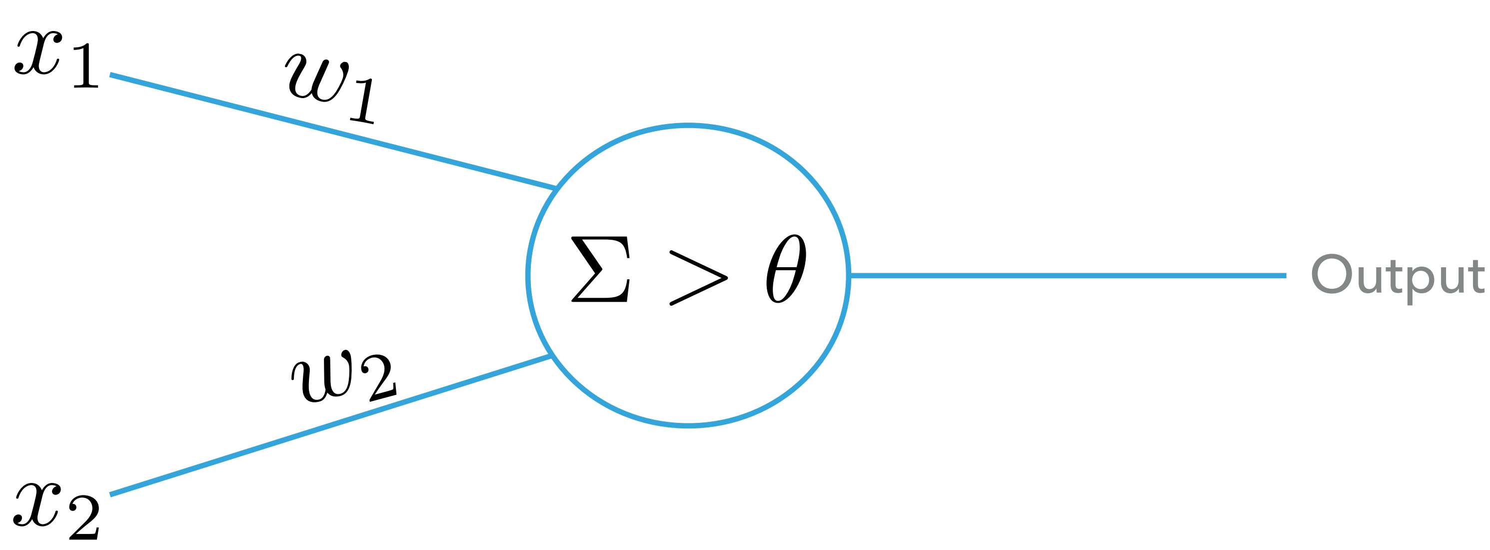
- Illustrative example:
  - Consider a measurement of two quantities x_1, and x_2.
  - Based on these measurements we determine if the perceptron is to give an output (value = 1) or not (value = 0).

$$w_1 x_1 + w_2 x_2 = \begin{cases} 0 \\ 1 \end{cases}$$

[1] F. Rosenblatt, Psych. Rev. **65** p386-408, 1958.

PERCEPTRONS

- Illustrative example:
  - Consider a measurement of two quantities x_1, and x_2.
  - Based on these measurements we determine if the perceptron is to give an output (value = 1) or not (value = 0).



[1] F. Rosenblatt, Psych. Rev. **65** p386-408, 1958.

PERCEPTRONS

- Illustrative example:
  - Consider a measurement of two quantities x_1, and x_2.
  - Based on these measurements we determine if the perceptron is to give an output (value = 1) or not (value = 0).

If $w_1 x_1 + w_2 x_2 > \theta$

Output = 1

else

Output = 0

[1] F. Rosenblatt, Psych. Rev. **65** p386-408, 1958.

PERCEPTRONS

- Illustrative example:
  - Consider a measurement of two quantities x_1, and x_2.
  - Based on these measurements we determine if the perceptron is to give an output (value = 1) or not (value = 0).

If $w_1 x_1 + w_2 x_2 > \theta$

Output = 1

else

Output = 0

This is called a binary activation function

[1] F. Rosenblatt, Psych. Rev. **65** p386-408, 1958.

PERCEPTRONS

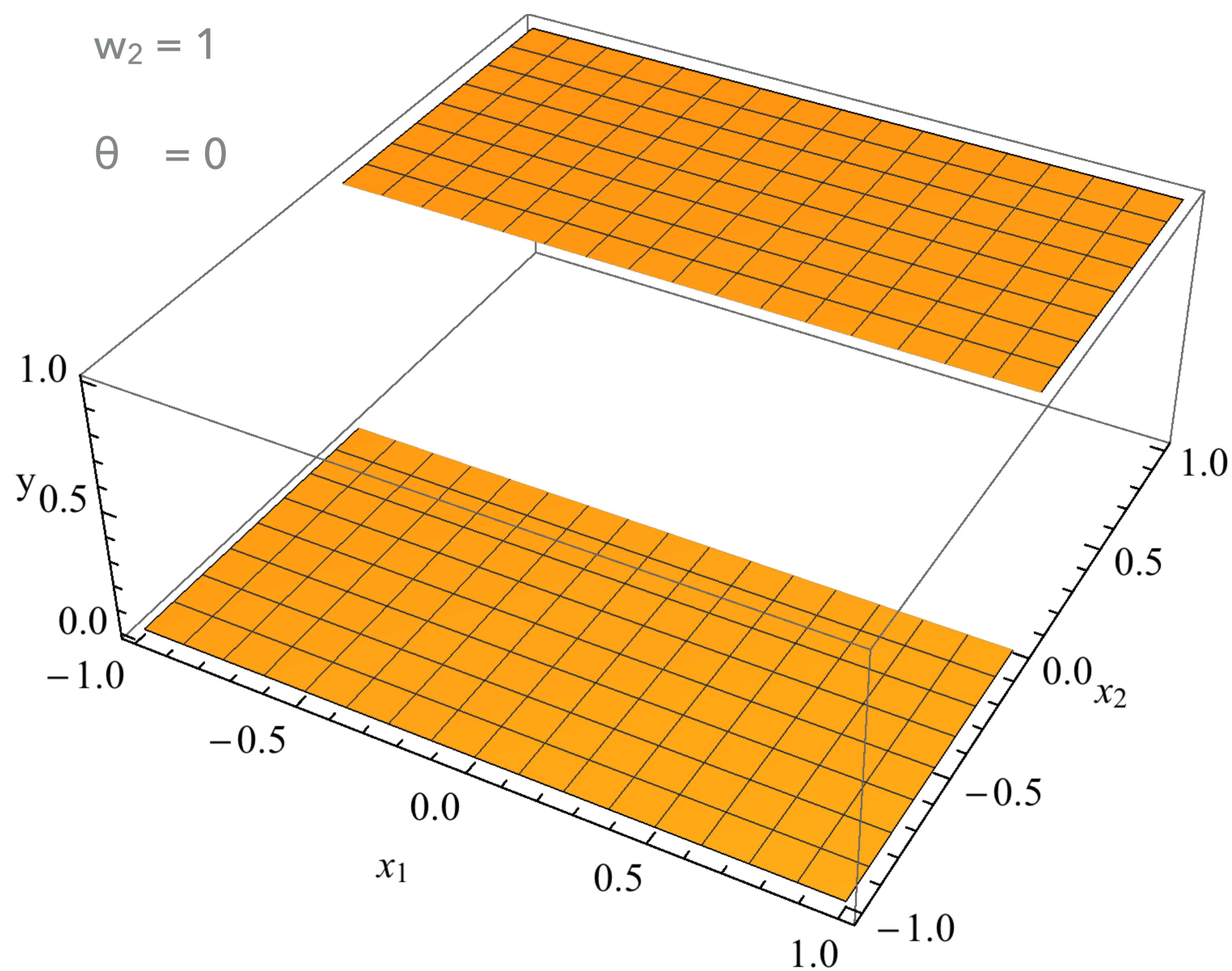
- Illustrative example:

$w_1 = 0$

$w_2 = 1$

$\theta \;\; = 0$

- Decision is made on x_2
- Output value is either 1 or 0 as some $f(x_1, x_2)$ that depends on the values of w_1, w_2 and θ.



[1] F. Rosenblatt, Psych. Rev. **65** p386-408, 1958.

PERCEPTRONS

In particle physics we often use machine learning to suppress background. Here y=1 corresponds to signal and y=0 corresponds to background.

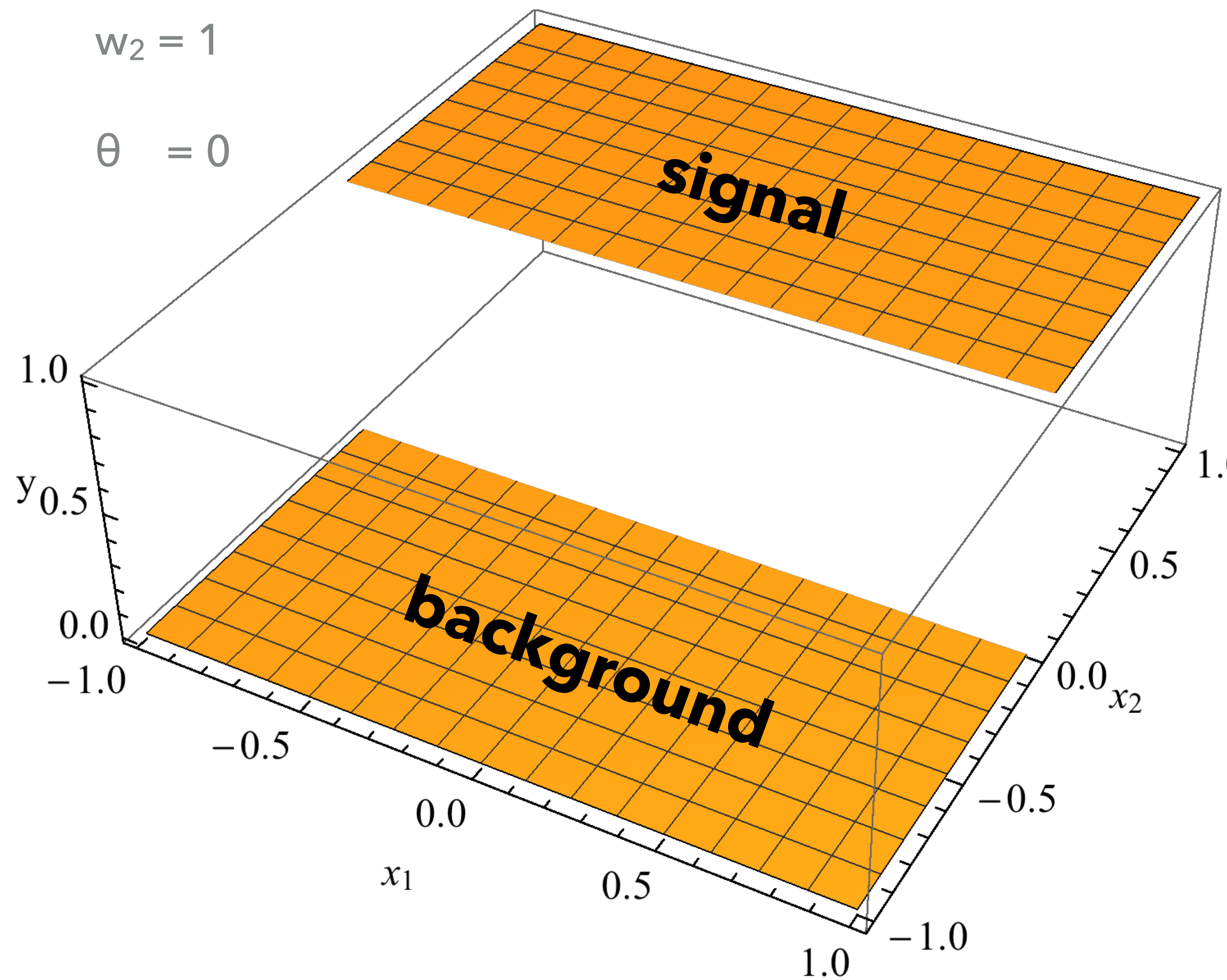
- Illustrative example:

$w_1 = 0$

$w_2 = 1$

$\theta = 0$

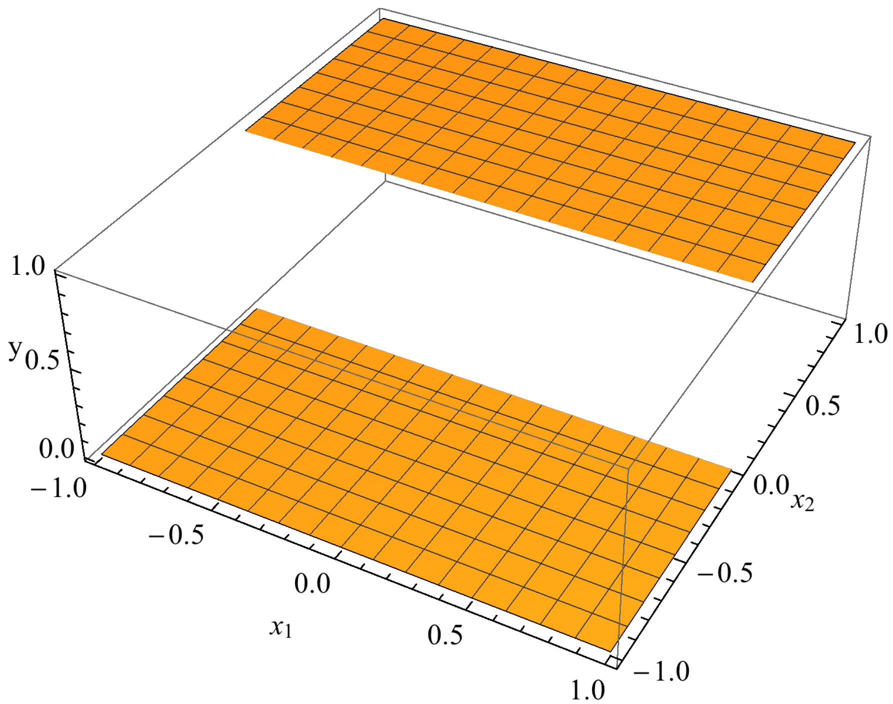
- Decision is made on x_2
- Output value is either 1 or 0 as some $f(x_1, x_2)$ that depends on the values of w_1, w_2 and θ.



[1] F. Rosenblatt, Psych. Rev. **65** p386-408, 1958.

PERCEPTRONS

- Illustrative examples:

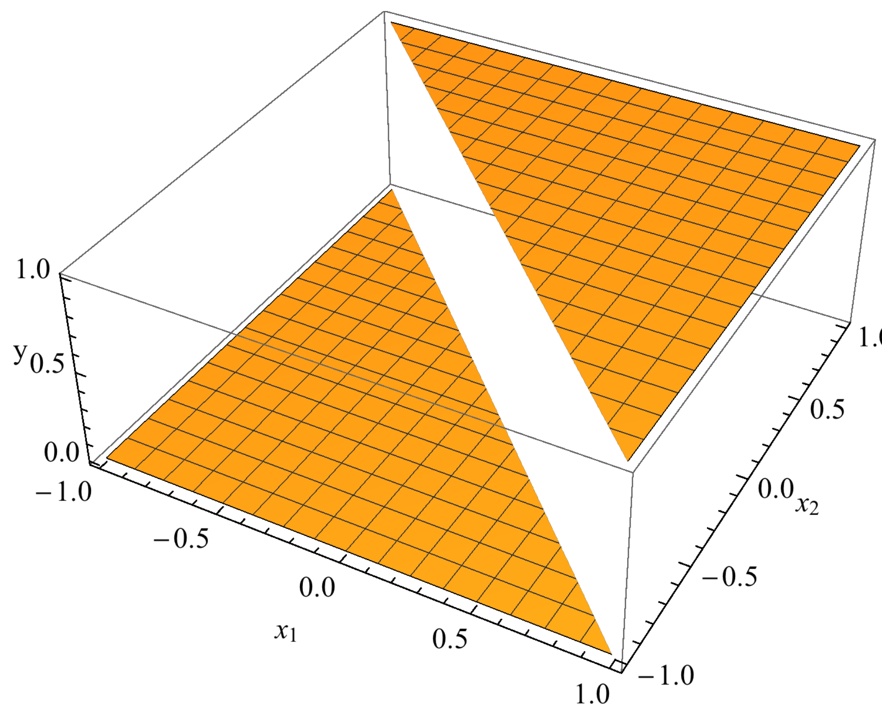


$w_1 = 0$

$w_2 = 1$

$\theta = 0$

Baseline for comparison, decision only on value of x_2

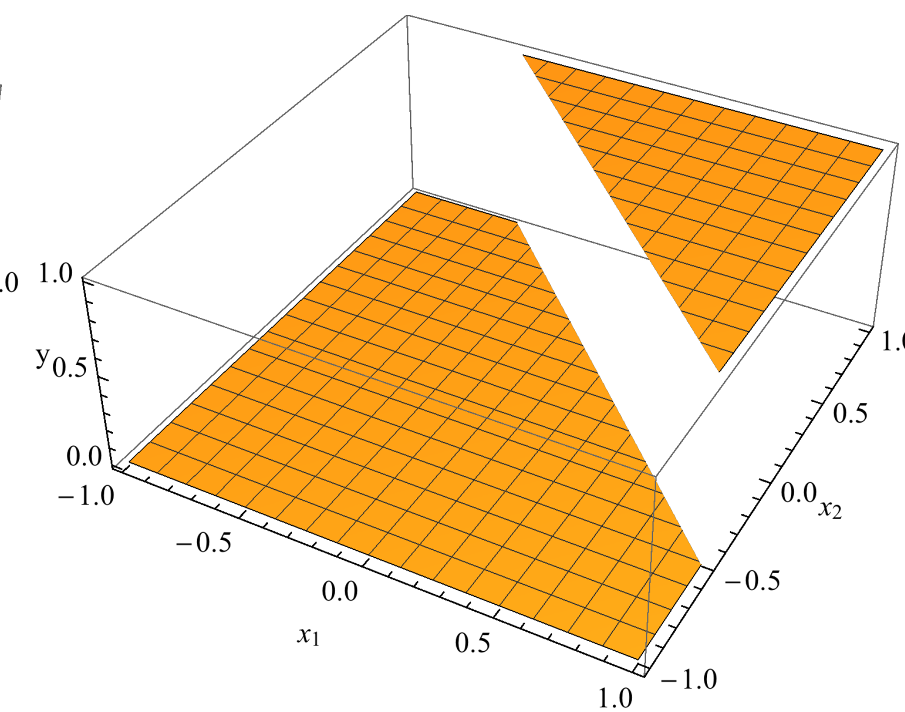


$w_1 = 1$

$w_2 = 1$

$\theta = 0$

Rotate decision plane in (x_1, x_2)



$w_1 = 1$

$w_2 = 1$

$\theta = 0.5$

Shift decision plane away from origin

[1] F. Rosenblatt, Psych. Rev. **65** p386-408, 1958.

PERCEPTRONS

- Illustrative example:
  - Consider a measurement of two quantities x_1, and x_2.
  - Based on these measurements we determine if the perceptron is to give an output (value = 1) or not (value = 0).
- We can generalise the problem to N quantities as

$$
\begin{aligned}
y &= f\left(\sum_{i=1}^{N} w_i x_i + \theta\right) \\
&= f(w^T x + \theta)
\end{aligned}
$$

[1] F. Rosenblatt, Psych. Rev. **65** p386-408, 1958.

PERCEPTRONS

- Illustrative example:
  - Consider a measurement of two quantities x_1, and x_2.
  - Based on these measurements we determine if the perceptron is to give an output (value = 1) or not (value = 0).
- We can generalise the problem to N quantities as

$$y = f\left(\sum_{i=1}^{N} w_i x_i + \theta\right)$$
$$= f(w^T x + \theta)$$

The argument is just the same functional form of Fisher's discriminant.

[1] F. Rosenblatt, Psych. Rev. **65** p386-408, 1958.

PERCEPTRONS

- The problem of determining the weights remains (we will discuss optimisation later on).
- For now assume that we can use some heuristic to choose weights that are deemed to be "optimal" for the task of providing a response given some input data example.

[1] F. Rosenblatt, Psych. Rev. **65** p386-408, 1958.

ACTIVATION FUNCTIONS

- The binary activation function of Rosenblatt is just one type of activation function.
  - This gives an all or nothing response.
- It can be useful to provide an output that is continuous between these two extremes.
  - For that we require additional forms of activation function.

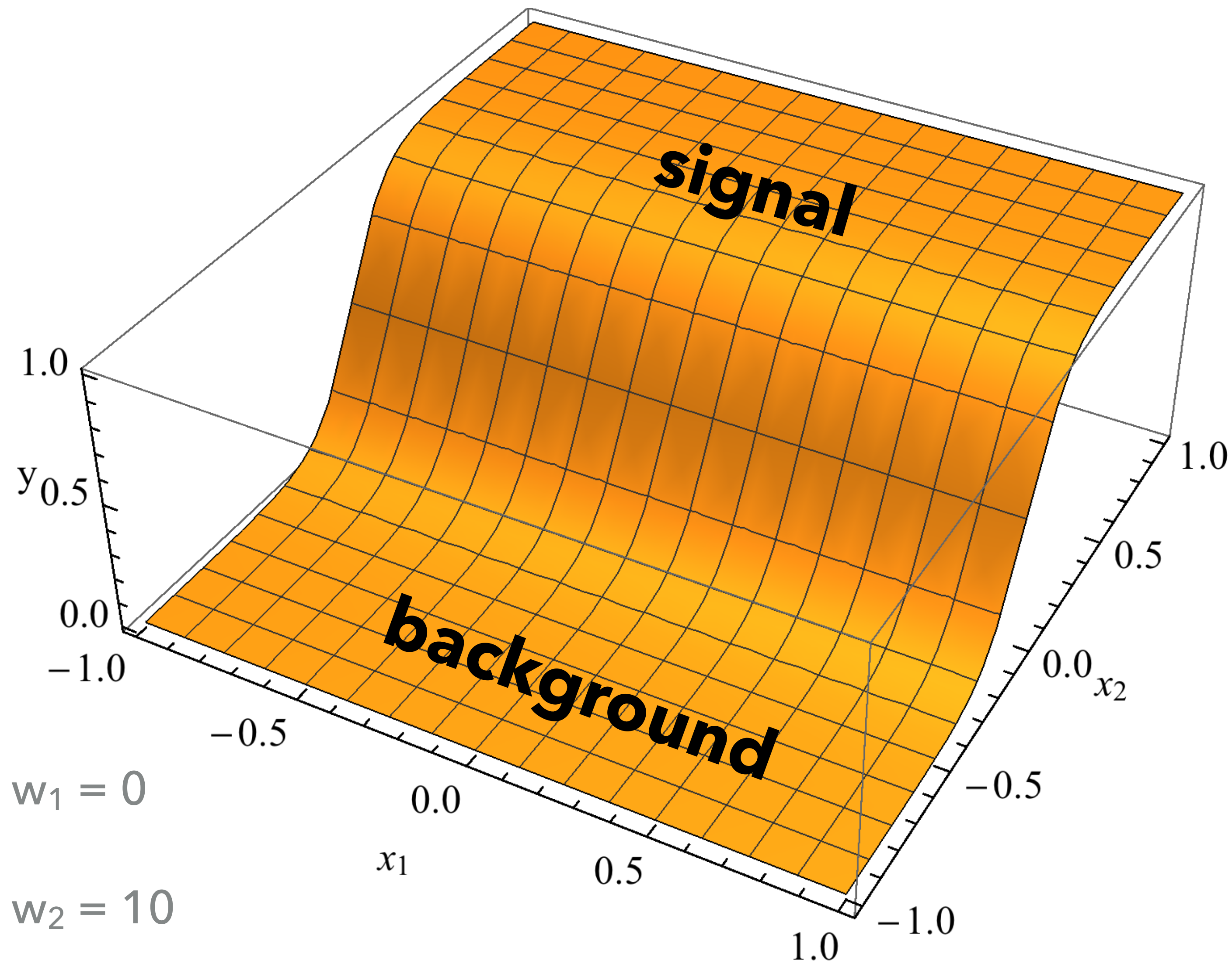
ACTIVATION FUNCTIONS

- TensorFlow has the following activation functions (`tf.nn.`**`ACTIVATIONFUNCTION`**)
  - relu (covered here)
  - leaky_relu (covered here)
  - relu6
  - crelu
  - elu
  - selu
  - softplus
  - softsign
  - dropout
  - bias_add
  - sigmoid (covered here)
  - tanh (covered here)

ACTIVATION FUNCTIONS: LOGISTIC (OR SIGMOID)

- A common activation function used in neural networks:

$$y = \frac{1}{1+e^{w^T x+\theta}} = \frac{1}{1+e^{(w_1 x_1 + w_2 x_2 + \theta)}}$$



$w_1 = 0$

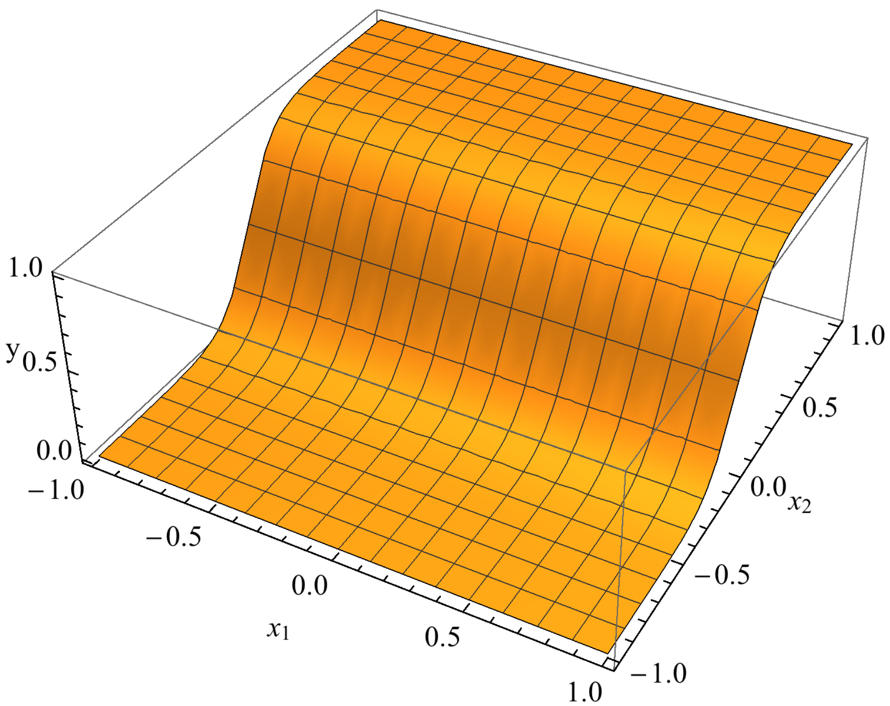
$w_2 = 10$

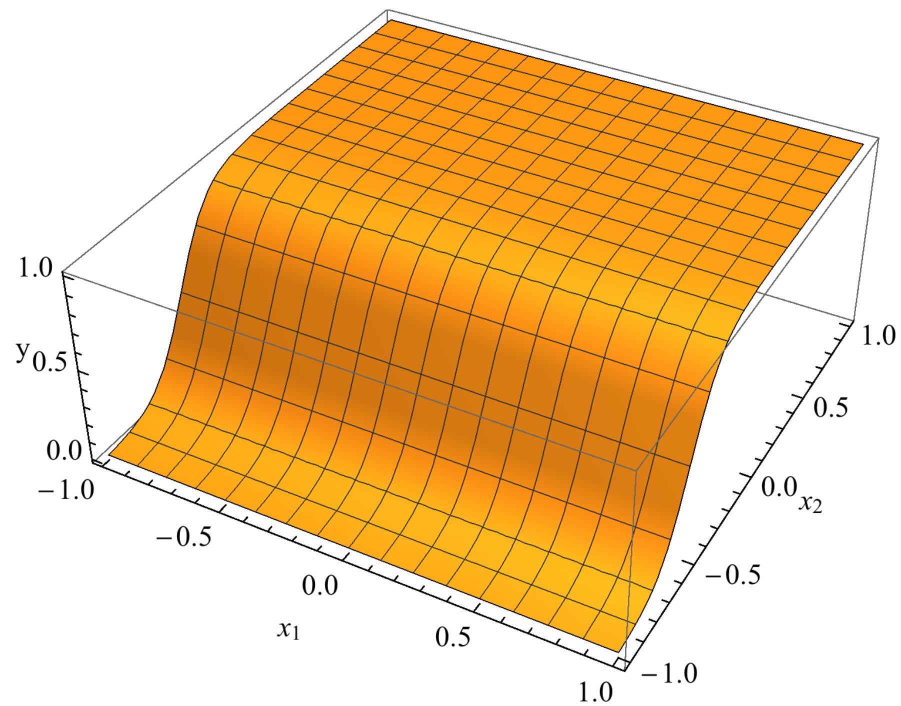
$\theta = 0$

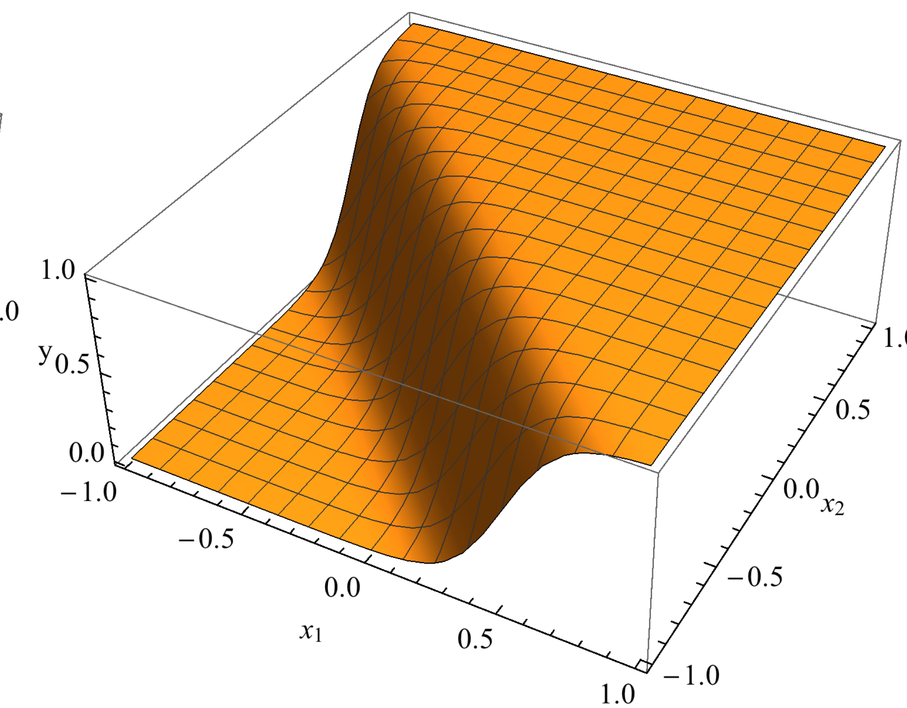
ACTIVATION FUNCTIONS: LOGISTIC (OR SIGMOID)

$$\frac{1}{1+e^{(w_1x_1+w_2x_2+\theta)}}$$

- A common activation function used in neural networks:







$w_1 = 0$ | $w_1 = 0$ | $w_1 = 10$

$w_2 = 10$ | $w_2 = 10$ | $w_2 = 10$

$\theta = 0$ | $\theta = -5$ | $\theta = -5$

Baseline for comparison, decision only on value of x_2

Offset from zero using θ

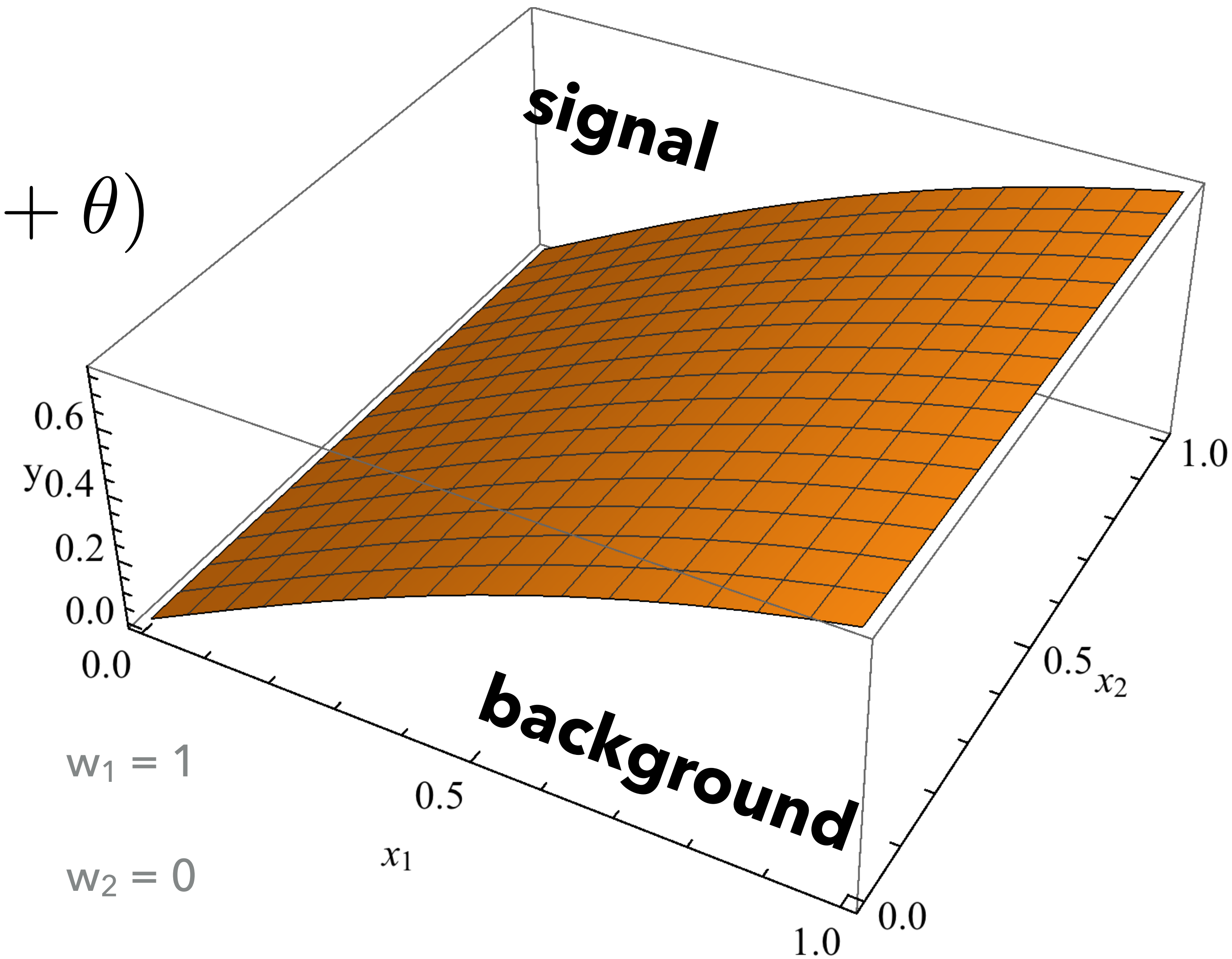
rotate "decision boundary" in (x_1, x_2)

ACTIVATION FUNCTIONS: HYPERBOLIC TANGENT

- A common activation function used in neural networks:

$$y = \tanh(w^T x + \theta)$$
$$= \tanh(w_1 x_1 + w_2 x_2 + \theta)$$

(Often used with θ = 0)



$w_1 = 1$

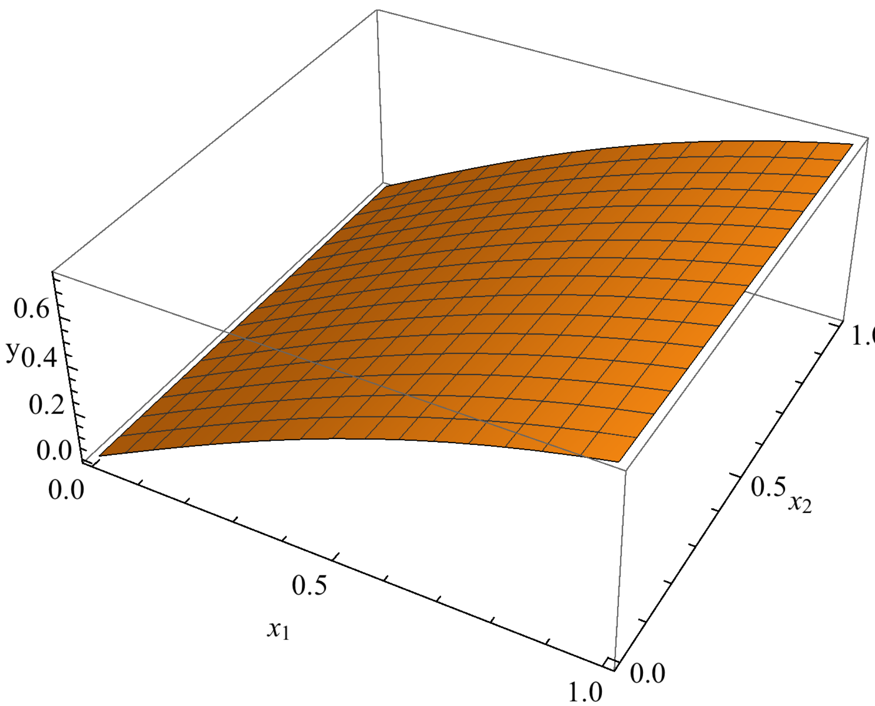
$w_2 = 0$

$\theta = 0$

ACTIVATION FUNCTIONS: HYPERBOLIC TANGENT

$$\tanh(w_1 x_1 + w_2 x_2 + \theta)$$

- A common activation function used in neural networks:

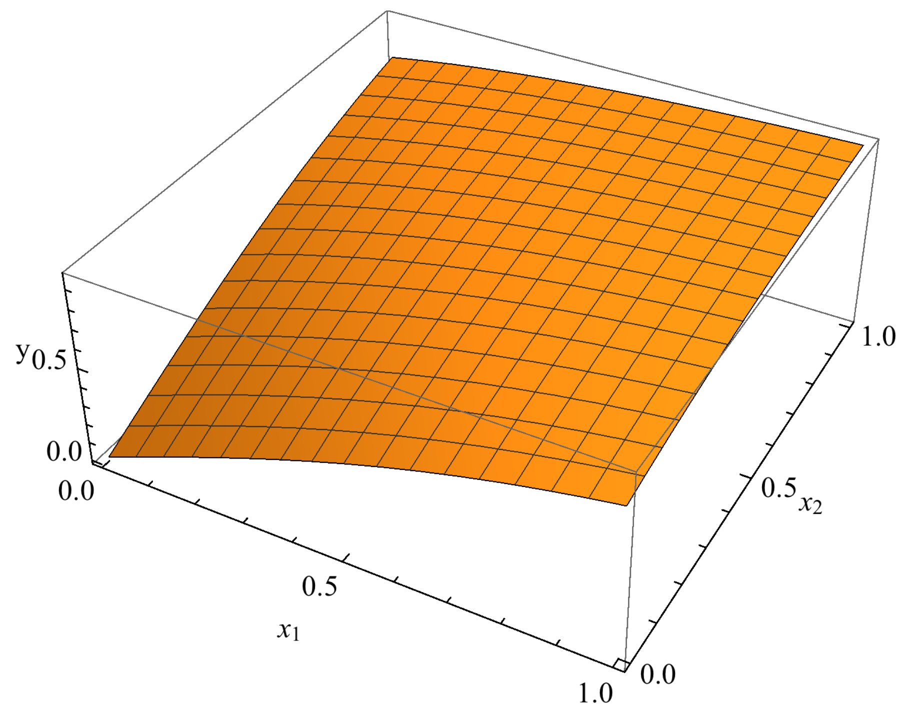


$w_1 = 1$

$w_2 = 0$

$\theta = 0$

Baseline for comparison, decision only on value of x_1

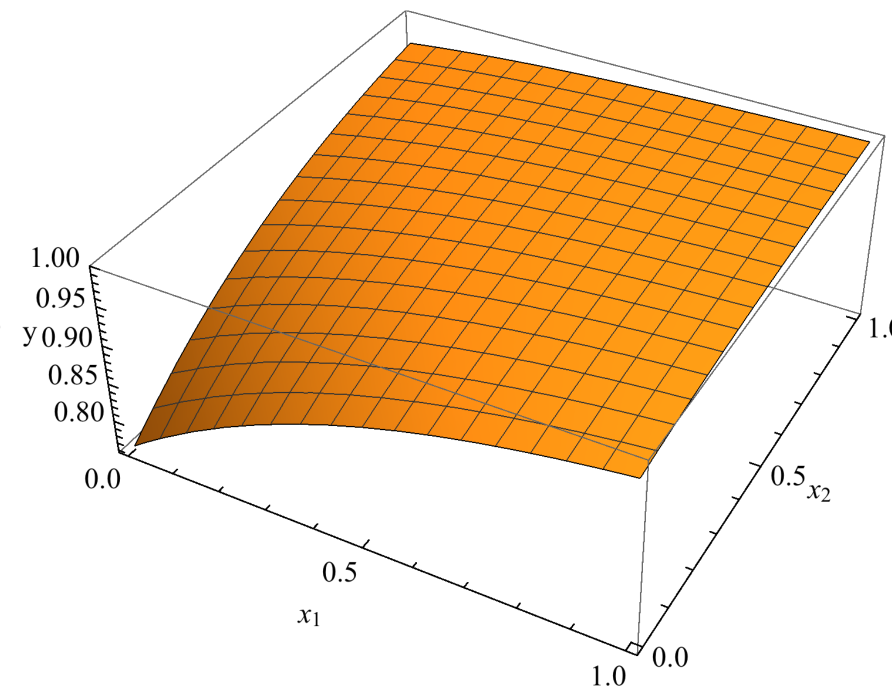


$w_1 = 1$

$w_2 = 1$

$\theta = 0$

rotate "decision boundary" in (x_1, x_2)



$w_1 = 1$

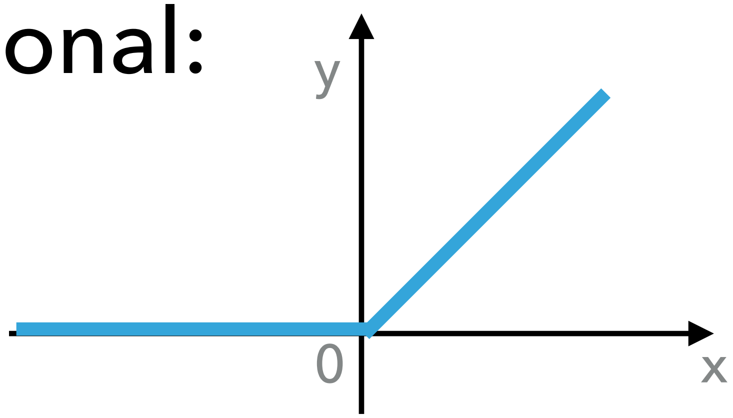
$w_2 = 1$

$\theta = -1$

Offset (vertically) from zero using θ

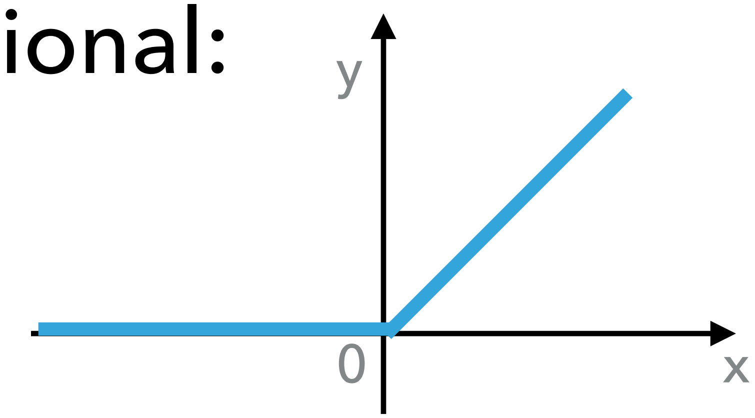
ACTIVATION FUNCTIONS: RELU

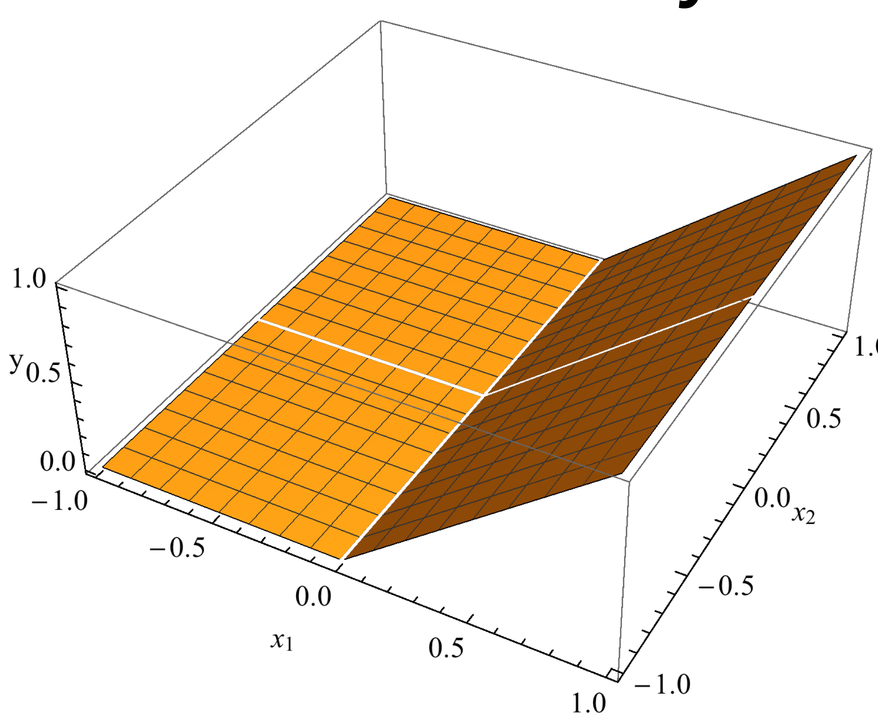
- The **Re**ctified **L**inear **U**nit activation function is commonly used for CNNs. This is given by a conditional:
  - If (x < 0) y = 0
  - otherwise y = x



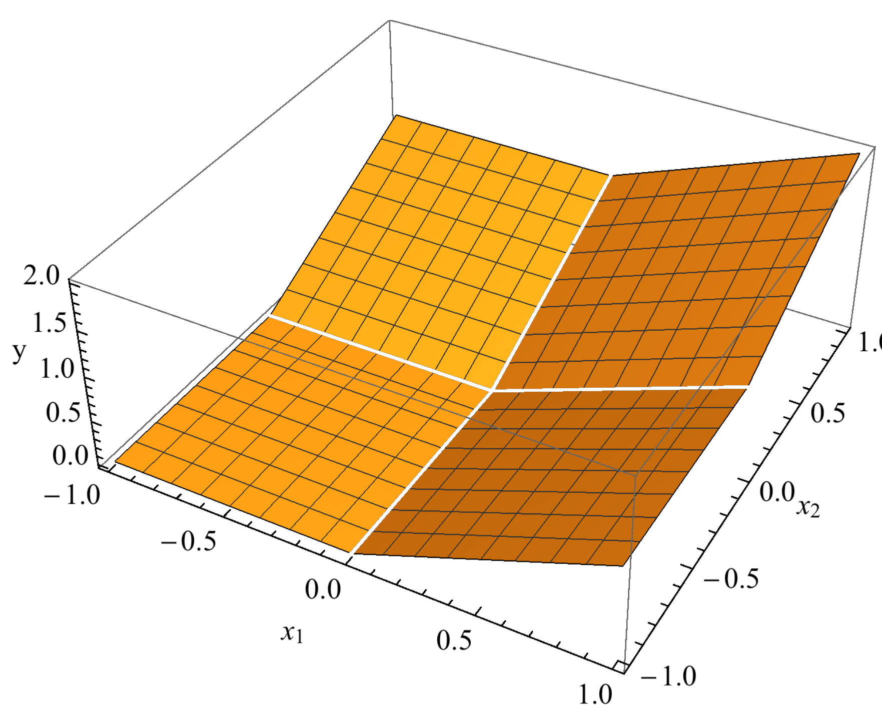
ACTIVATION FUNCTIONS: RELU

- The **Re**ctified **L**inear **U**nit activation function is commonly used for CNNs. This is given by a conditional:
  - If (x < 0) y = 0
  - otherwise y = x

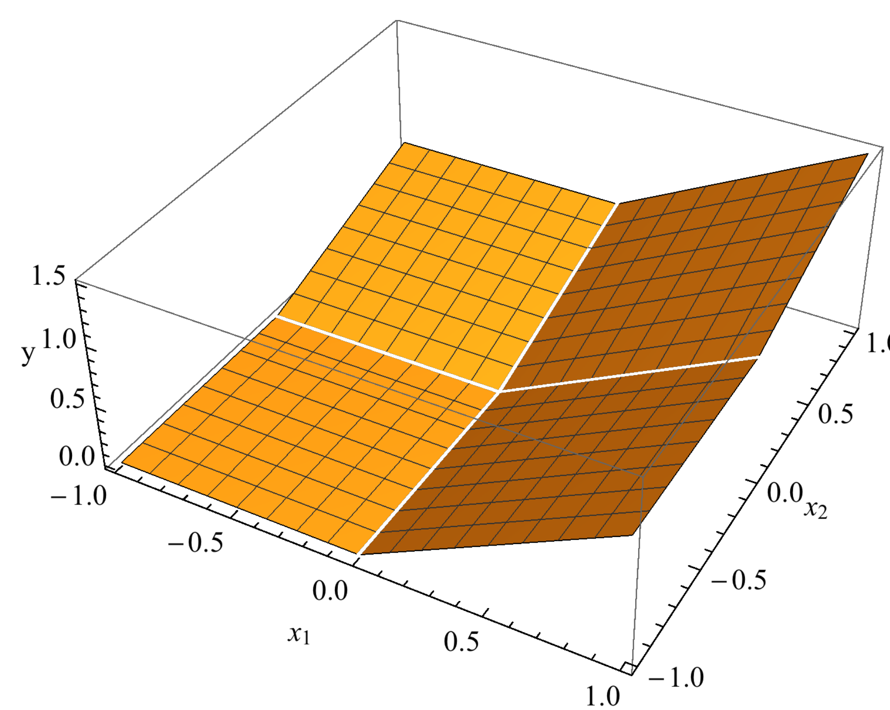




$w_1 = 1, w_2 = 0$



$w_1 = 1$, $w_2 = 1$

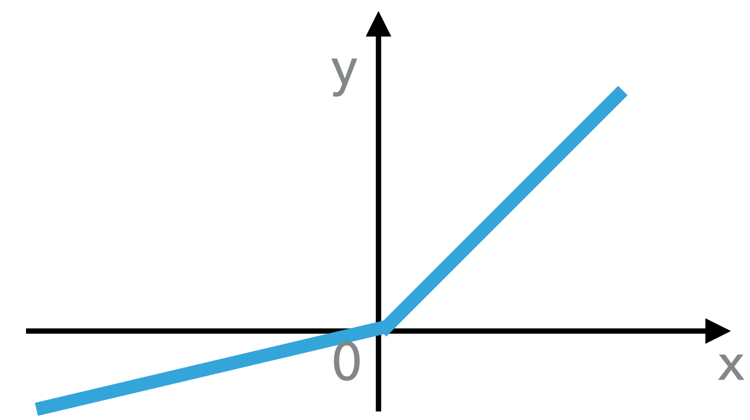


$w_1 = 1$, $w_2 = 0.5$

Importance of features in the perceptron still depend on weights as illustrated in these plots.

ACTIVATION FUNCTIONS: PRELU VARIANT

- The ReLU activation function can be modified to avoid gradient singularities.
- This is the **PReLU** or **Leaky ReLU** activation function
  - If (x < 0) y = a*x
  - otherwise y = x



- Collectively we can write the (P)ReLU activation function as

$$f(x) = \max(0, x) + a \min(0, x)$$

- Can be used effectively for need CNNs (more than 8 convolution layers), whereas the ReLU activation function can have convergence issues for such a configuration[2].
- If a is small (0.01) it is referred to as a leaky ReLU function[1]. The default implementation in TensorFlow has a=0.2[3].

[1] Maas, Hannun, Ng, ICML2013.
[2] He, Zhang, Ren and Sun, arXiv:1502.01852
[3] See https://github.com/tensorflow/tensorflow/blob/r1.8/tensorflow/python/ops/nn_ops.py

ACTIVATION FUNCTIONS: RELU

- Performs better than a sigmoid for a number of applications[1].
  - Weights for a relu are typically initialised with a truncated normal, OK for shallow CNNs, but there are convergence issues with deep CNNs when using this initialisation approach[1].

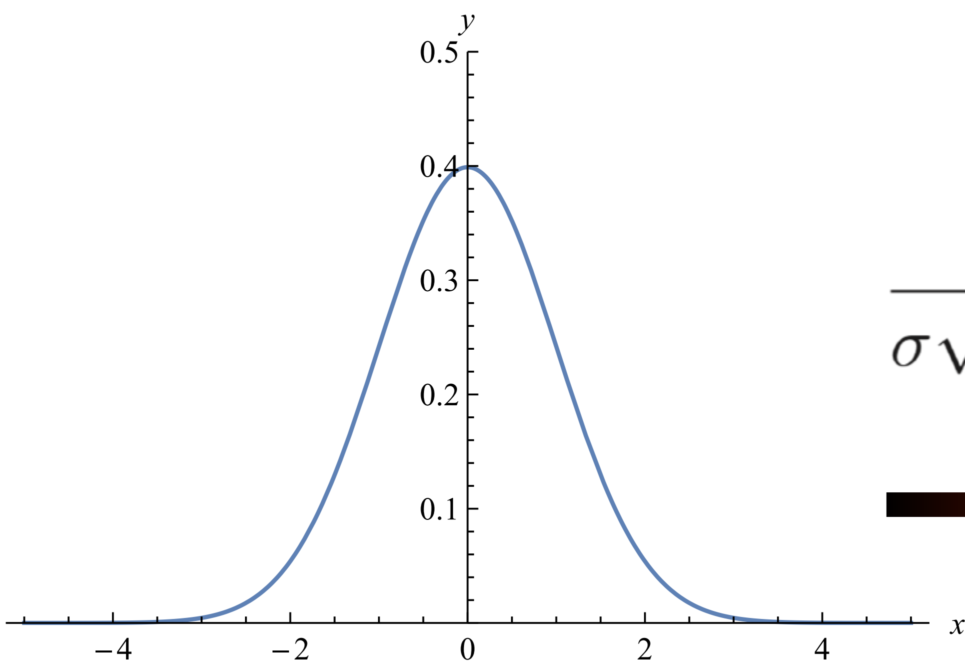
```
initial = tf.truncated_normal(shape, stddev=0.1)
```

  - Other initialisation schemes have been proposed to avoid this issue for deep CNNs (more than 8 conv layers) as discussed in Ref [2].

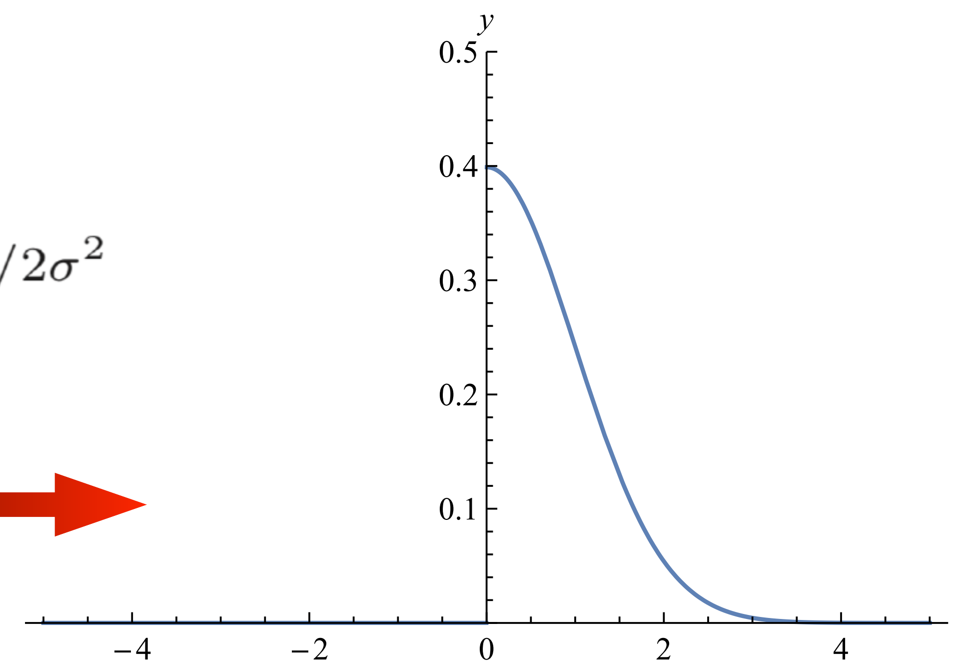
[1] Maas, Hannun, Ng, ICML2013.
[2] He, Zhang, Ren and Sun, arXiv:1502.01852

ACTIVATION FUNCTIONS: RELU

- N.B. Gradient descent optimisation algorithms will not change the weights for an activation function if the initial weight is set to zero.
  - This is why a truncated normal is used for initialisation, rather than a Gaussian that has x∈[-∞, ∞].



$$\frac{1}{\sigma\sqrt{2\pi}}e^{-(x-\mu)^2/2\sigma^2}$$

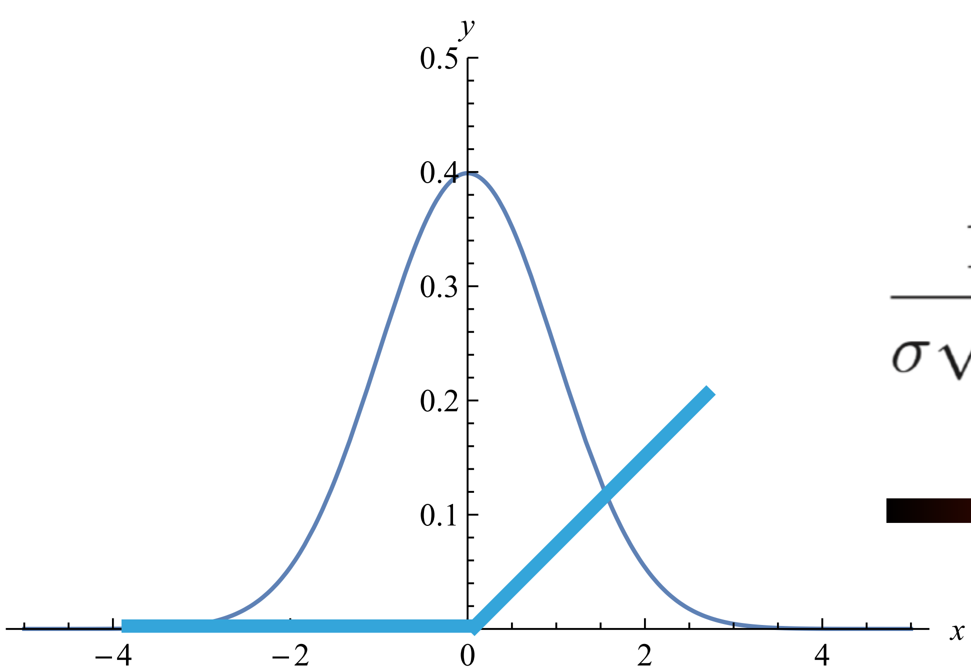


[1] Maas, Hannun, Ng, ICML2013.
[2] He, Zhang, Ren and Sun, arXiv:1502.01852

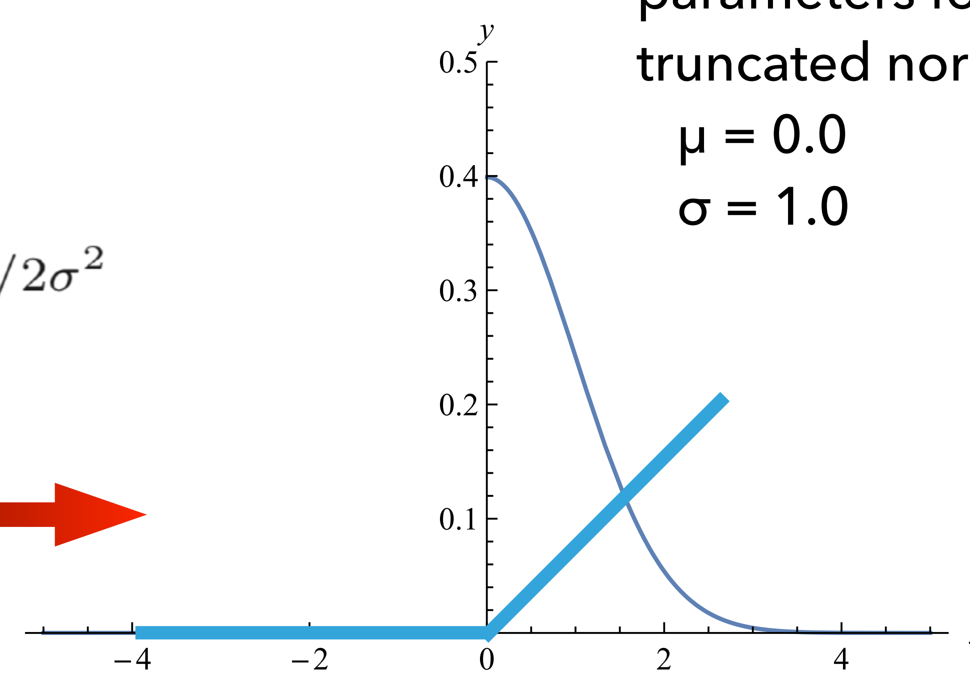
ACTIVATION FUNCTIONS: RELU

- N.B. Gradient descent optimisation algorithms will not change the weights for an activation function if the initial weight is set to zero.
  - This is why a truncated normal is used for initialisation, rather than a Gaussian that has x∈[-∞, ∞].

TensorFlow default parameters for the truncated normal are:
μ = 0.0
σ = 1.0



$$\frac{1}{\sigma\sqrt{2\pi}}e^{-(x-\mu)^2/2\sigma^2}$$



[1] Maas, Hannun, Ng, ICML2013.
[2] He, Zhang, Ren and Sun, arXiv:1502.01852

ACTIVATION FUNCTIONS: DATA PREPARATION

- Input features are arbitrary; whereas activation functions have a standardised input domain of [-1, 1] or [0, 1].
  - Limits the range with which we have to adjust hyper-parameters to find an optimal solution.
  - Avoids large or small hyper-parameters.
  - Other algorithms have more stringent requirements for data-preprocessing when being fed into them.
    - All these points indicate that we need to prepare data appropriately before feeding it into a perceptron, and hence network.

ACTIVATION FUNCTIONS: DATA PREPARATION

- Input features are arbitrary; whereas activation functions have a standardised input domain of [-1, 1] or [0, 1].
  - We can map our input feature space onto a standardised domain that matches some range that matches that of the activation function.
  - Saves work for the optimiser in determining hyper-parameters.
  - Standardises weights to avoid numerical inaccuracies; and set common starting weights.
  - e.g.
    - having an energy or momentum measured in units of 10^{12} eV, would require weights $O(10^{-12})$ to obtain an O(1) result for $w_i x_i$.
    - Mapping eV $\longmapsto$ TeV would translate 10^{12} eV $\longmapsto$ 1TeV, and allow for O(1) weights leading to an O(1) result for $w_i x_i$.
    - Comparing weights for features that are standardised allows the user to develop an intuition as to what the corresponding activation function will look like.

ACTIVATION FUNCTIONS: DATA PREPARATION

- A good paper to read on data preparation is [1]. This includes the following suggestions:
  - Standardising input features onto [-1, 1] results in faster optimisation using gradient descent algorithms.
  - Shift the features to have a mean value of zero.
  - It is also possible to speed up optimisation by de-correlating input variables[1].
  - Having done this one can also scale the features to have a similar variance.

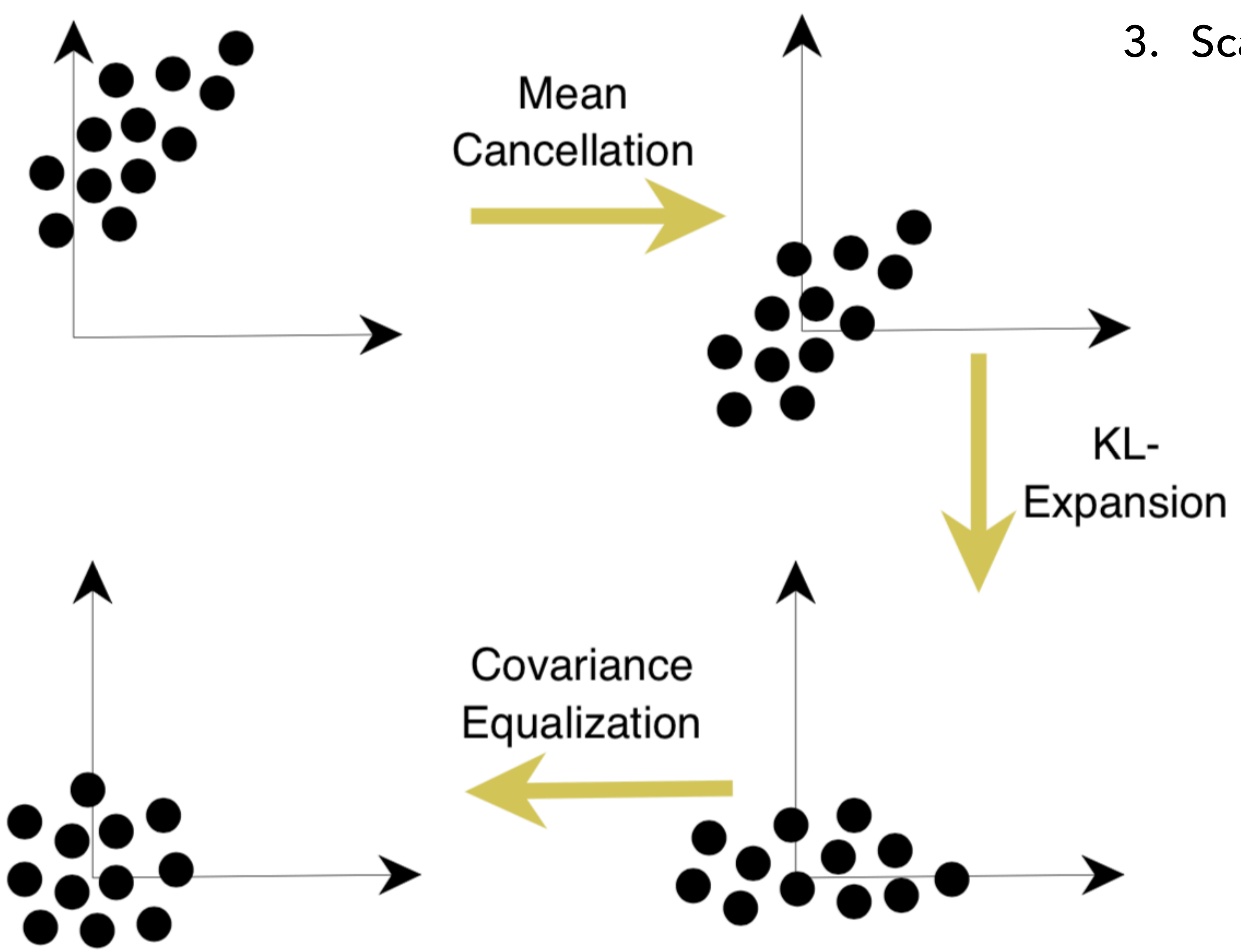
[1]Decorrelation of features is not essential assuming a sufficiently general optimisation algorithm is being used. The rationale is that in general if one can decorrelate features then we just have to minimise the cost as a function of weights for one feature at a time, rather than being concerned about the dependence of weights on more than one feature. So this is a choice made to simplify the minimisation process, and in to speed up that process.

[1] LeCun et al., Efficient BackProp, Neural Networks Tricks of the Trade, Springer 1998

ACTIVATION FUNCTIONS: DATA PREPARATION

- e.g.

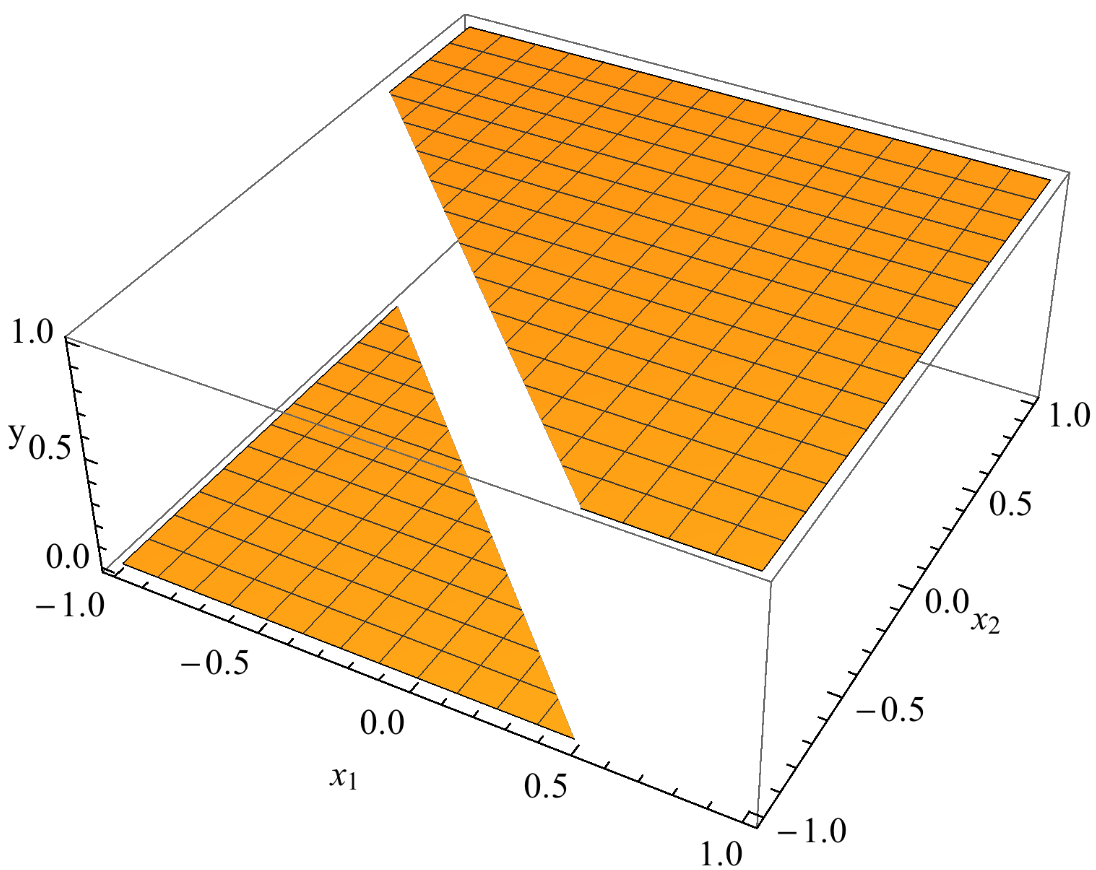
1. Shift the distribution to have a zero mean
2. Decorrelate input features
3. Scale to match covariance of features.



[1] LeCun et al., Efficient BackProp, Neural Networks Tricks of the Trade, Springer 1998 (Fig. 3)

ARTIFICIAL NEURAL NETWORKS (ANNs)

- A single perceptron can be thought of as defining a hyperplane that separates the input feature space into two regions.



A binary threshold activation function is an equivalent algorithm to cutting on a fisher discriminant to distinguish between types of training example.

$$\mathcal{F} = w^T x + \beta$$

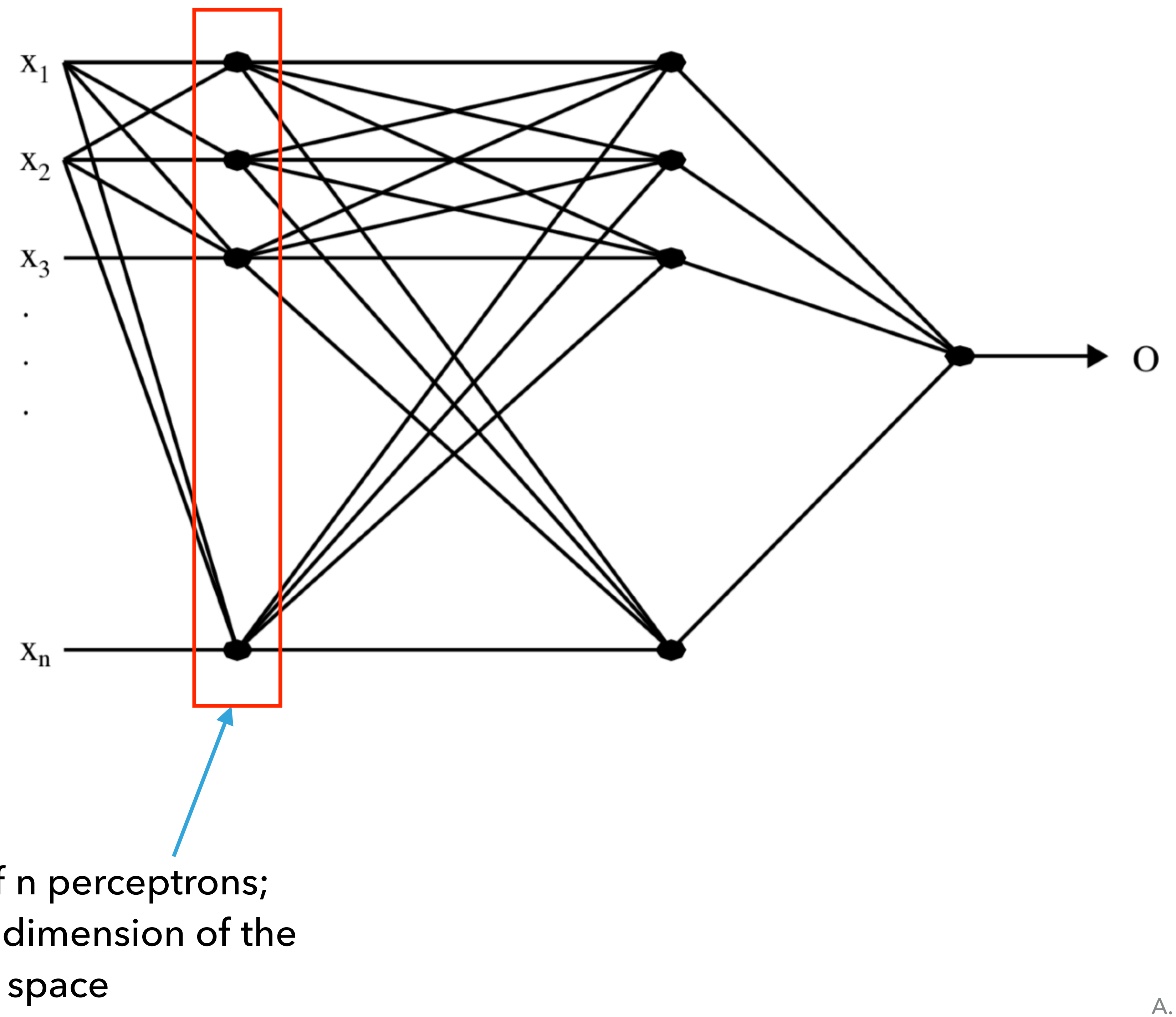
The only real difference is the heuristic used to determine the weights.

ARTIFICIAL NEURAL NETWORKS (ANNS)

- A single perceptron can be thought of as defining a hyperplane that separates the input feature space into two regions.
  - This is a literal illustration for the binary threshold perceptron.
  - The other perceptrons discussed have a gradual transition from one region to the other.
- We can combine perceptrons to impose multiple hyperplanes on the input feature space to divide the data into different regions.
- Such a system is an artificial neural network. There are various forms of ANNs; in HEP this is usually synonymous with a multi-layer perceptron (MLP).
  - An MLP has multiple layers of perceptrons; the outputs of the first layer of perceptrons are fed into a subsequent layer, and so on. Ultimately the responses of the final layer are brought together to compute an overall value for the network response.

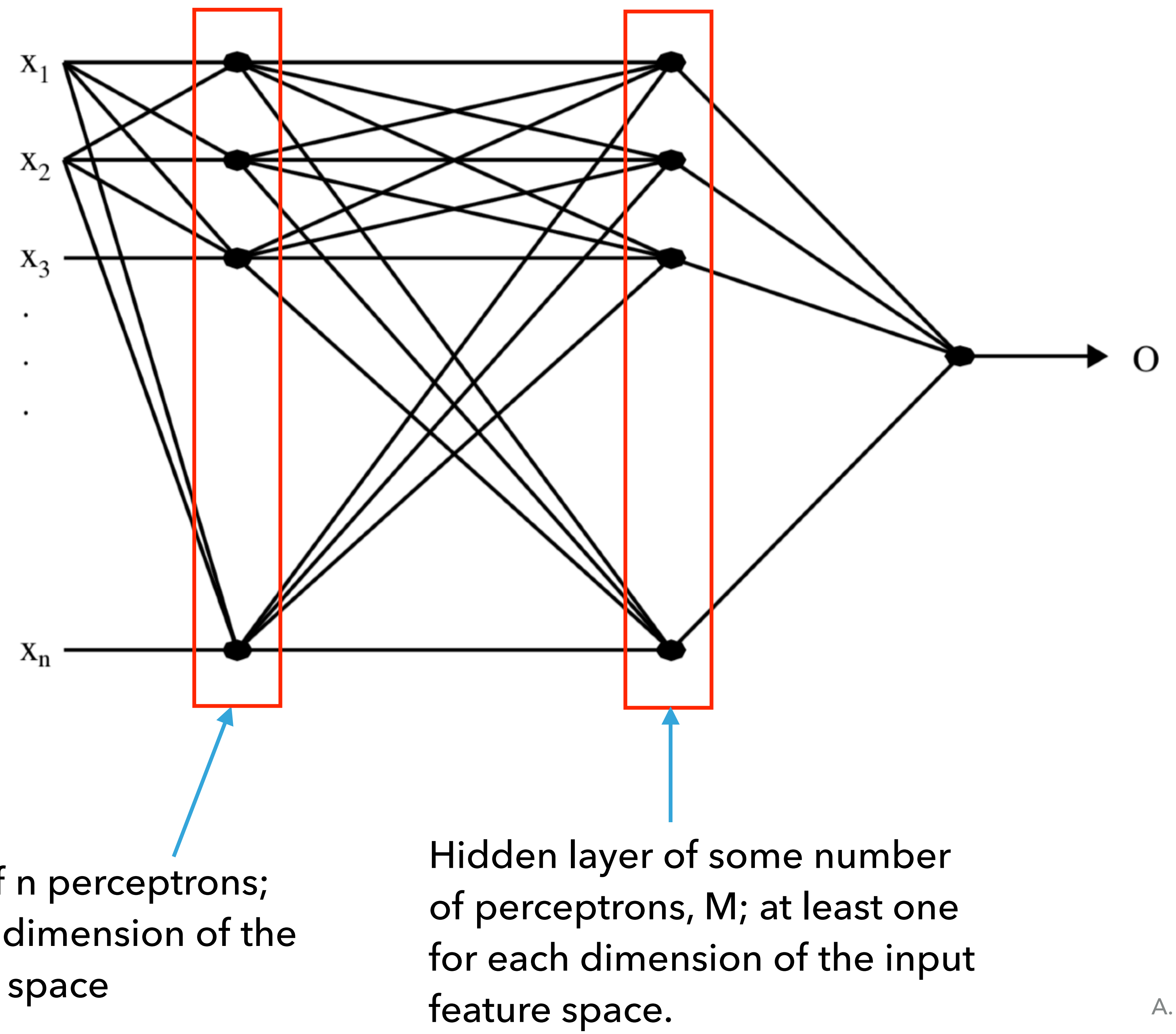
MULTILAYER PERCEPTRONS

- Illustrative example: Input data example: $x = \{x_1, x_2, x_3, \ldots, x_n\}$



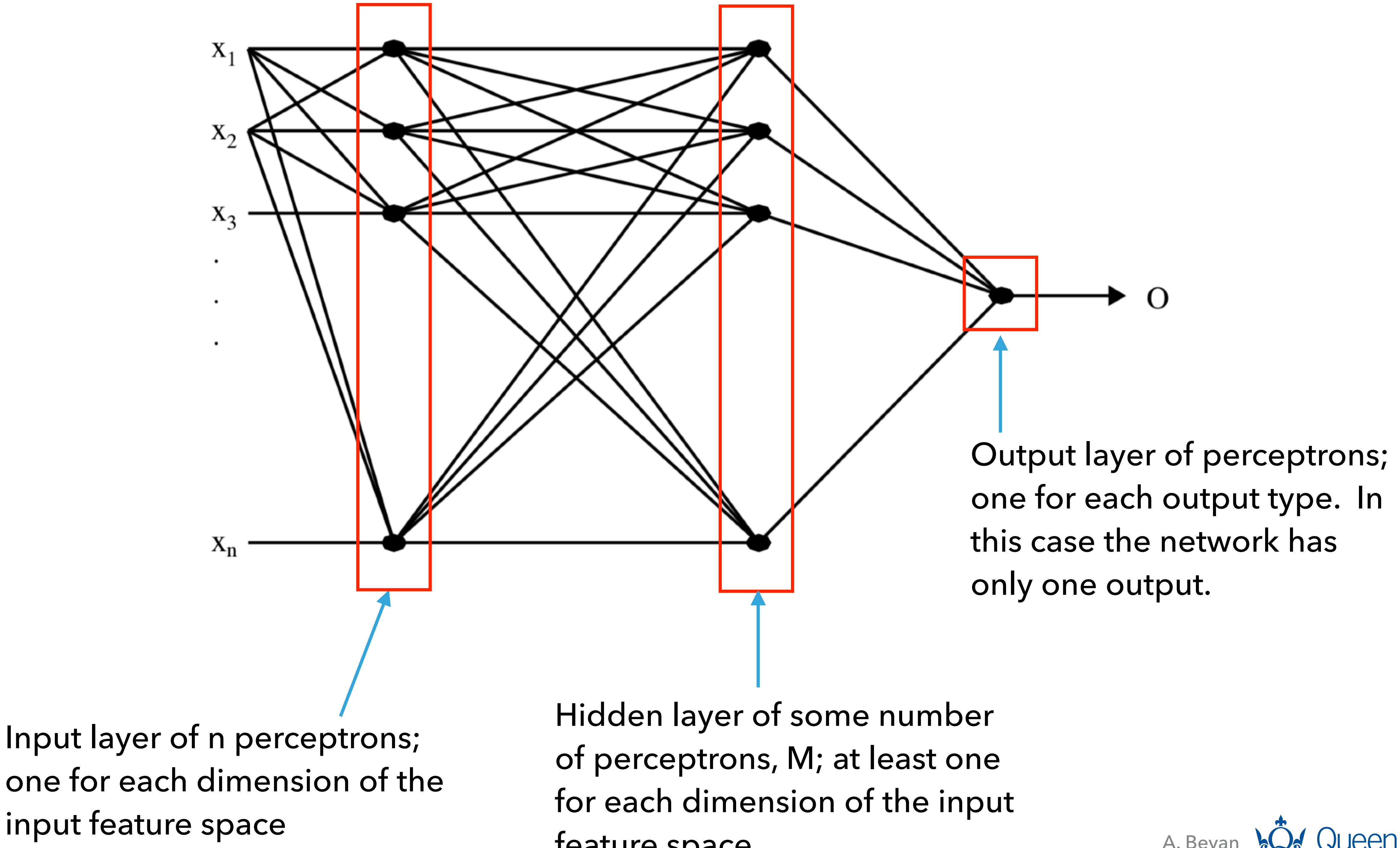
MULTILAYER PERCEPTRONS

- Illustrative example: Input data example: $x = \{x_1, x_2, x_3, \ldots, x_n\}$



MULTILAYER PERCEPTRONS

- Illustrative example: Input data example: $x = \{x_1, x_2, x_3, \ldots, x_n\}$



TRAINING

- Parameter tuning is referred to as training.
  - A perceptron of the form $f(w^T x+\theta)$ has n+1=dim(x)+1 hyper-parameters to be tuned.
  - The input layer of perceptrons in an MLP has n(n+1) hyper parameters to be tuned.
  - … and so on.
- We tune parameters based on some metric[1] called the loss function.
- We optimise the hyper-parameters of a network in order to minimise the loss function for an ensemble of data.
- The process is discussed in more detail under the heading Optimisation.

[1]Also called a figure of merit. The general term when applied to machine learning is the loss function.

SUMMARY

- Neural networks are built on perceptrons:
  - Inspired by desire to understand the biological function of the eye and how we perceive based on visual input.
  - The output threshold of a perceptron can be all or nothing, or be continuous between those extremes.
- Artificial neural networks are constructed from perceptrons.
- Perceptron/network weights need to be determined via some optimisation process, called training.
- … This leads us on to issues related to training and toward deep neural networks.

SUGGESTED READING

- The suggestions made here are for some of the standard text books on the subject. These require a higher level of math than we use in this course, but may have less emphasis on the practical application of the methods we discuss here as a consequence.

- MacKay: *Information theory, inference and learning algorithms*
  - Chapter: V
- C. Bishop: *Neural Networks for Pattern Recognition*
  - Chapters: 3 and 4
- C. Bishop: *Pattern Recognition and Machine Learning*
  - Chapter: 5
- T. Hastie, R. Tibshirani, J. Friedman, *Elements of statistical learning*
  - Chapter: 11

- In addition to books, you may find interesting articles posted on the preprint archive: https://arxiv.org. There are several useful categories as part of the Computing Research Repository (CoRR) related to this course including *Artificial Intelligence*. Note that these are research papers, so again they will generally have a strong mathematical content.