ALEXANDER BOOTH, LINDA CREMONESI,
ABBEY WALDRON

PRACTICAL MACHINE LEARNING

Queen Mary
University of London

ABOUT ME

- Dr Abbey Waldron [she/her] (call me Abbey)
- Neutrino physicist working on experiments in the US (MINERvA, NOvA, DUNE)
- Machine learning for detecting neutrinos in liquid argon time projection chambers (including Domain Adversarial Neural Networks, Graph Neural Networks, GANs etc)

LINDA CREMONESI

- Dr Linda Cremonesi [she/her] (Linda)
- Neutrino physicist working on experiments in the US (NOvA, DUNE)
- Analysis co-ordinator for NOvA, DUNE det-sim convenor

ALEXANDER BOOTH

- Dr Alexander Booth [he/him/his] (Alex)
- Neutrino physicist working on experiments in the US (NOvA, DUNE)
- NOvA reconstruction convenor, DUNE production expert

ABOUT YOU!

pollev.com/abbeywaldron047

THIS COURSE – 4 WORKSHOPS (LECTURE+EXERCISES)

- Lecture 1 (Monday) Introduction to machine learning with neural networks and linear regression
- Lecture 2 (Tuesday) Optimisation and non-linear regression with neural networks
- Lecture 3 (Wednesday) Classification and convolutional neural networks for image classification
- Lecture 4 (Thursday) Robustness and adversarial examples to image classification problems

SOFTWARE

- We will be using Python and TensorFlow 2.11 throughout the course
- We don't assume any knowledge of TensorFlow, but you should already have a good command of Python

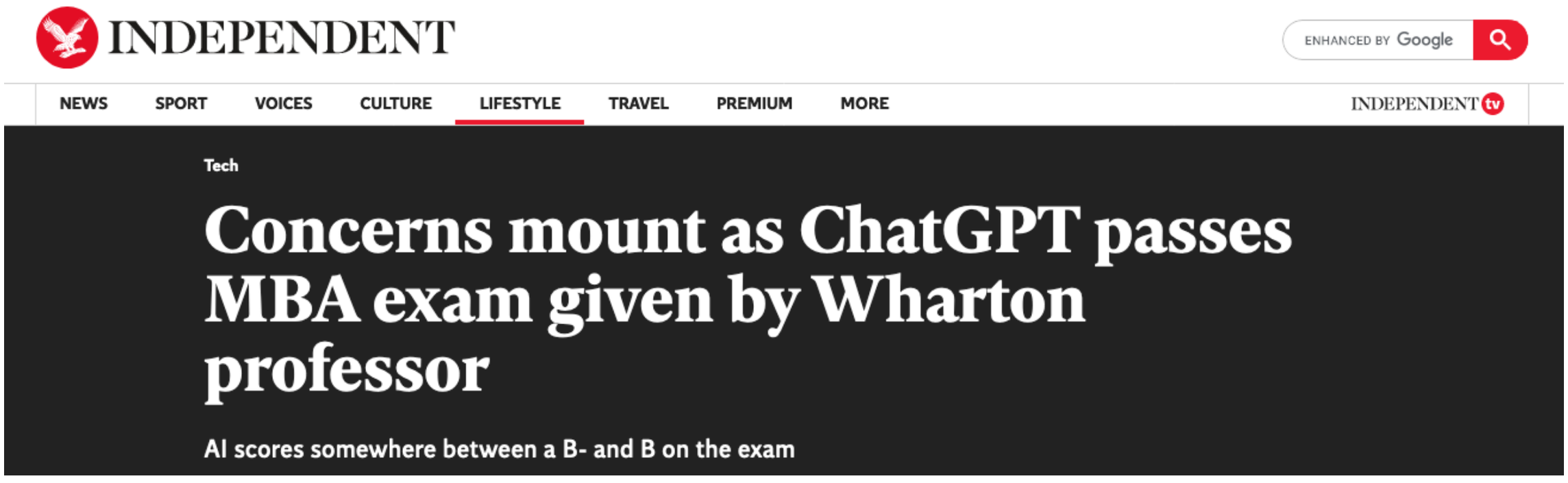
EXAMPLES OF MACHINE LEARNING

CAN YOU THINK OF EXAMPLES OF MACHINE LEARNING?

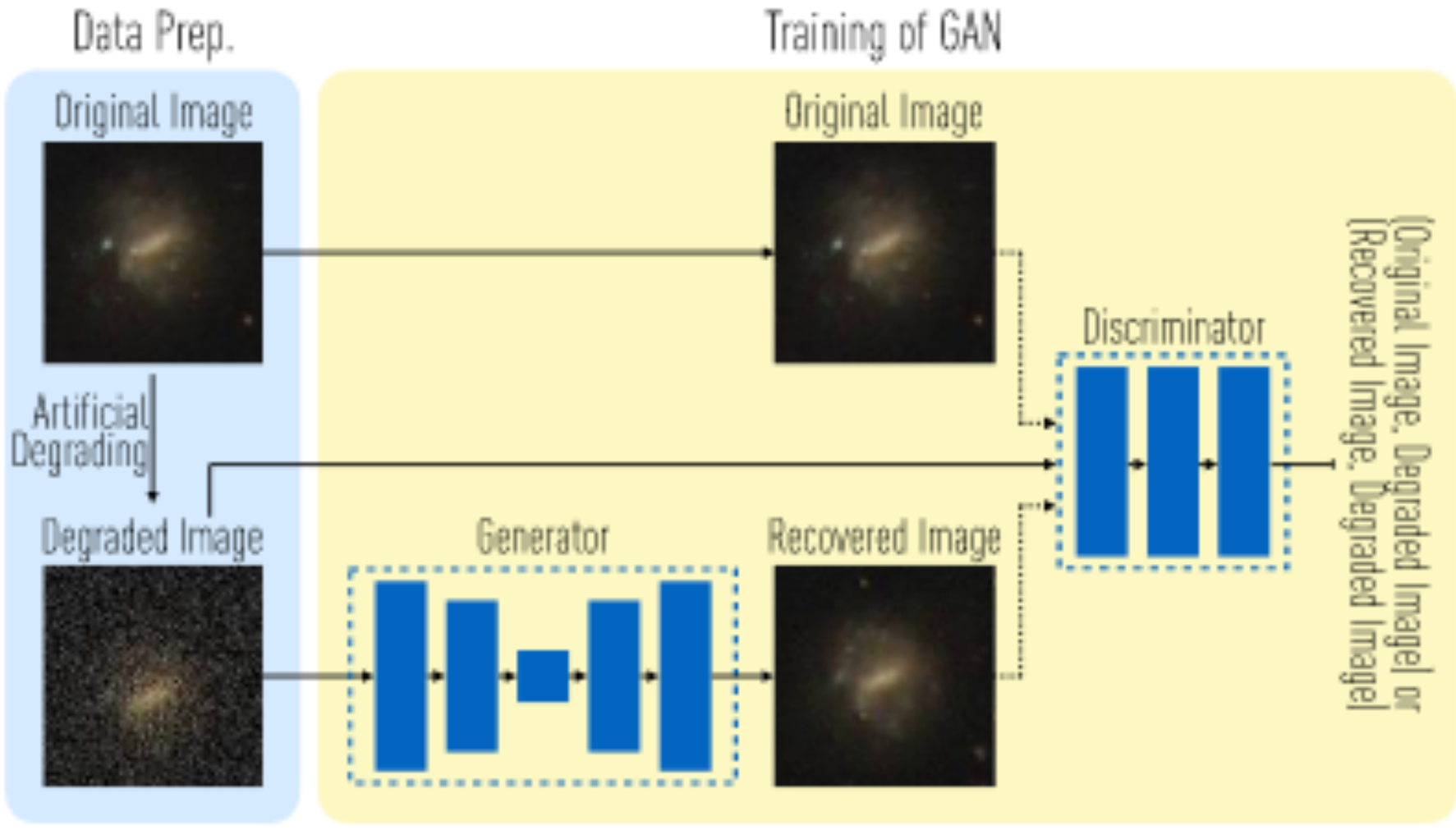
EXAMPLES OF MACHINE LEARNING

- Image recognition - human level on some tasks since 2012
- Speech recognition - consumer level
- Object detection e.g. in “self”-driving cars
- Particle identification in fundamental particle physics
- Art generation
- …

CHATGPT MBA

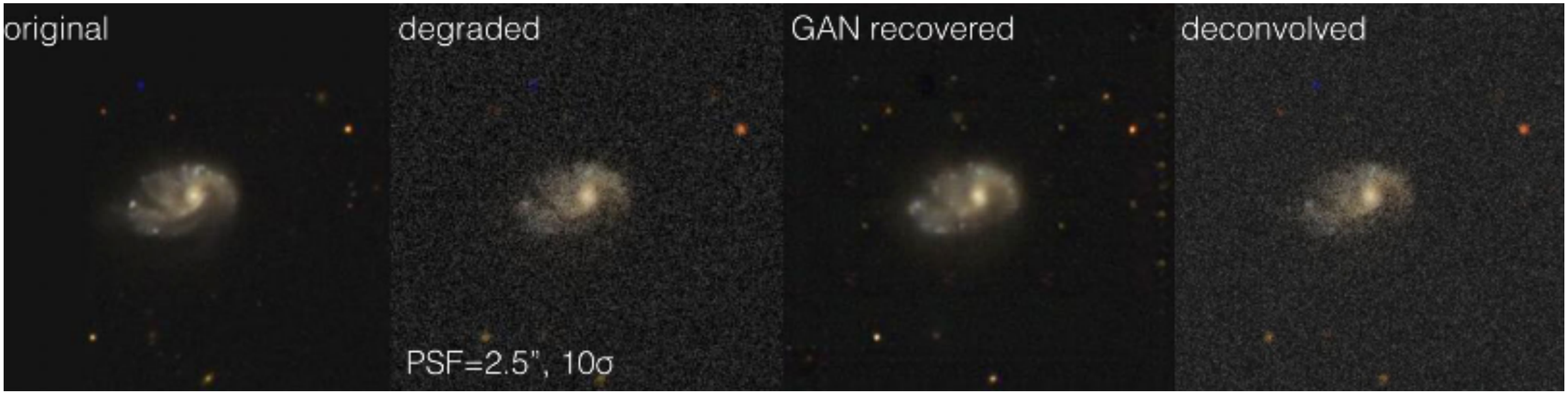

INDEPENDENT
ENHANCED BY Google
NEWS
SPORT
VOICES
CULTURE
LIFESTYLE
TRAVEL
PREMIUM
MORE
INDEPENDENT tv
Tech
Concerns mount as ChatGPT passes MBA exam given by Wharton professor
AI scores somewhere between a B- and B on the exam

GALAXY GAN

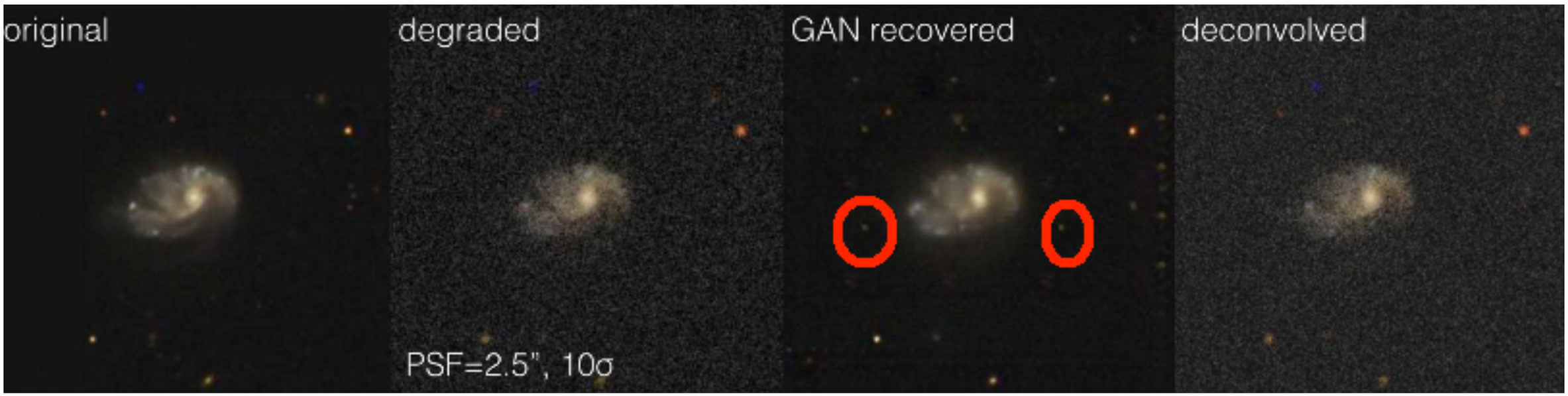


K. Schawinski et al. MNRAS467,L110–L114 (2017)

GALAXY GAN



GALAXY GAN



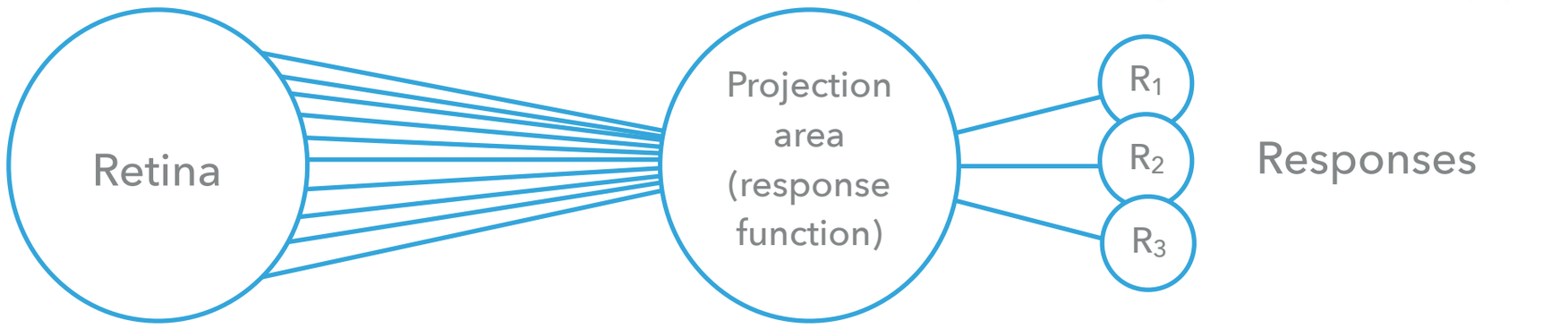
PRACTICAL MACHINE LEARNING

NEURAL NETWORKS

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

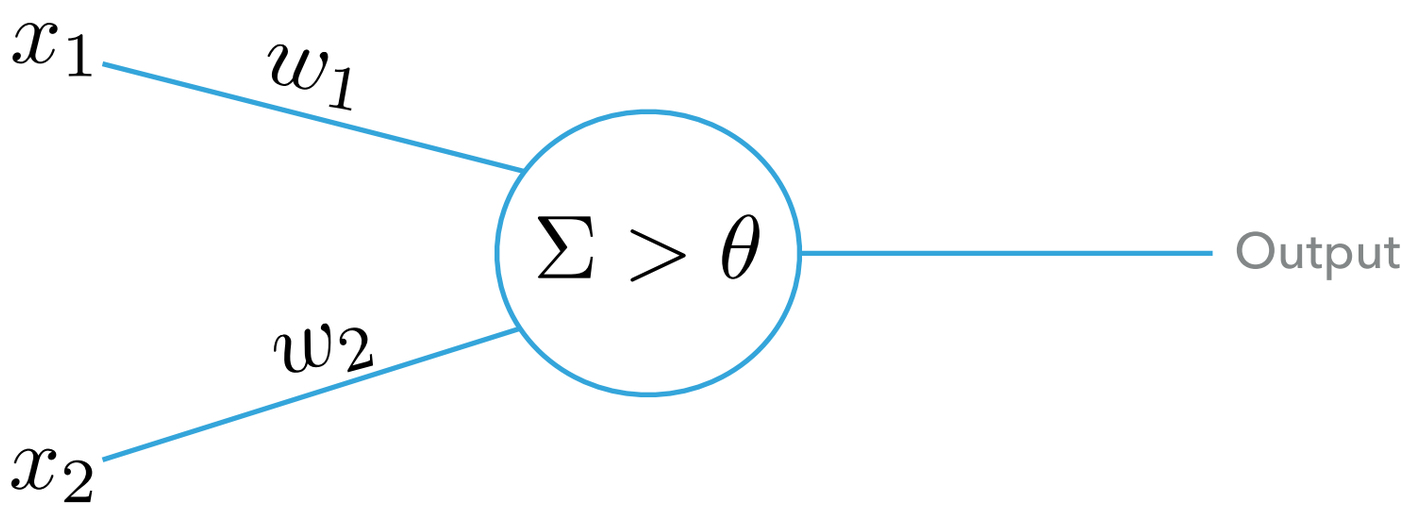
- Illustrative example:
  - Consider a measurement of two quantities x_1, and x_2.
  - Based on these measurements we determine if the perceptron is to give an output (value = 1) or not (value = 0).

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

[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

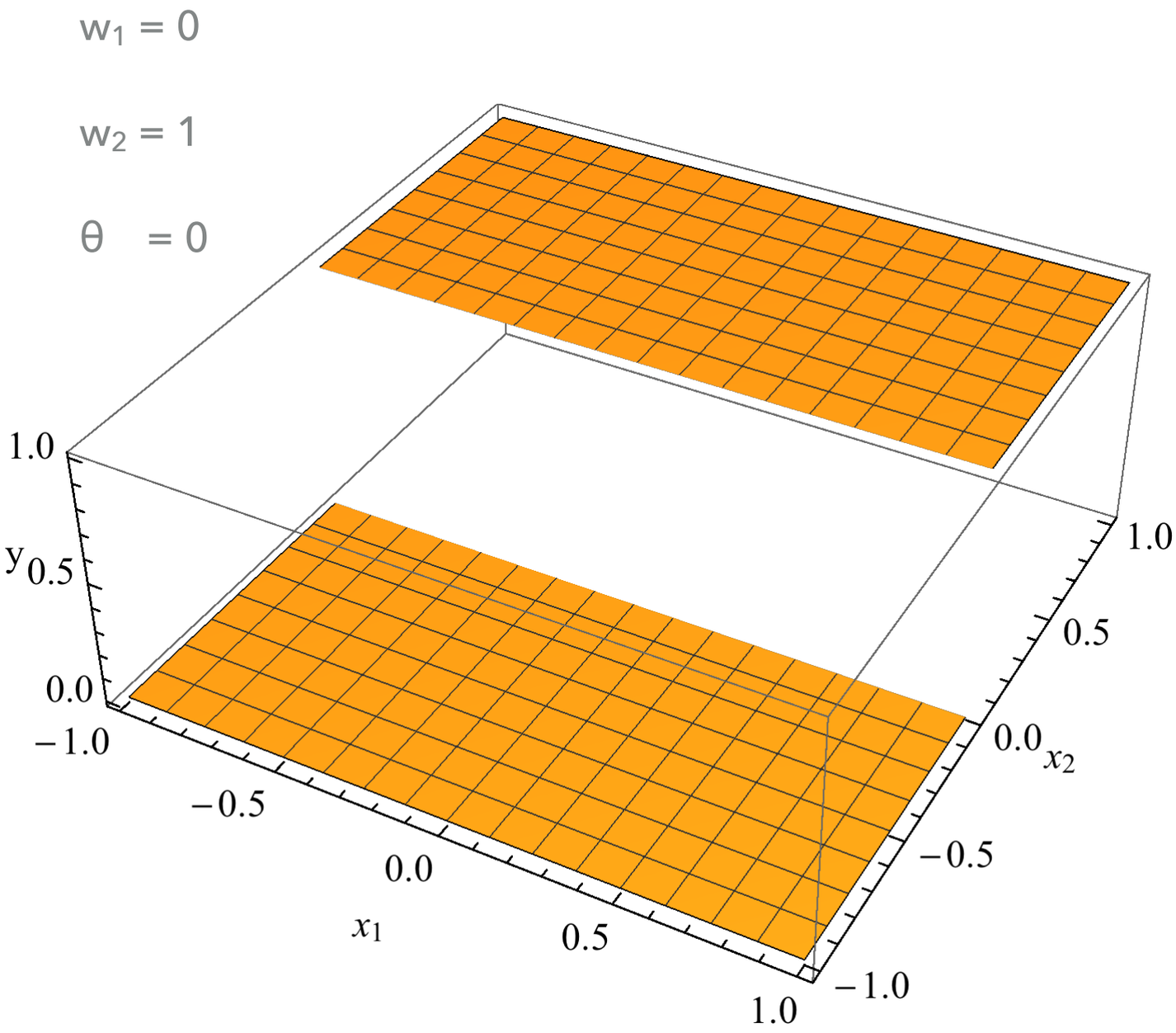
[1] F. Rosenblatt, Psych. Rev. **65** p386-408, 1958.

PERCEPTRONS

- Illustrative example:

- 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 example:

- Decision is made on x_2

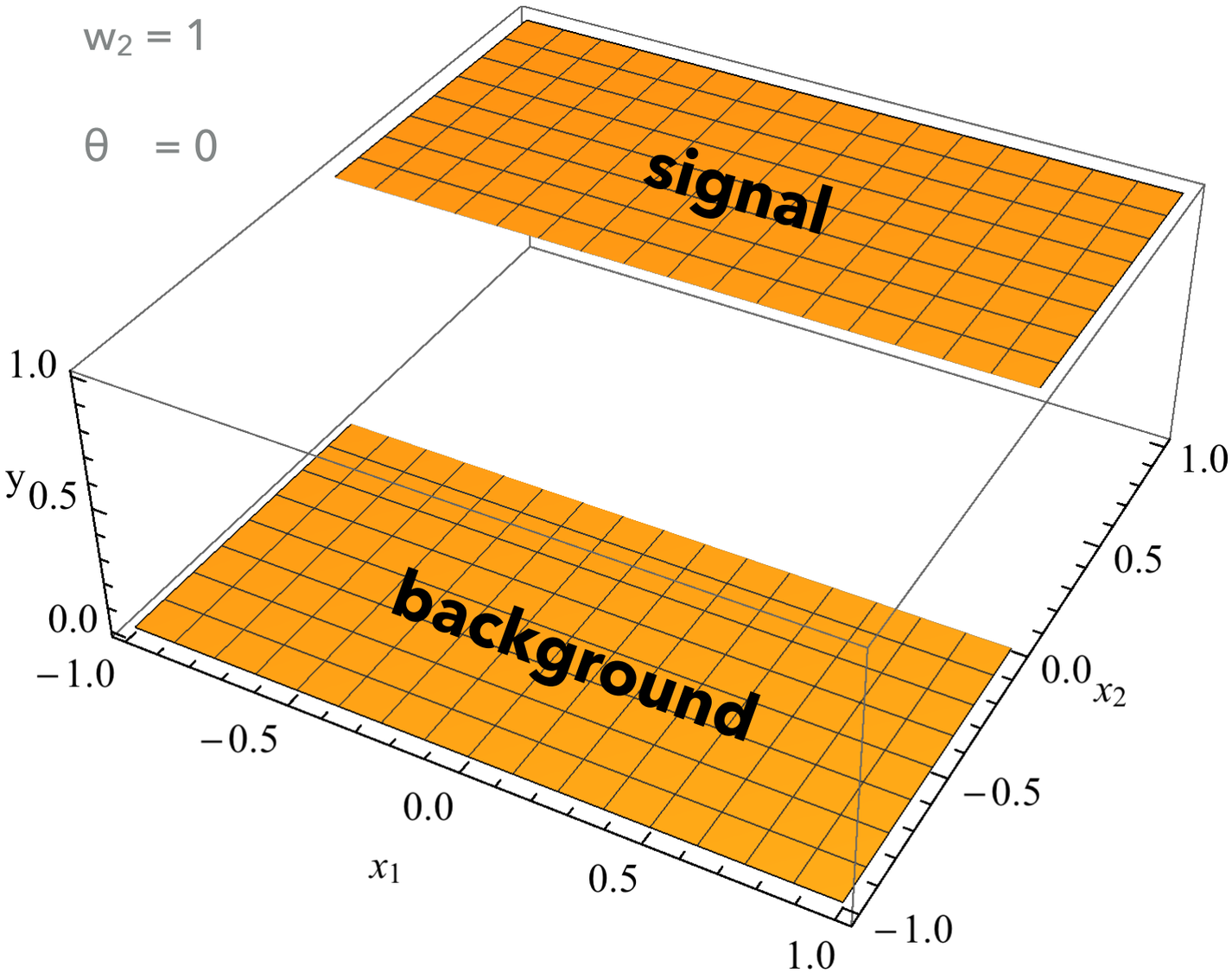
- Output value is either 1 or 0 as some $f(x_1, x_2)$ that depends on the values of w_1, w_2 and θ.

$w_1 = 0$

$w_2 = 1$

$\theta = 0$

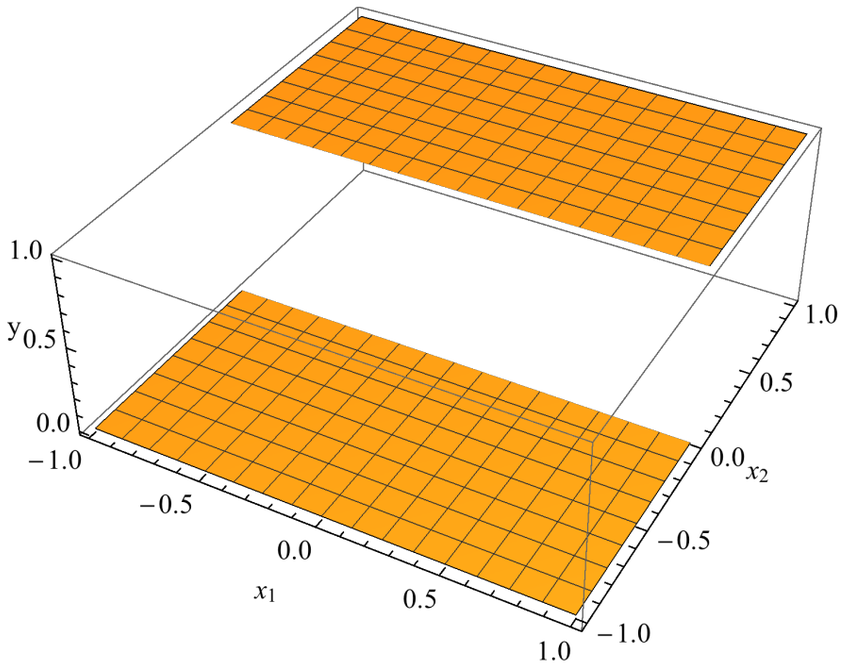
In particle physics we often use machine learning to suppress background. Here y=1 corresponds to signal and y=0 corresponds to background.



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

PERCEPTRONS

- Illustrative examples:

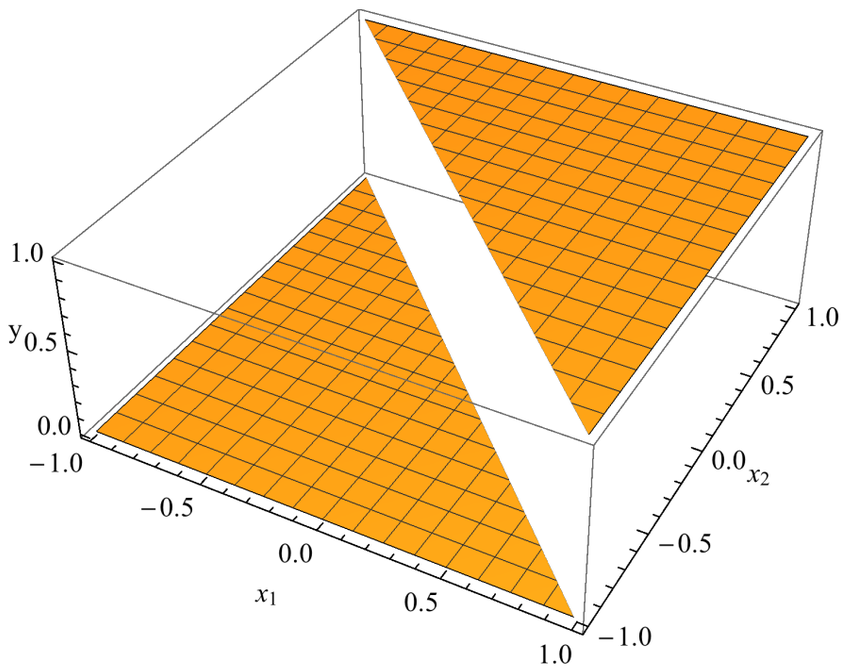


$w_1 = 0$

$w_2 = 1$

$\theta = 0$

Baseline for comparison, decision only on value of x_2

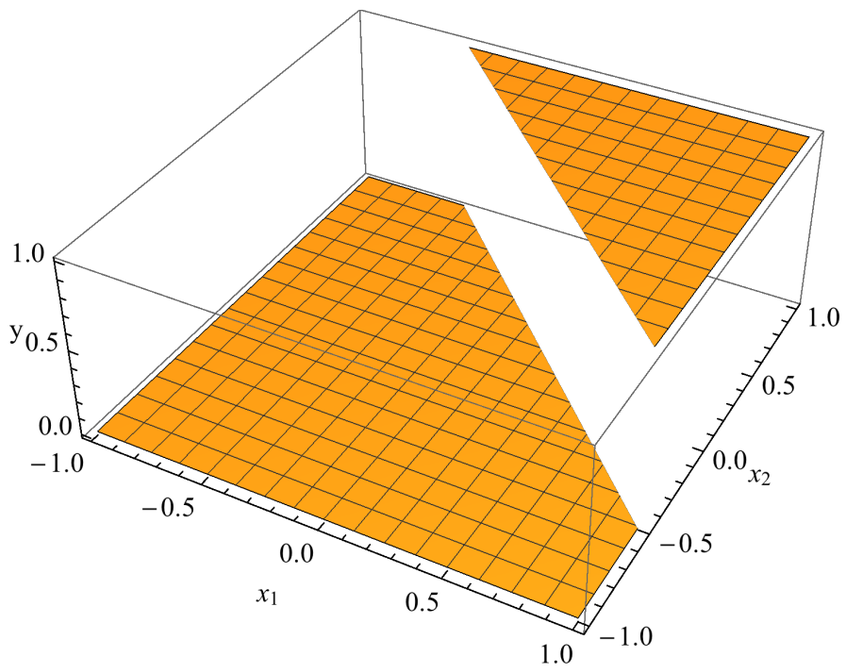


$w_1 = 1$

$w_2 = 1$

$\theta = 0$

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

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

[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

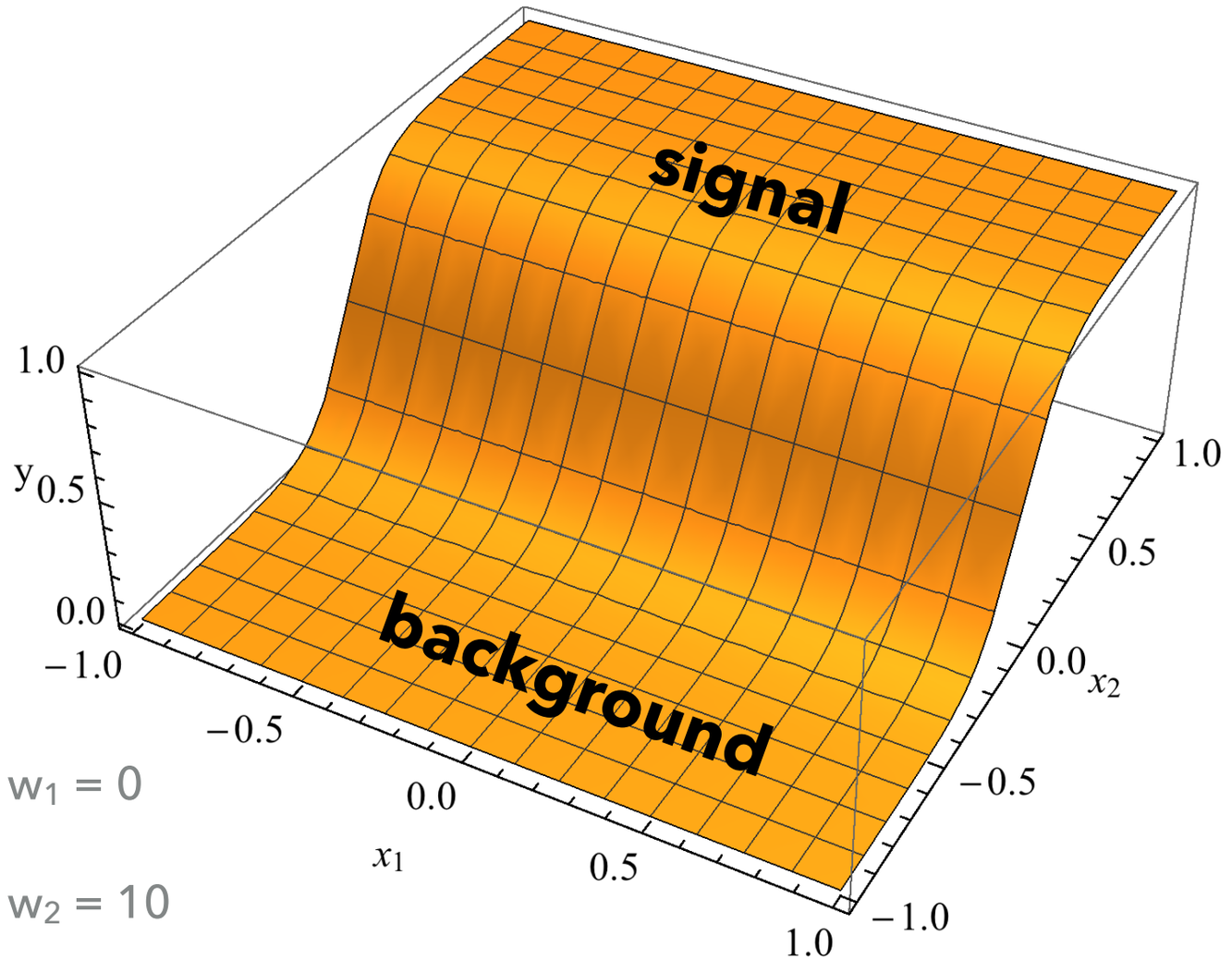
- TensorFlow has the following activation functions (`tf.nn.`**`ACTIVATIONFUNCTION`**)
  - relu (covered here)
  - leaky_relu (covered here)
  - relu6
  - crelu
  - elu
  - selu
  - softplus
  - softsign
  - dropout
  - bias_add
  - sigmoid (covered here)
  - tanh (covered here)

ACTIVATION FUNCTIONS: LOGISTIC (OR SIGMOID)

- A common activation function used in neural networks:

$$y = \frac{1}{1 + e^{w^T x + \theta}}$$

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



$w_1 = 0$

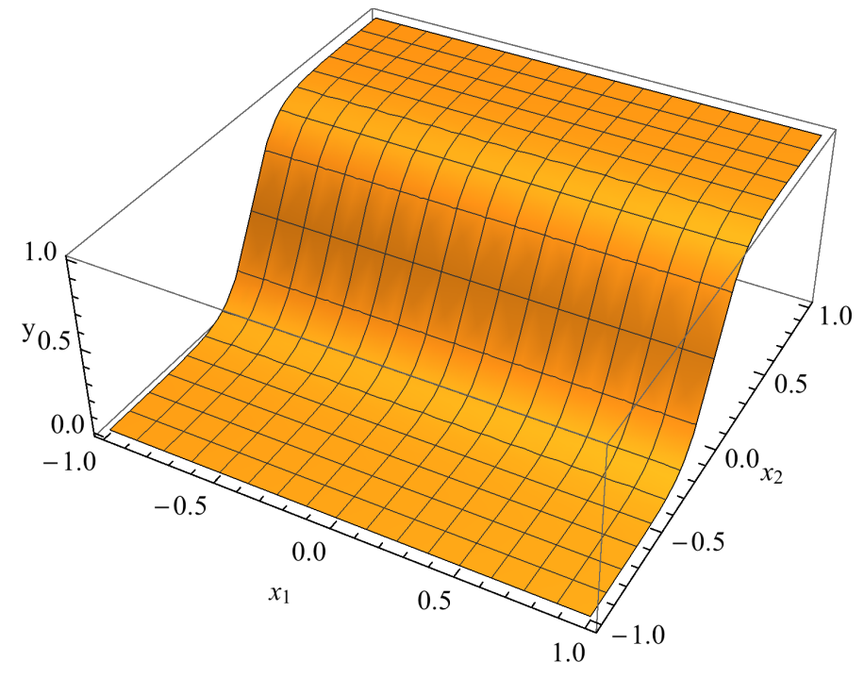
$w_2 = 10$

$\theta = 0$

ACTIVATION FUNCTIONS: LOGISTIC (OR SIGMOID)

$$\frac{1}{1 + e^{(w_1 x_1 + w_2 x_2 + \theta)}}$$

- A common activation function used in neural networks:

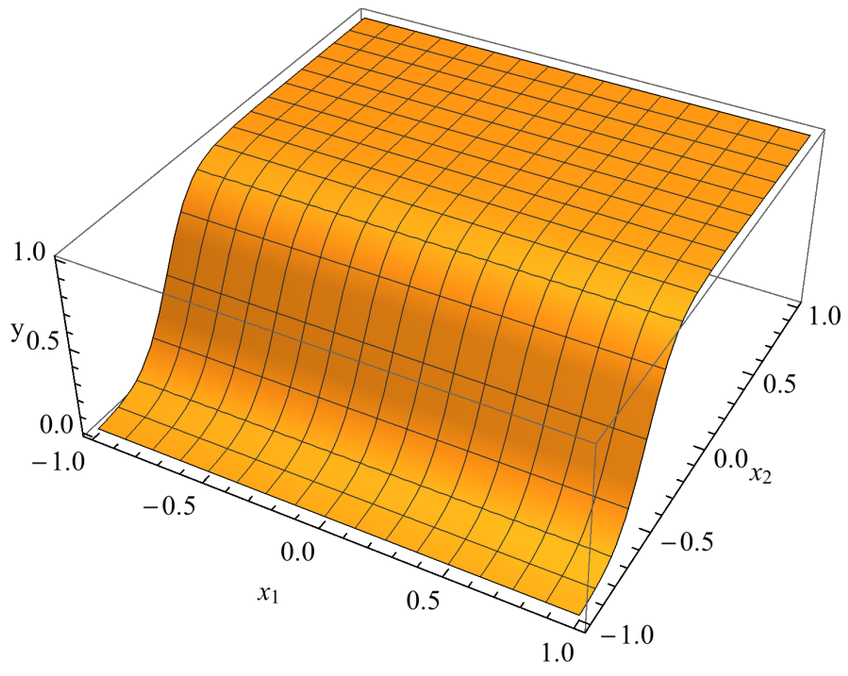


$w_1 = 0$

$w_2 = 10$

$\theta = 0$

Baseline for comparison, decision only on value of x_2

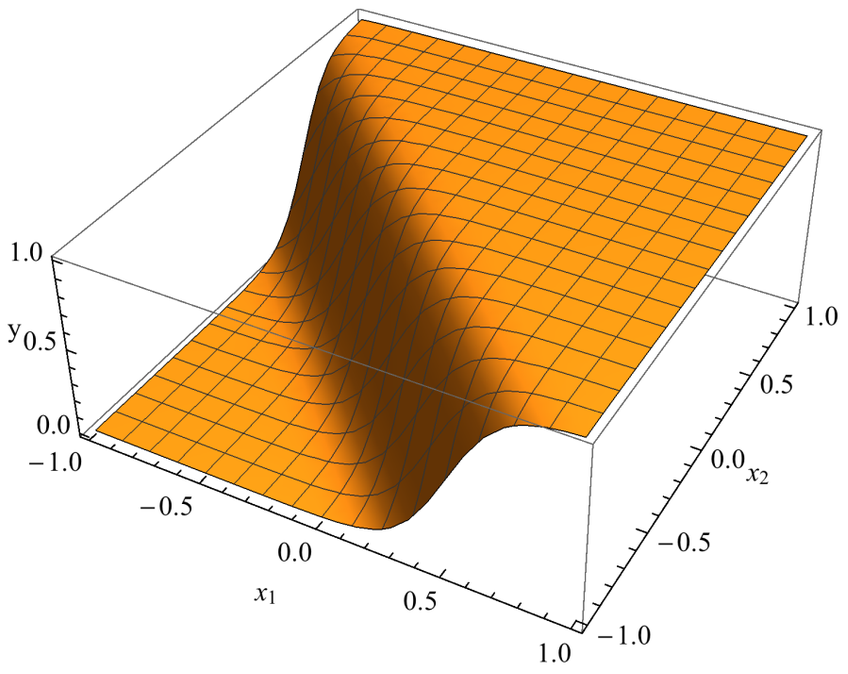


$w_1 = 0$

$w_2 = 10$

$\theta = -5$

Offset from zero using θ



$w_1 = 10$

$w_2 = 10$

$\theta = -5$

rotate "decision boundary" in (x_1, x_2)

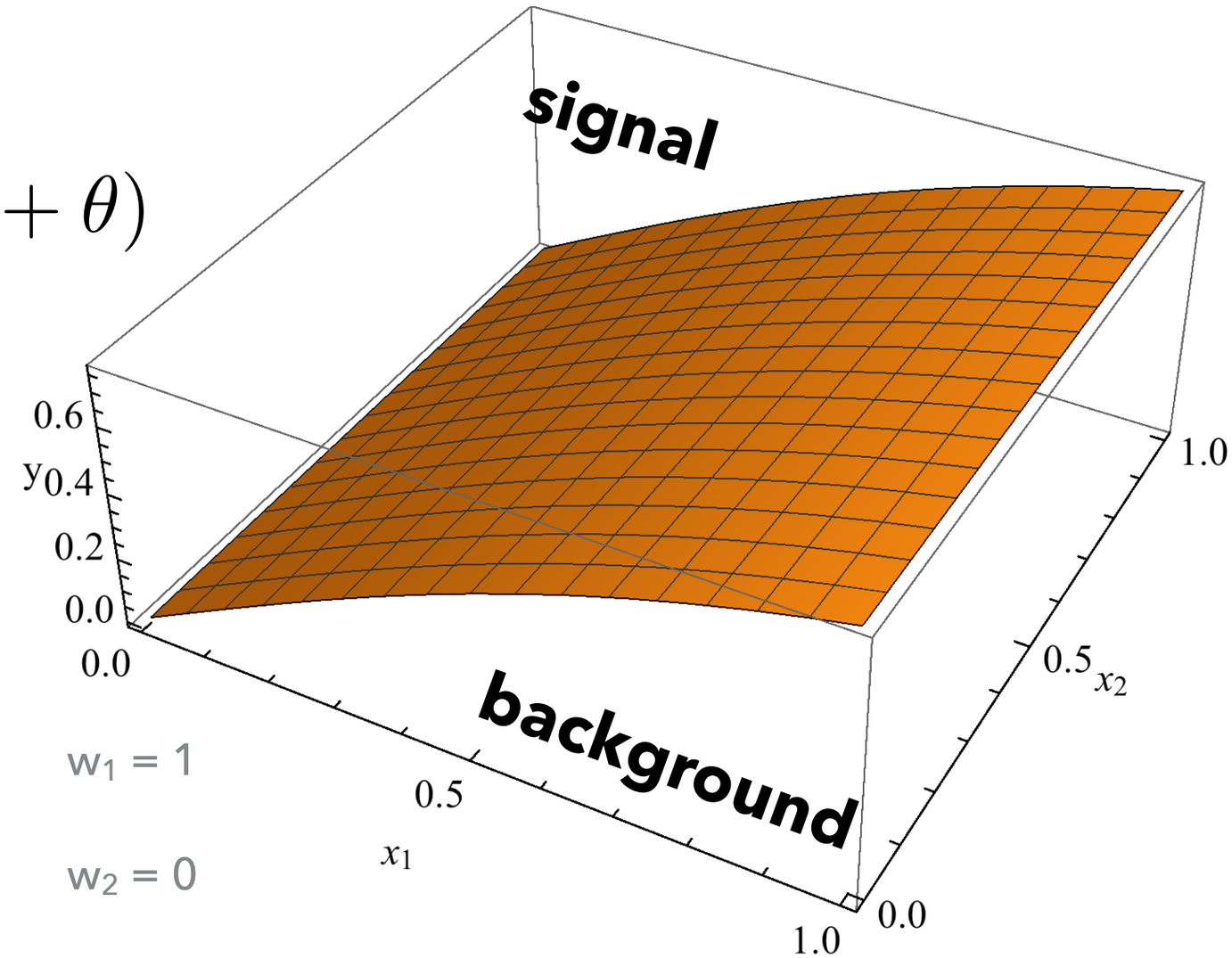
Queen Mary University of London

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)



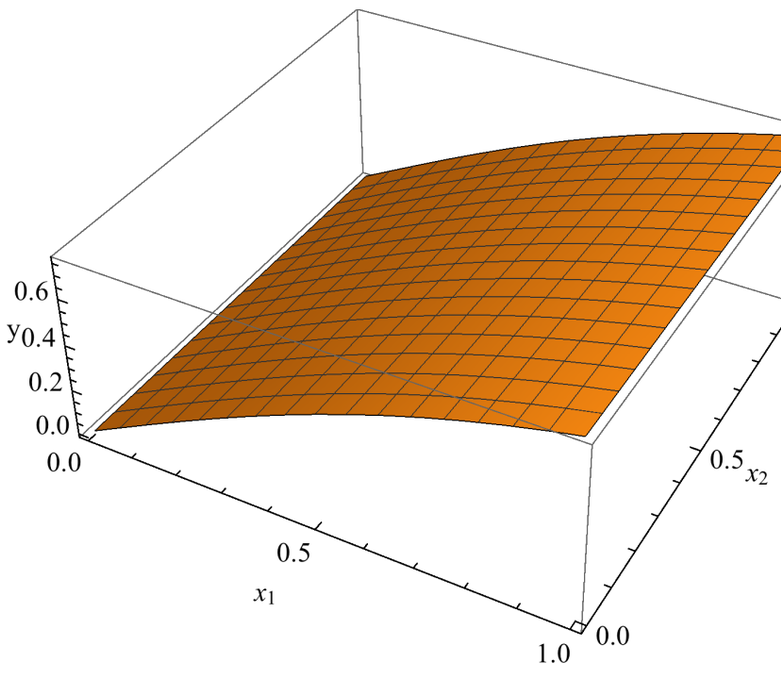
$w_1 = 1$

$w_2 = 0$

$\theta = 0$

ACTIVATION FUNCTIONS: HYPERBOLIC TANGENT $\tanh(w_1 x_1 + w_2 x_2 + \theta)$

- A common activation function used in neural networks:

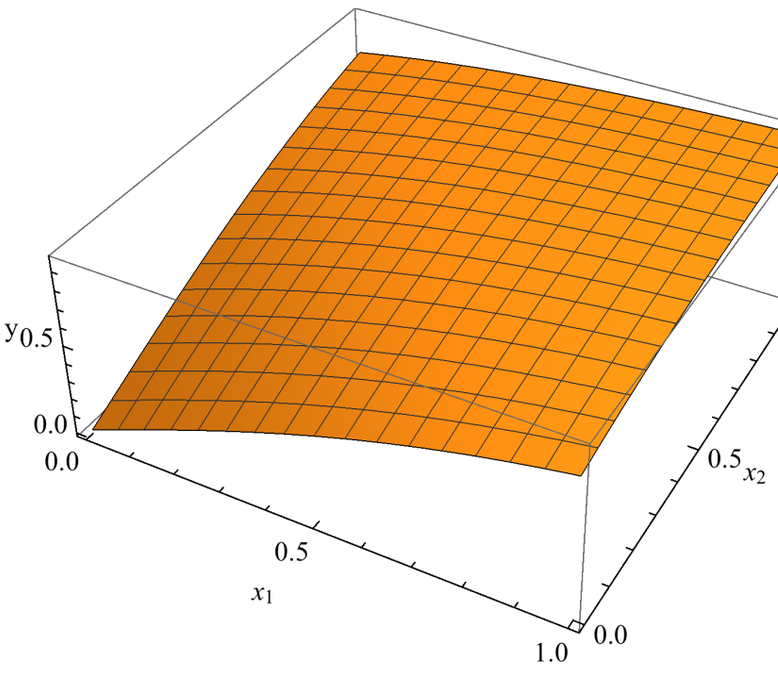


$w_1 = 1$

$w_2 = 0$

$\theta = 0$

Baseline for comparison, decision only on value of x_1

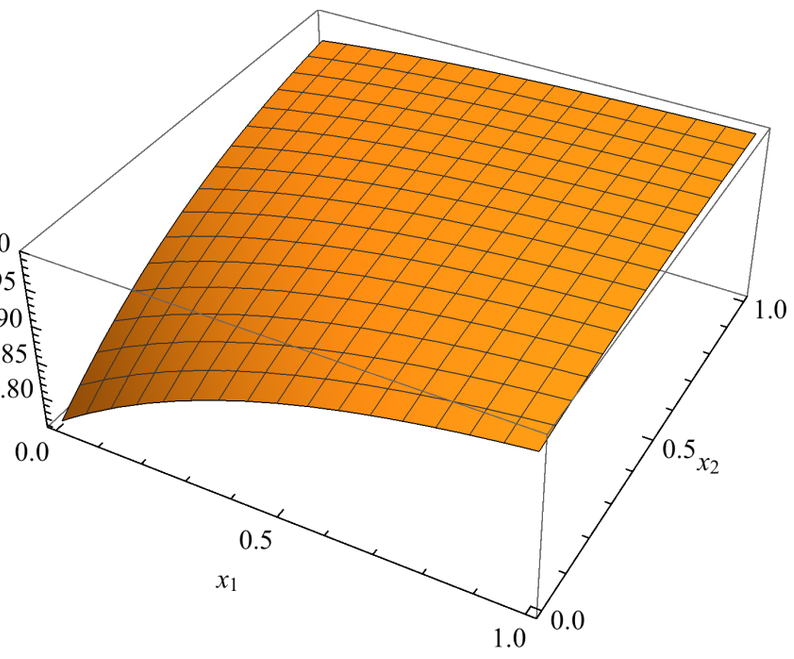


$w_1 = 1$

$w_2 = 1$

$\theta = 0$

rotate "decision boundary" in (x_1, x_2)



$w_1 = 1$

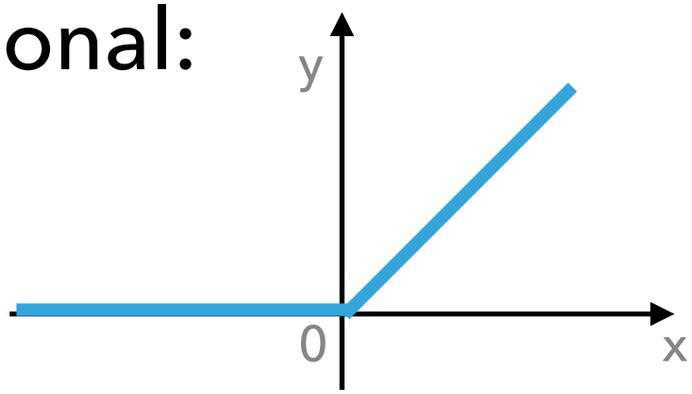
$w_2 = 1$

$\theta = -1$

Offset (vertically) from zero using θ

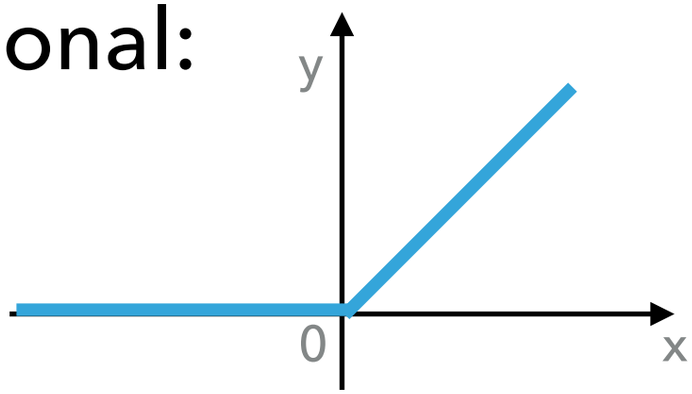
ACTIVATION FUNCTIONS: RELU

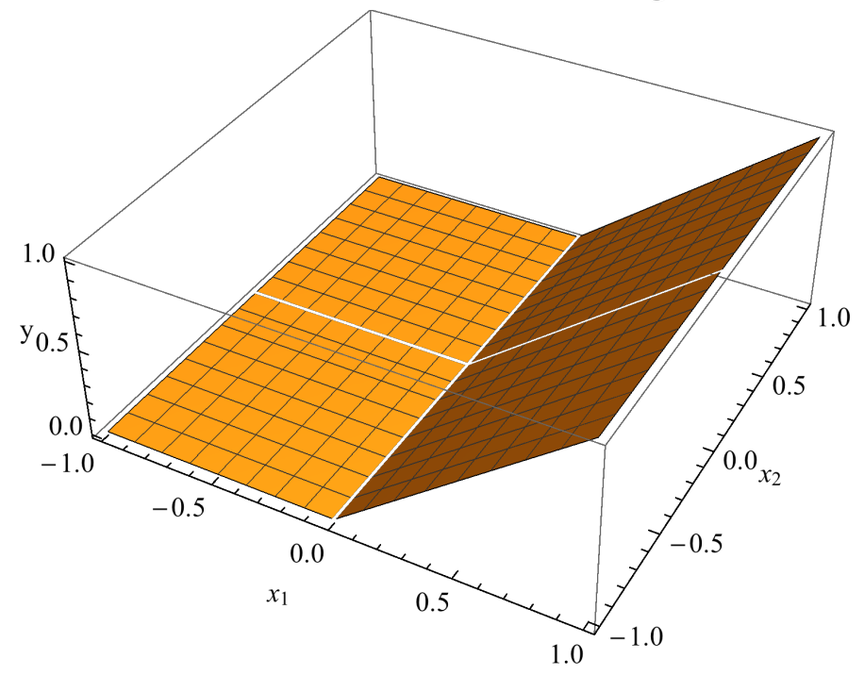
- The **Re**ctified **L**inear **U**nit activation function is commonly used for CNNs. This is given by a conditional:
  - If (x < 0) y = 0
  - otherwise y = x



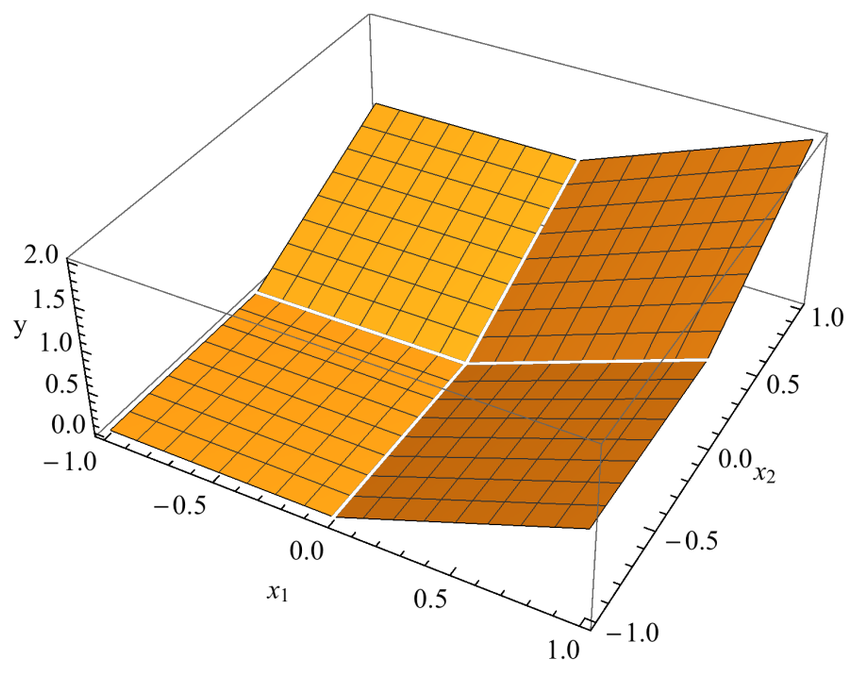
ACTIVATION FUNCTIONS: RELU

- 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 $y = x$

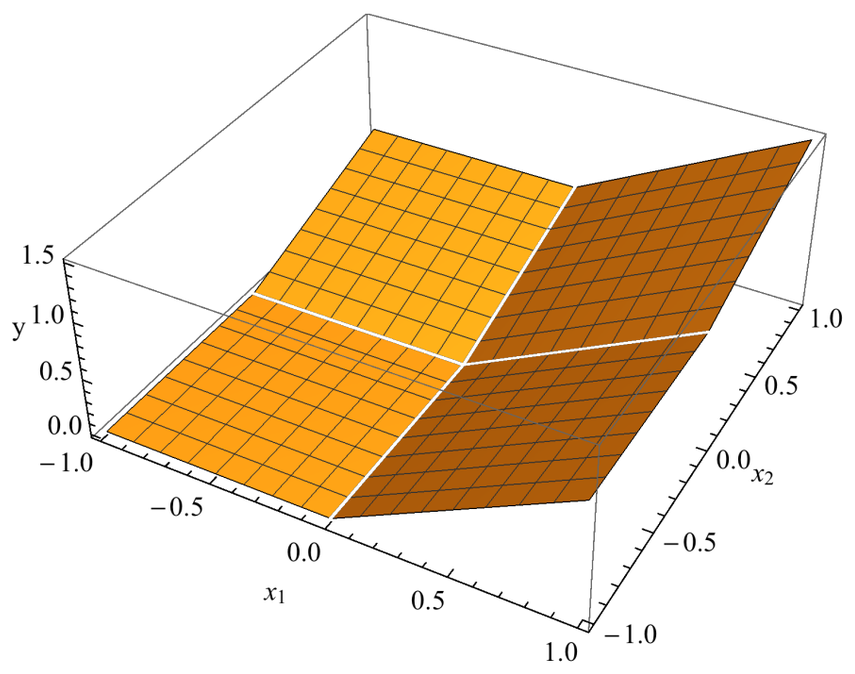




$w_1 = 1, w_2 = 0$



$w_1 = 1, w_2 = 1$

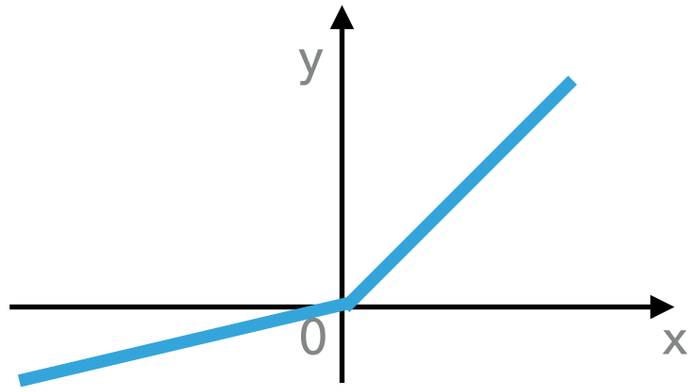


$w_1 = 1, w_2 = 0.5$

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].

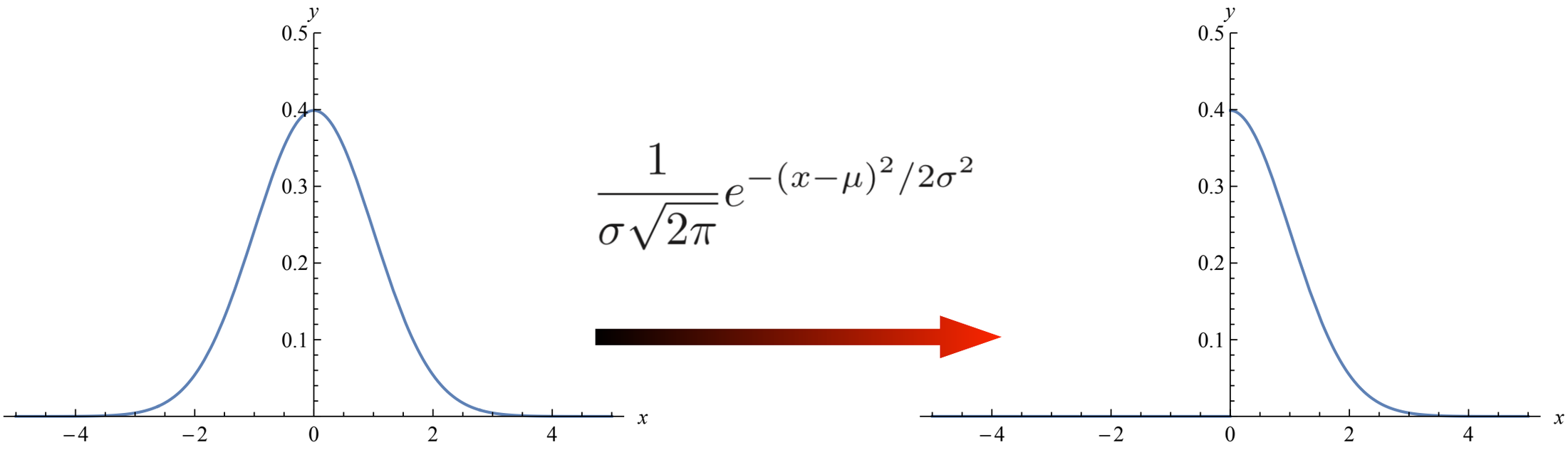
```
initial = tf.truncated_normal(shape, stddev=0.1)
```

- 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

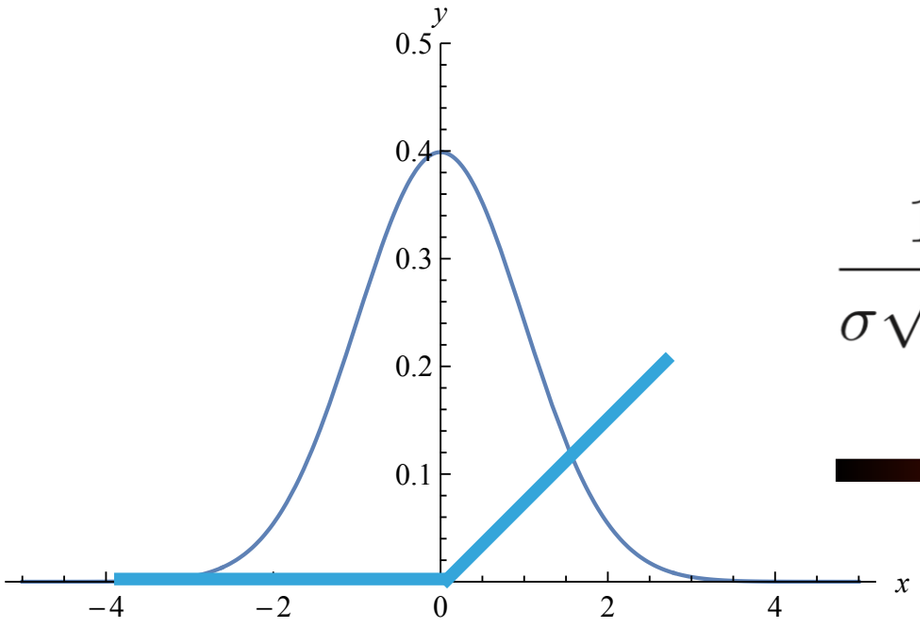
- N.B. Gradient descent optimisation algorithms will not change the weights for an activation function if the initial weight is set to zero.
  - This is why a truncated normal is used for initialisation, rather than a Gaussian that has x∈[-∞, ∞].



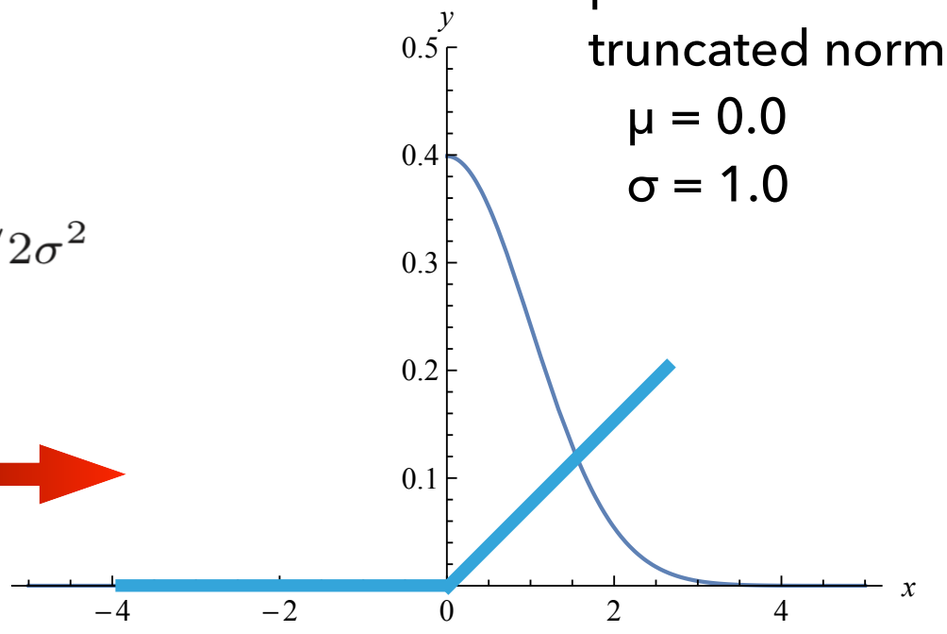
[1] Maas, Hannun, Ng, ICML2013.
[2] He, Zhang, Ren and Sun, arXiv:1502.01852

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 $x \in [-\infty, \infty]$.



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



TensorFlow default parameters for the truncated normal are:

$\mu = 0.0$

$\sigma = 1.0$

[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 parameters to find an optimal solution.
  - Avoids large or small 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 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.

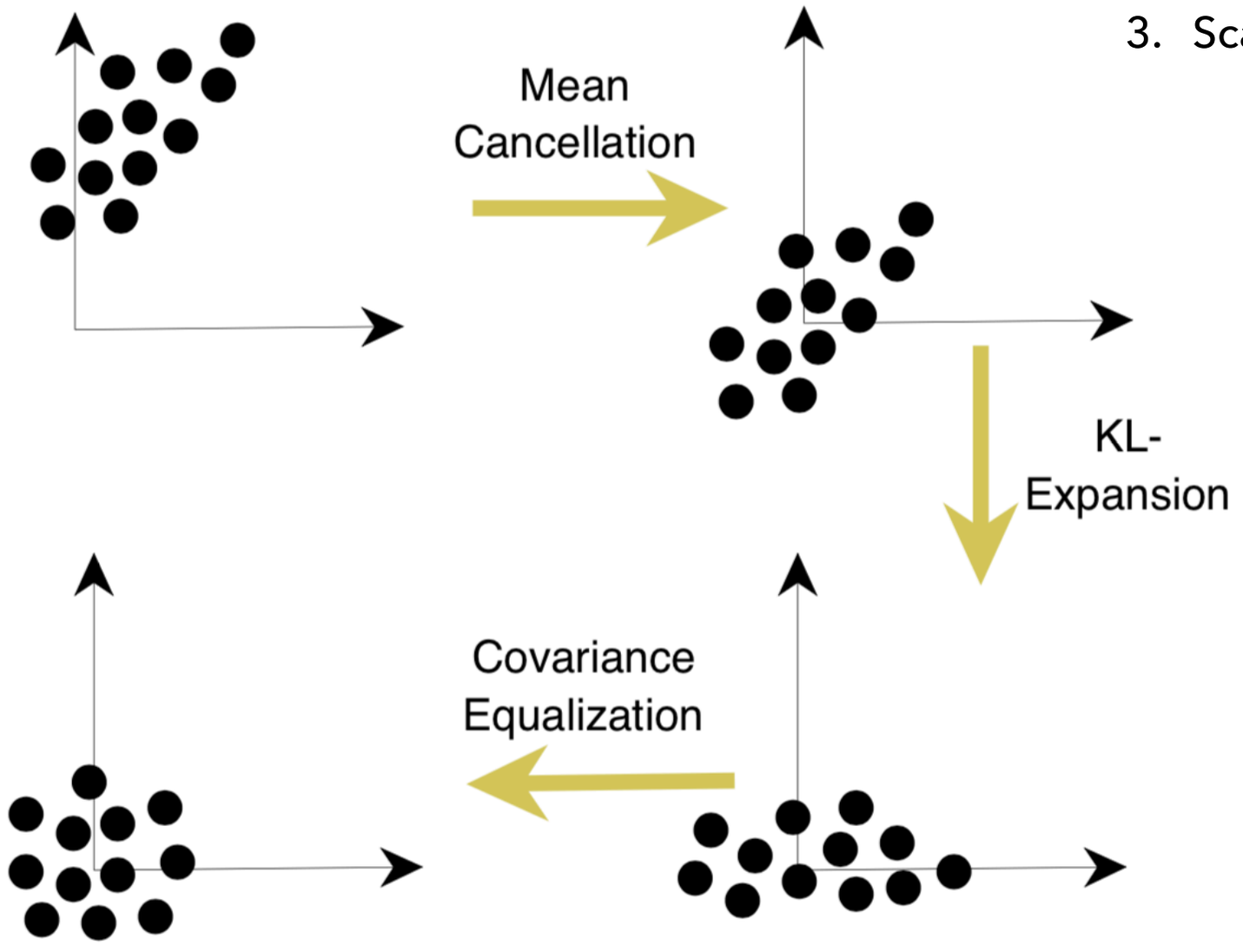
[1]Decorrelation of features is not essential assuming a sufficiently general optimisation algorithm is being used. The rationale is that in general if one can decorrelate features then we just have to minimise the cost as a function of weights for one feature at a time, rather than being concerned about 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.

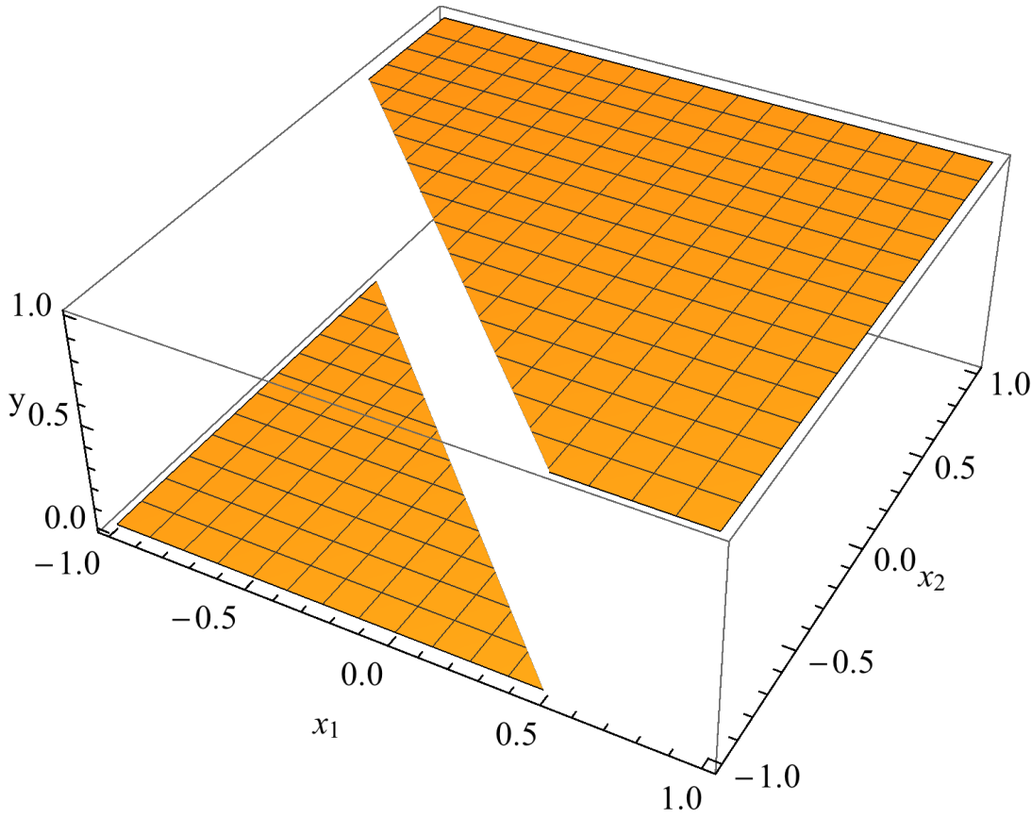
1. Shift the distribution to have a zero mean
2. Decorrelate input features
3. Scale to match covariance of features.



[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.

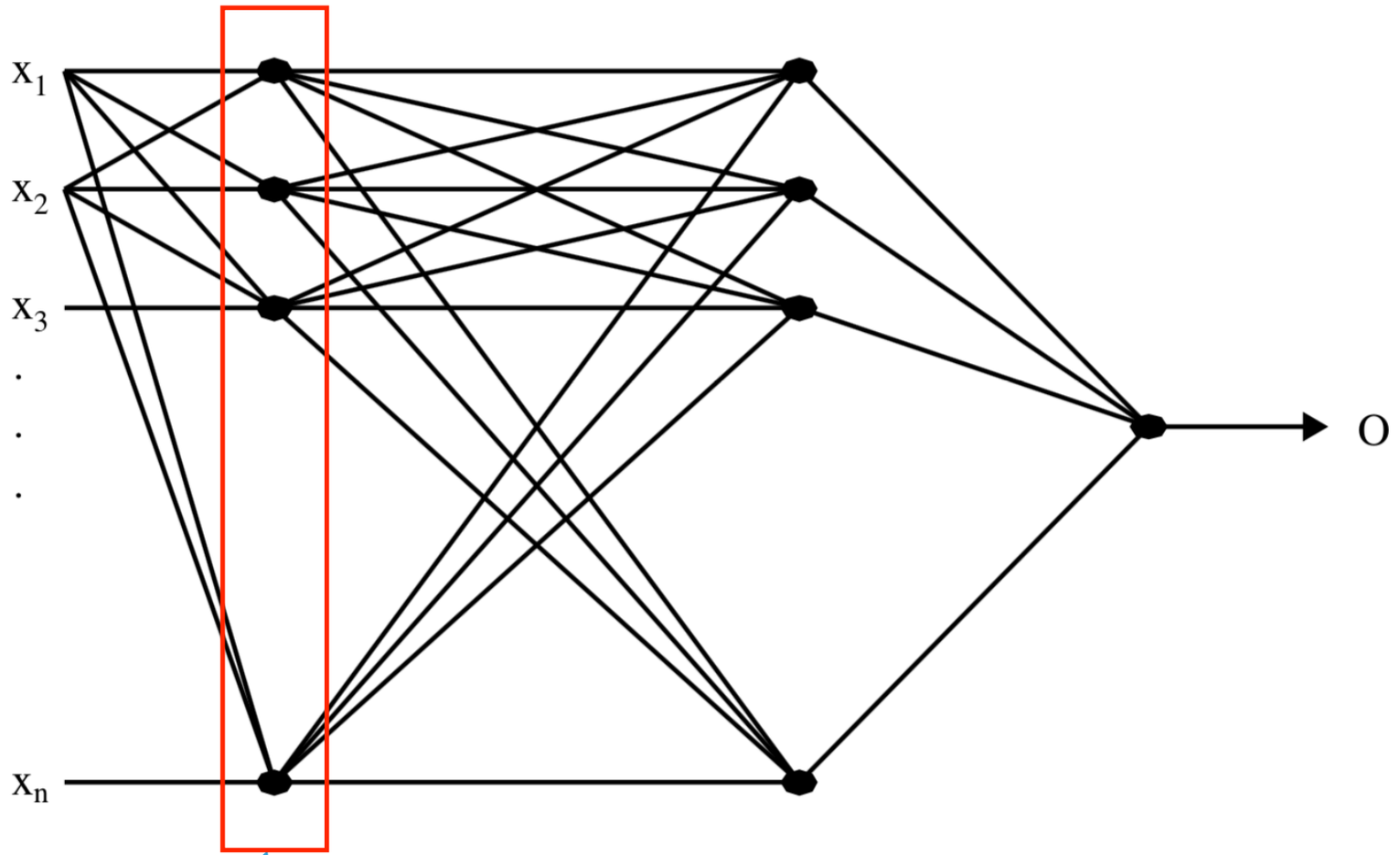


ARTIFICIAL NEURAL NETWORKS (ANNs)

- 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

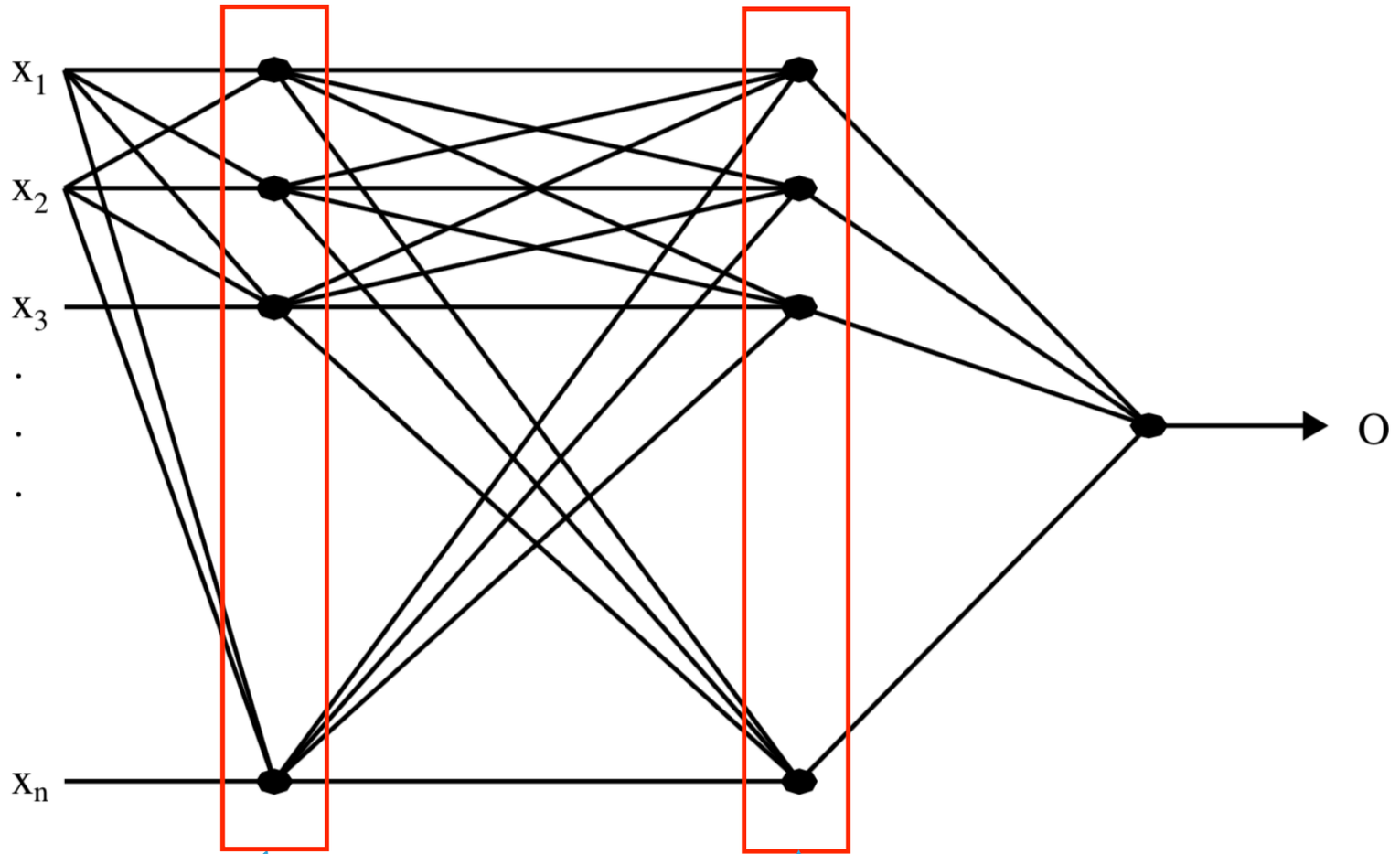
- Illustrative example: Input data example: $x = \{x_1, x_2, x_3, \ldots, x_n\}$



Input layer of n perceptrons; one for each dimension of the input feature space

MULTILAYER PERCEPTRONS

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

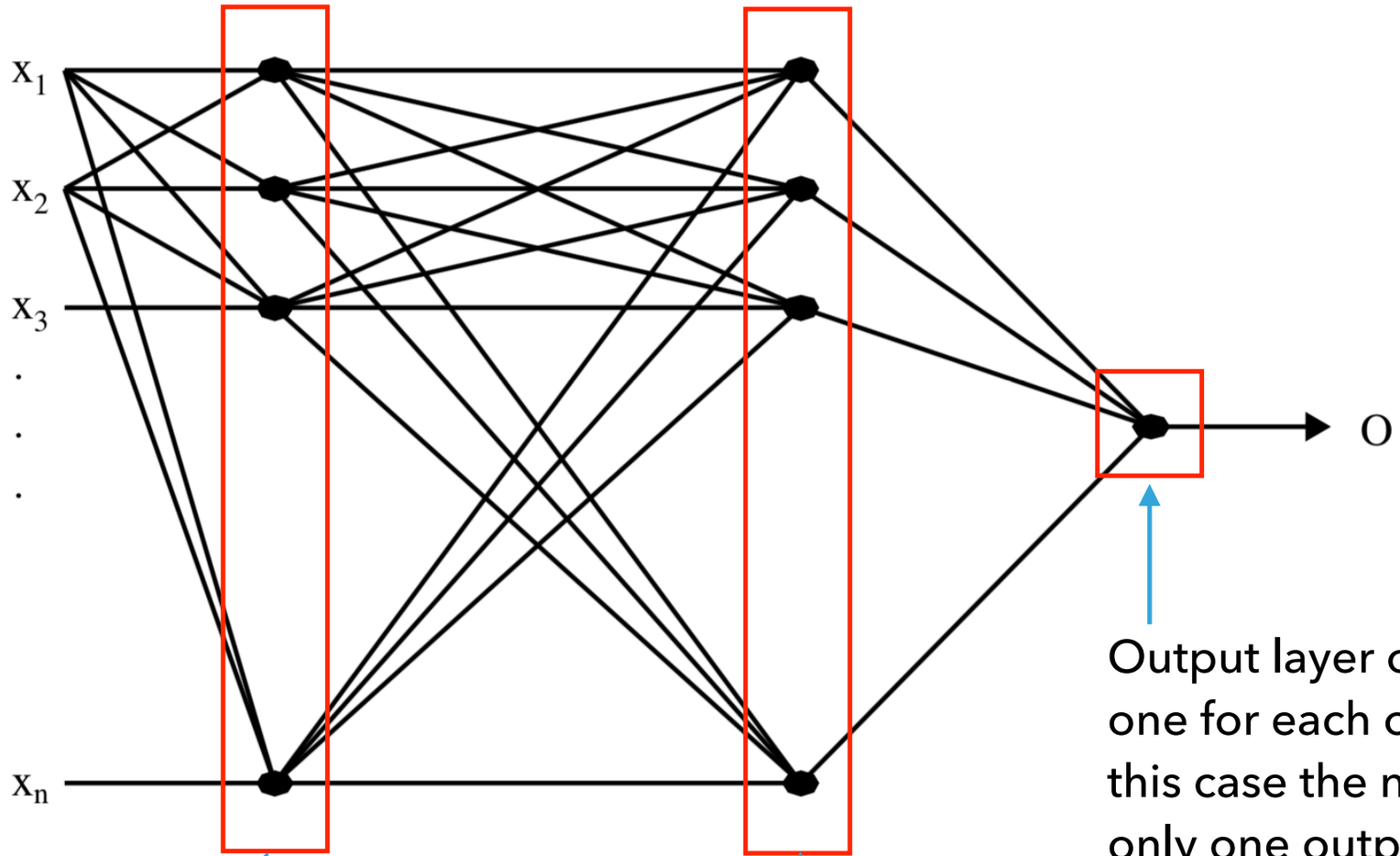


Input layer of n perceptrons; one for each dimension of the input feature space

Hidden layer of some number of perceptrons, M; at least one for each dimension of the input feature space.

MULTILAYER PERCEPTRONS

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



Output layer of perceptrons; one for each output type. In this case the network has only one output.

Input layer of n perceptrons; one for each dimension of the input feature space

Hidden layer of some number of perceptrons, M; at least one for each dimension of the input feature space.

TRAINING

- Parameter tuning is referred to as training
- 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.

LINEAR REGRESSION

- Consider the equation:

$$y = f(x)$$

$$mx + c$$

- This describes a straight line where
  - m is the slope of the line
  - c is the constant offset (y value at x=0).
- The problem is how to select the values of m and c in order to obtain the best possible model of the data.
  - To do this we need to make some assumptions.

* If people wish to read up on the subject of likelihood fitting a good starting point is the book by Edwards entitled Likelihood.

LINEAR REGRESSION

- 1) Assume that the data have a linear relationship so that we can describe the relationship between x and y with this model.
- 2) Define some figure of merit that can be optimised in order to determine the values of the parameters m and c.
- Define some method that can be used in order to extract the optimal values of m and c.
  - In scientific applications we also want to know the uncertainty (or error) on m and c.

* If people wish to read up on the subject of likelihood fitting a good starting point is the book by Edwards entitled Likelihood.

LINEAR REGRESSION

- 1) Assume that the data have a linear relationship so that we can describe the relationship between x and y with this model.

✓ Let's assume that this function is valid for the problem

- 2) Define some figure of merit that can be optimised in order to determine the values of the parameters m and c.

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - \hat{y}}{\sigma_i} \right)^2$$

y_i = y value for i^{th} example

σ_i = error on y value of i^{th} example

$\hat{y}$ = estimate of y given x using the model

χ^2 = Sum over all examples of the normalised squared residual.

* If people wish to read up on the subject of likelihood fitting a good starting point is the book by Edwards entitled Likelihood.

LINEAR REGRESSION

- 1) Assume that the data have a linear relationship so that we can describe the relationship between x and y with this model.

✓ Let's assume that this function is valid for the problem

- 2) Define some figure of merit that can be optimised in order to determine the values of the parameters m and c.

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - (mx + c)}{\sigma_i} \right)^2$$

y_i = y value for i^{th} example

σ_i = error on y value of i^{th} example

m and c are model parameters to be determined

χ^2 = Sum over all examples of the normalised squared residual.

* If people wish to read up on the subject of likelihood fitting a good starting point is the book by Edwards entitled Likelihood.

LINEAR REGRESSION

- 1) Assume that the data have a linear relationship so that we can describe the relationship between x and y with this model.

✓ Let's assume that this function is valid for the problem

- 2) Define some figure of merit that can be optimised in order to determine the values of the parameters m and c.

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - (mx + c)}{\sigma_i} \right)^2$$

y_i = y value for i^{th} example

σ_i = error on y value of i^{th} example

m and c are model parameters to be determined

χ^2 = Sum over all examples of the normalised squared residual.

Note - if we make a simplification that the error is similar for all data points, then we can simplify the problem by neglecting σ_i [effectively we set these values to unity].

* If people wish to read up on the subject of likelihood fitting a good starting point is the book by Edwards entitled Likelihood.

LINEAR REGRESSION

- 3) To move forward we need a set of data examples (N pairs of y and x values) to compute the χ^2 sum.
- We also need to be able to systematically vary m and c to optimise this figure of merit:
- The optimal value of these parameters corresponds to the pair that result in the smallest χ^2 value. This will result in a model that matches the data the best.
- This does not guarantee that the optimal choice of m and c will result in a good model (overfitting/overtraining will be discussed later in the course).

LINEAR REGRESSION

- 3 contd.) We will use a gradient descent parameter optimisation algorithm (See the optimisation lecture notes later in the course).
  - For now you can treat this optimisation process as a black box.
    - Visualise systematically choosing pairs of m and c, and for each point in this 2D space compute X^2. From the ensemble of points in this hyperspace, one can then select the minimum.
    - Algorithmically this is expensive so we use algorithms that approximate the search for the minimum that is computationally more efficient (and adaptable to higher dimensional parameter spaces).
- The analytic solution for this problem is given at the end of these slides.

LINEAR REGRESSION

- Illustration of the optimisation process for a 1D problem.
- Take an ensemble of measurements of some quantity S*

i =	1	2	3	4	5	6	7
S	0.662	0.625	0.897	0.614	0.925	0.694	0.601
σ_S	0.039	0.091	0.100	0.160	0.160	0.061	0.239

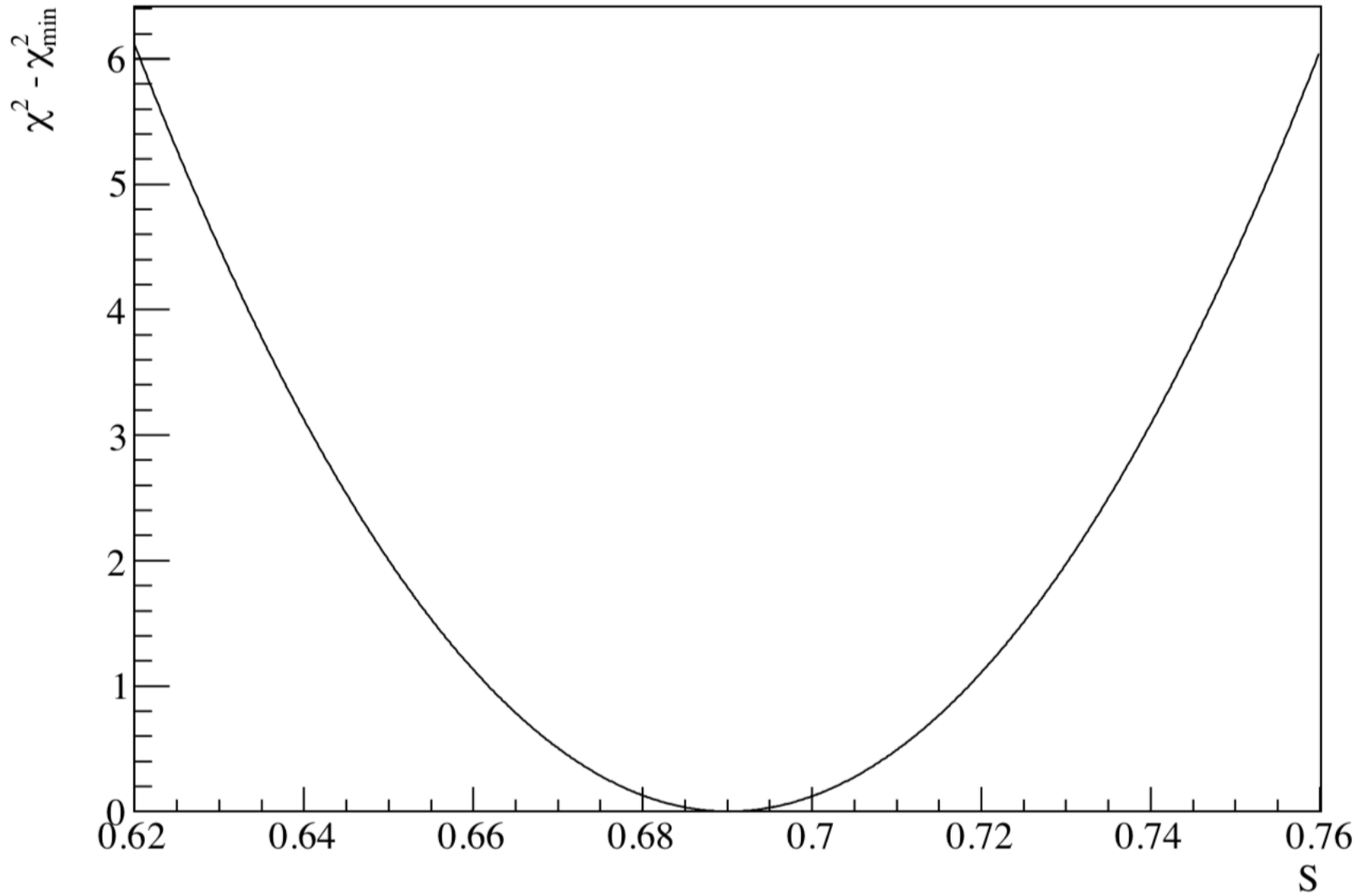
- We want to extract the average value of S from these data, which can be done by scanning through assumed values (over some sensible range) and computing:

$$\chi^2 = \sum_{i=1}^{7} \left(\frac{S_i - S}{\sigma_{S_i}} \right)^2$$

*S is a parameter that is related to matter-antimatter asymmetry in sub-atomic quantum systems. S=sin2β, where β is a manifestation of a phase difference between matter and antimatter decays in certain decays of neutral B mesons. The background behind this measurement is discussed in this Symmetry magazine article and from a technical perspective in this book: The Physics of the B Factories.

LINEAR REGRESSION

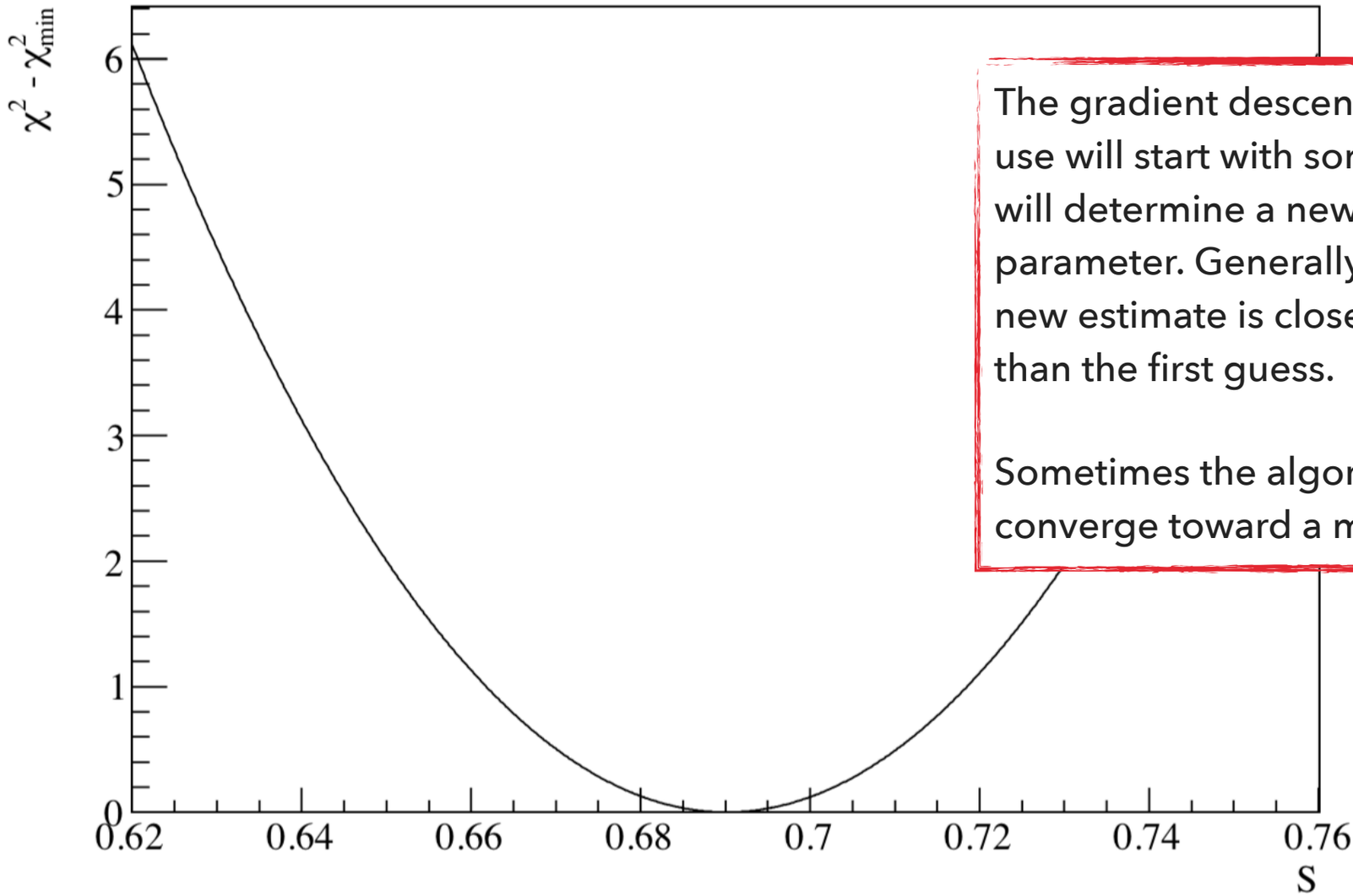
- On doing this we obtain a parabolic curve, where a change in one unit from the minimum corresponds to a change of 1σ (the error) in S.



*S is a parameter that is related to matter-antimatter asymmetry in sub-atomic quantum systems. S=sin2β, where β is a manifestation of a phase difference between matter and antimatter decays in certain decays of neutral B mesons. The background behind this measurement is discussed in this Symmetry magazine article and from a technical perspective in this book: The Physics of the B Factories.

LINEAR REGRESSION

- On doing this we obtain a parabolic curve, where a change in one unit from the minimum corresponds to a change of 1σ (the error) in S.



The gradient descent algorithm we will use will start with some initial value and will determine a new estimate of the parameter. Generally we expect that this new estimate is closer to the minimum than the first guess.

Sometimes the algorithm will fail to converge toward a minimum.

*S is a parameter that is related to matter-antimatter asymmetry in sub-atomic quantum systems. S=sin2β, where β is a manifestation of a phase difference between matter and antimatter decays in certain decays of neutral B mesons. The background behind this measurement is discussed in this Symmetry magazine article and from a technical perspective in this book: The Physics of the B Factories.

LINEAR REGRESSION

- We can read the minimum off of a 1D X^2 scan.
- However for a 2D problem we have to scan through points in a grid, and from that array of results choose the optimal.
  - e.g. see the logarithmic grid search performed by R's libsvm package, which performs a 2D parameter scan.
- For more dimensions than 2 it is computationally expensive to perform a parameter scane, and difficult to visualise this approach.

*S is a parameter that is related to matter-antimatter asymmetry in sub-atomic quantum systems. S=sin2β, where β is a manifestation of a phase difference between matter and antimatter decays in certain decays of neutral B mesons. The background behind this measurement is discussed in this Symmetry magazine article and from a technical perspective in this book: The Physics of the B Factories.

LINEAR REGRESSION

- `Example_LinearRegression.py`

- We need to select some parameters for the optimisation process:

```
learning_rate    = 0.005
training_epochs  = 10
min_x = 0
max_x = 1
Ngen  = 100
gradient  = 2.0
intercept = 0.5
noise     = 0.1  # data are inherently noisy, so we generate noise
```

`learning_rate`:
step size for the optimisation process

`training_epochs`:
number of times the optimisation is run

`Ngen`:
number of simulated examples to use (=N in the X^2 sum)

`gradient` and `intercept`:
the parameters m and c, respectively.

`noise`:
Data are inherently noisy, this parameter introduces an element of randomness to the value of y for a given x.

LINEAR REGRESSION

- `Example_LinearRegression.py`

- Implementing linear regression function for optimisation:

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - (mx + c)}{\sigma_i} \right)^2$$

LINEAR REGRESSION

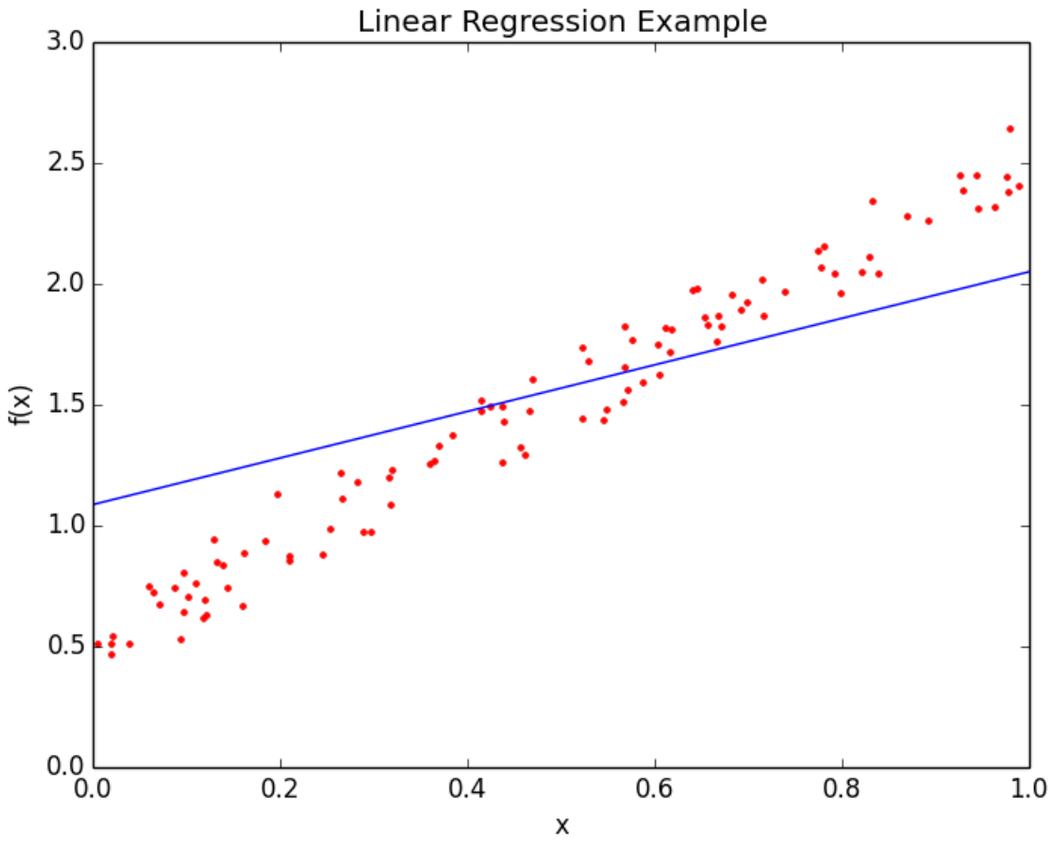
- `Example_LinearRegression.py`

- The output of this script can be seen below:

Input Values:
m = 2.0
c = 0.5

Training Parameters*:
N = 100
N_{Epochs} = 1
Learning rate = 0.005

Output Values:
m = 0.963…
c = 1.087…



Recall that the parameters m and c are initialised using a random number.

The optimisation algorithm will iteratively search through the (m, c) space to determine the optimal set of parameters.

The solution found depends on:
- the starting point
- learning rate
- number of training epochs

* These will be explained in more detail when we discuss optimisation.

LINEAR REGRESSION

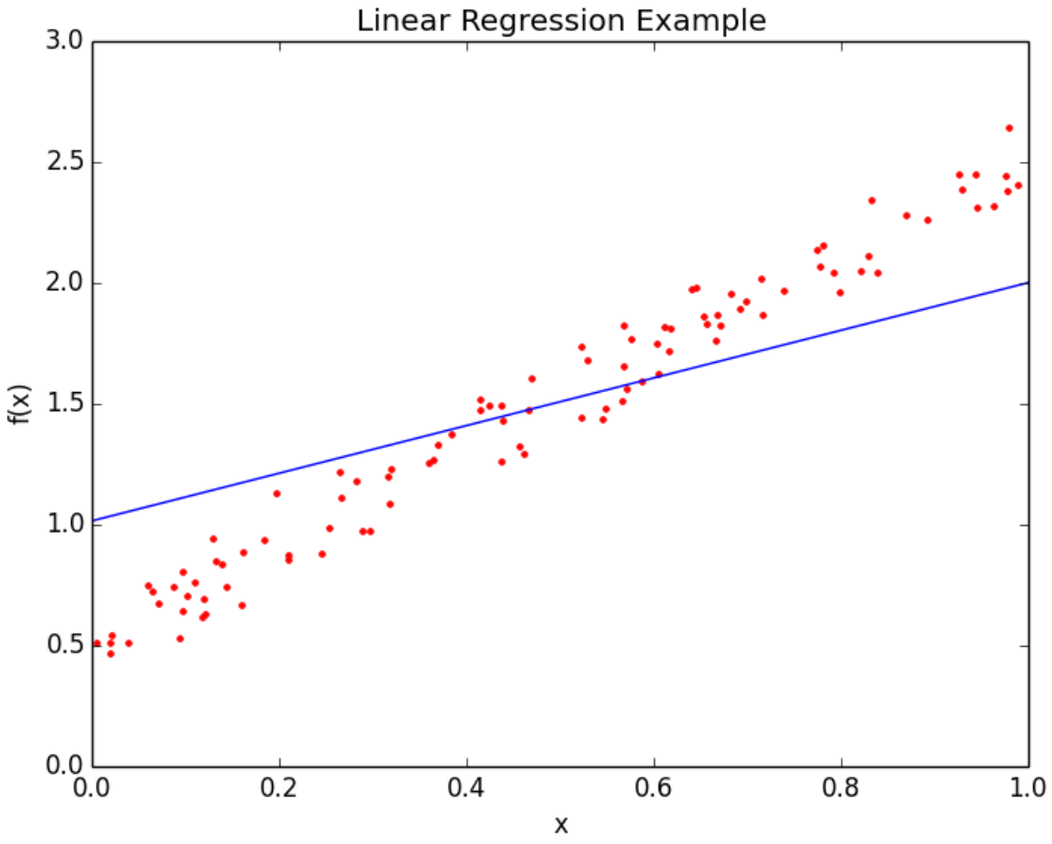
- `Example_LinearRegression.py`

- The output of this script can be seen below:

Input Values:
m = 2.0
c = 0.5

Training Parameters*:
N = 100
N_{Epochs} = 2
Learning rate = 0.005

Output Values:
m = 0.985 …
c = 1.016 …



* These will be explained in more detail when we discuss optimisation.

LINEAR REGRESSION

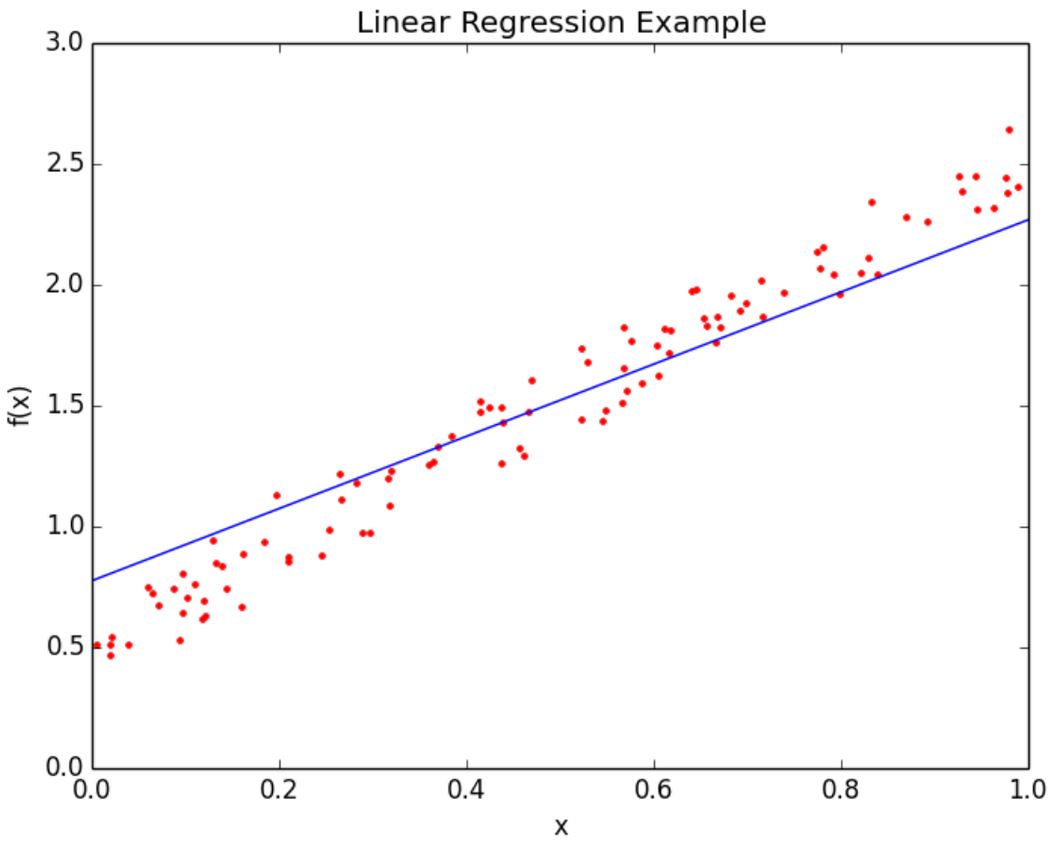
- `Example_LinearRegression.py`

- The output of this script can be seen below:

Input Values:
m = 2.0
c = 0.5

Training Parameters*:
N = 100
N_{Epochs} = 10
Learning rate = 0.005

Output Values:
m = 1.493…
c = 0.776…



* These will be explained in more detail when we discuss optimisation.

LINEAR REGRESSION

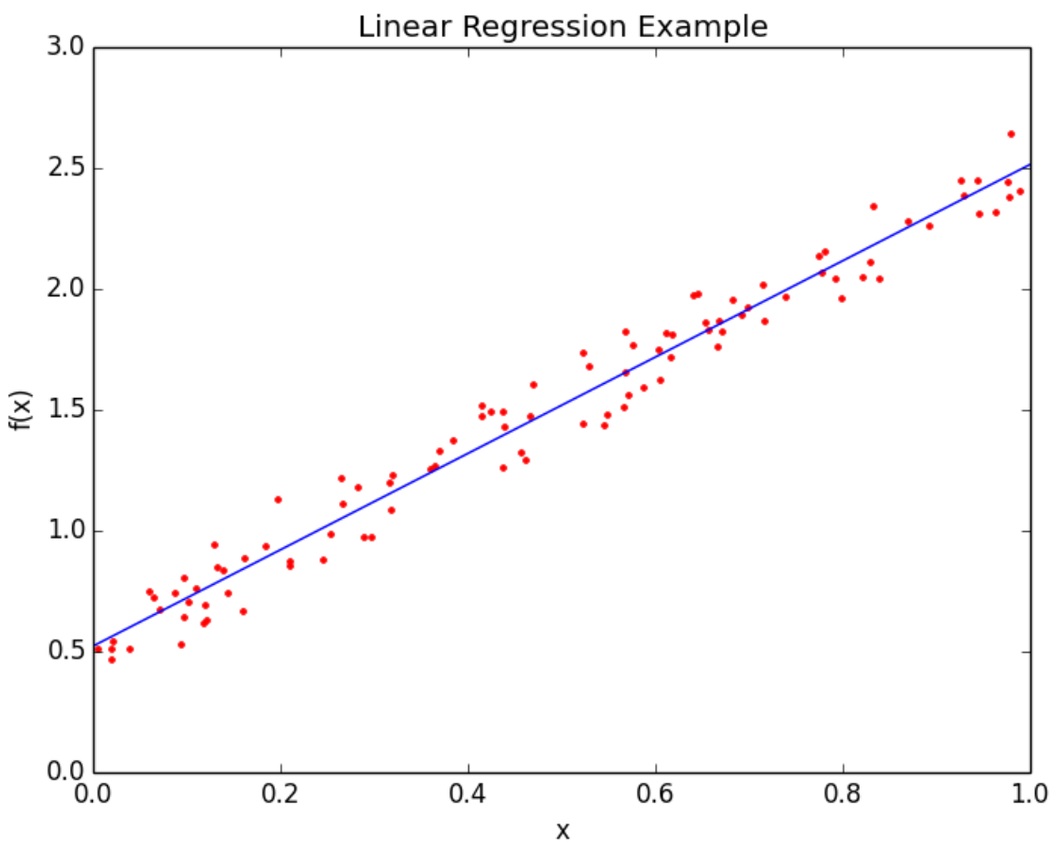
- `Example_LinearRegression.py`

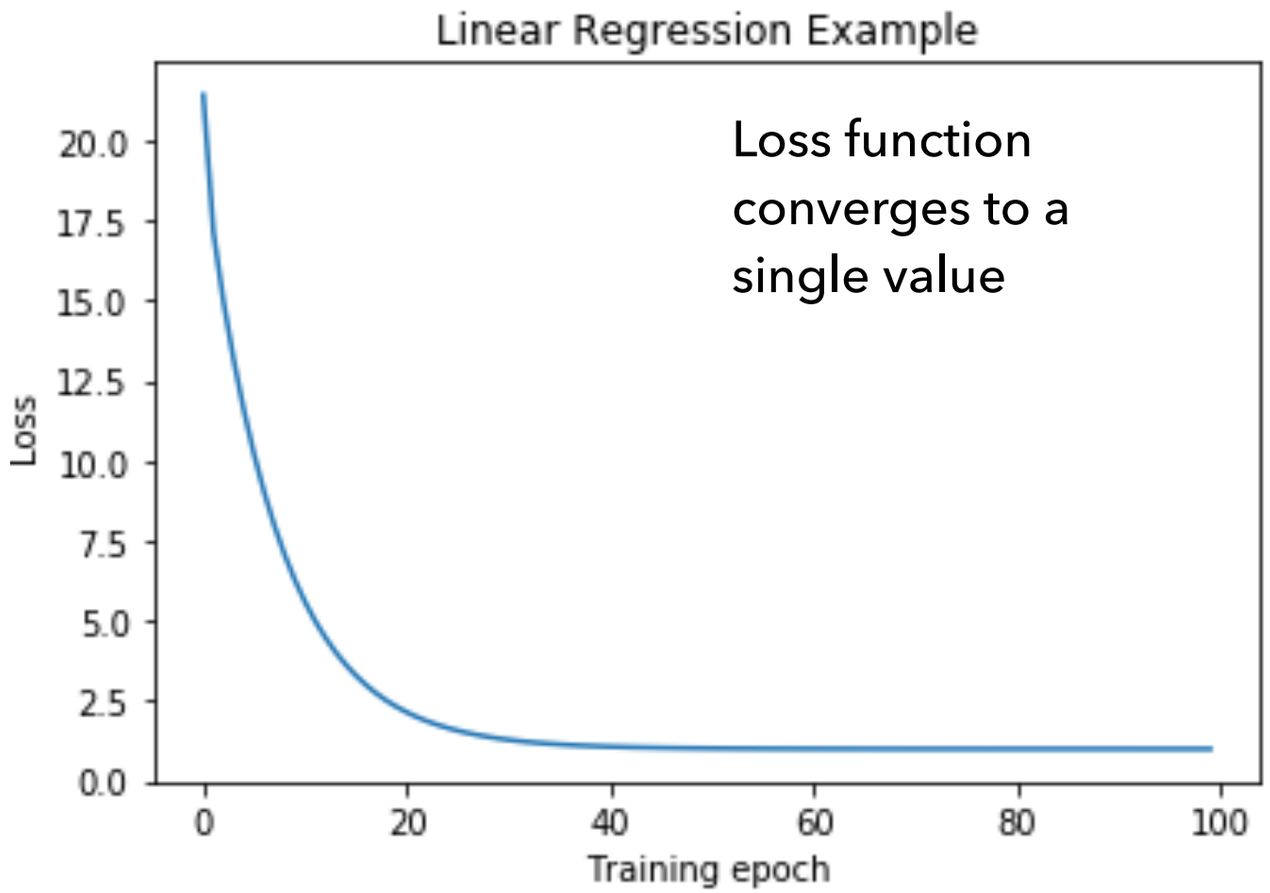
- The output of this script can be seen below:

Input Values:
m = 2.0
c = 0.5

Training Parameters*:
N = 100
N_{Epochs} = 100
Learning rate = 0.005

Output Values:
m = 1.993…
c = 0.523…





* These will be explained in more detail when we discuss optimisation.

LINEAR REGRESSION

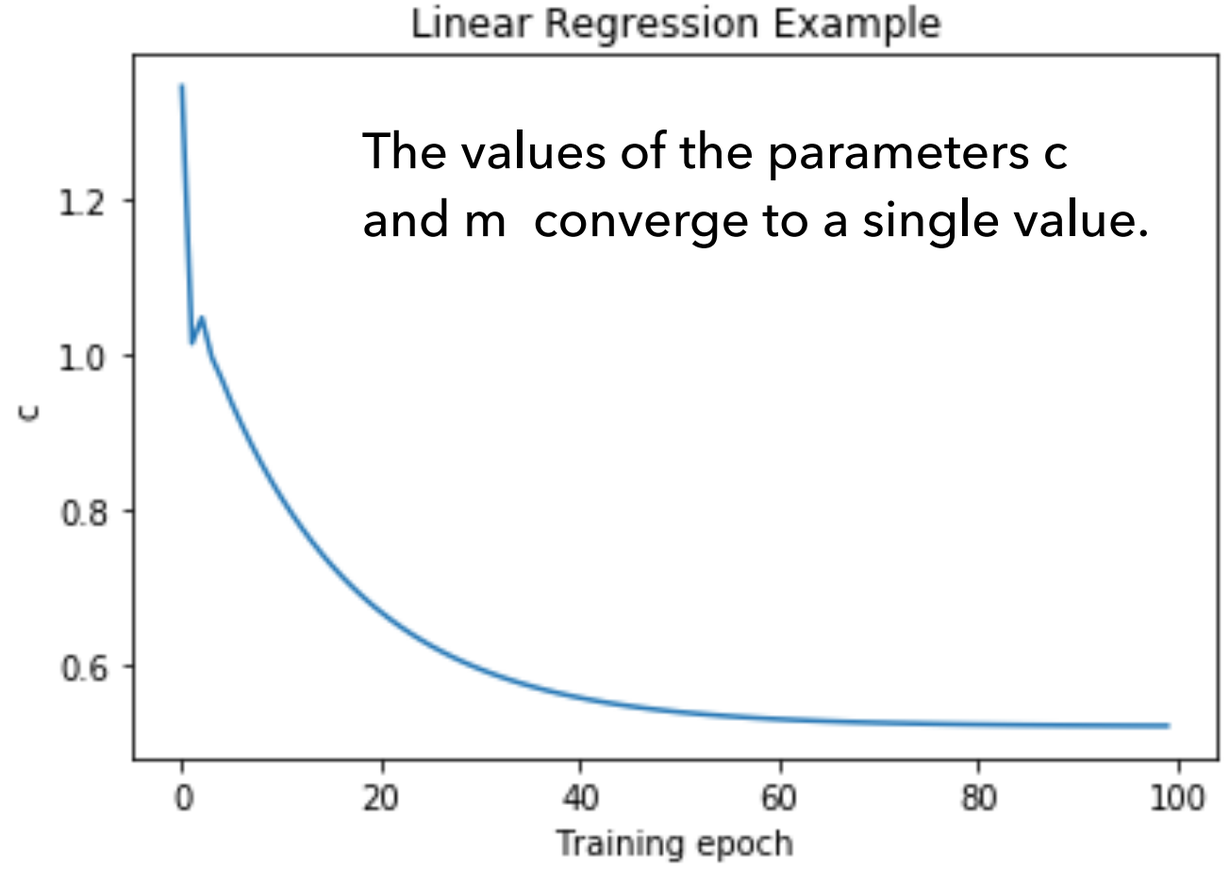
- `Example_LinearRegression.py`

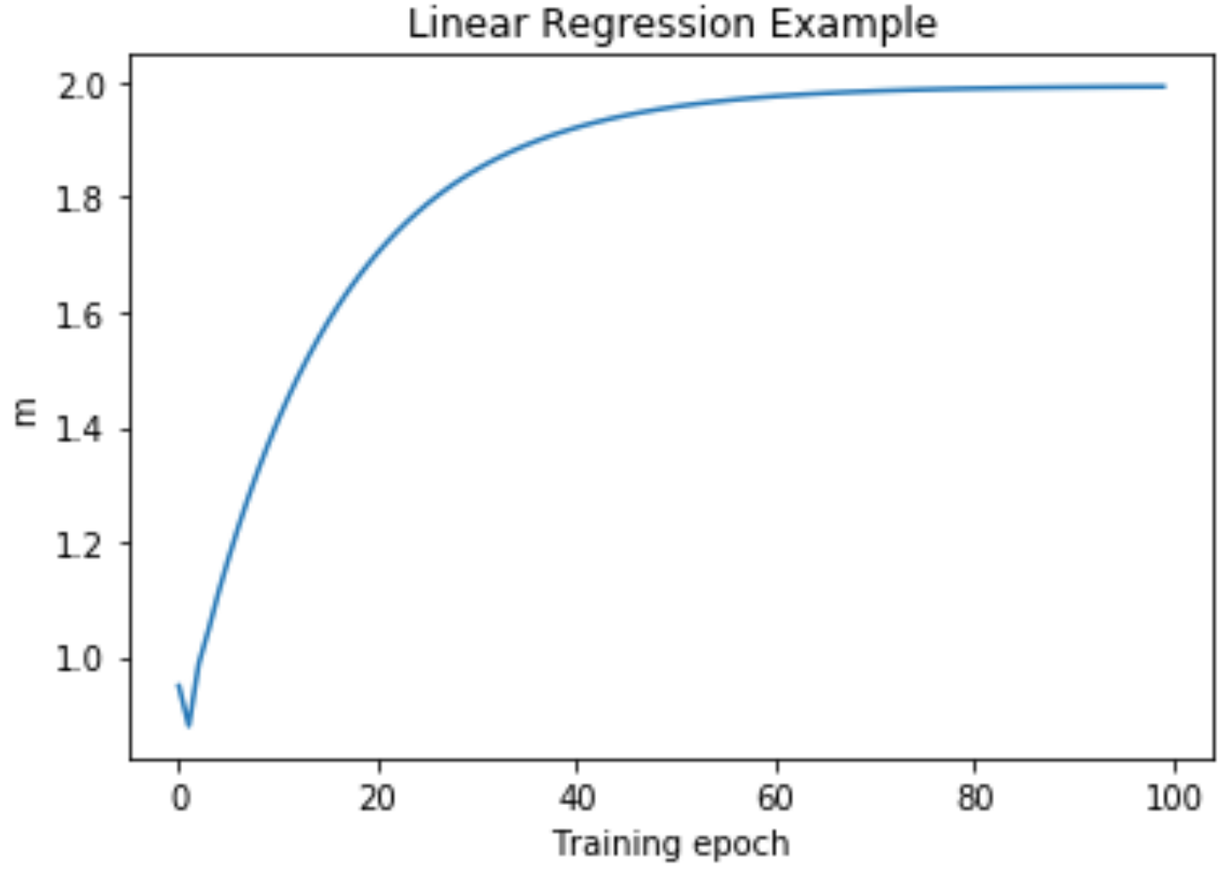
- The output of this script can be seen below:

Input Values:
m = 2.0
c = 0.5

Training Parameters*:
N = 100
N_{Epochs} = 100
Learning rate = 0.005

Output Values:
m = 1.993...
c = 0.523...





* These will be explained in more detail when we discuss optimisation.

LINEAR REGRESSION

- **Summary**:
  - The linear regression problem discussed here uses a loss function that is based on the square residuals of data vs some prediction.
  - This allows us to model a simple linear relationship between some input x and some output y, where the functional form is just:
    - $y = mx+c$
  - where m and c are parameters to determine.
- If we addressed the issue of uncertainties in the data, we could also extract uncertainties on the optimal values of m and c obtained, and this would be fitting.
- This approach and will be generalised when we consider neural networks and extensions to non-linear problems that can not be solved analytically.
  - For machine learning problems we don't care about the uncertainty on the optimal parameters determined(*).

(*) Bayesian networks do allow for a probability distribution for weights, and so this remark is really method dependent. That detail is beyond the scope of this course.

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.

CONACYT FELLOWSHIPS

- CONACyT funding for students to study abroad
- Queen Mary will match funding: https://www.qmul.ac.uk/scholarships/items/conacyt-scholarships.html
- Covers master's degrees and PhDs
  Master's course "AI and ML for Science"
- Advice from previous students on applications: https://www.qmul.ac.uk/scholarships/database/conacyt-scholars/

SCHOLARSHIPS FOR IPN GRADUATES

Additional program for graduates of IPN

Funding to PhD or Masters at Queen Mary, University of London

https://www.ipn.mx/posgrado/convocatorias/convocatorias-internas.html

NOTEBOOK FOR TODAY

https://github.com/abbeywaldron/cinvestav_ML_2024