Several additional lecture note sets are referred to throughout. These have more detail and references to the original works in the literature.

# State of the art ML tools and methods

*Belle II Physics Week, 10th October 2025*


## Prof. Adrian Bevan

a.j.bevan@qmul.ac.uk

# Opening remarks

- I've been using AI and ML in physics analyses since 2002, and this is my perspective on the state of the art in ML relevant for Belle II
- Given graduate lectures on the subject since 2007
- Incorporated this into the undergraduate curriculum at QMUL since 2009
- Developed PGT and undergraduate AI programmes
- I still have a lot to learn about the field, and it is impossible to keep up with the pace of change.
  - So don't feel the need to learn and understand everything - with our day job we can't…
- Physicists have an advantage over most CS experts because it is not good enough for something to work, we have to understand why it works, quantify the error on performance, and question the data and the output.

# Opening remarks

- AI has been around for decades - this is not new
- We live in an AI industrial revolution, with an unprecedented level of change in what is state-of-the-art, this is an introductory snapshot
- Traditional AI methods evolved into to deep learning, and now generative AI
- The explainability and interpretability of methods is limited, & an active research area
- No algorithm is a substitute for *you* being smart
- Most of the work is done in understanding and preparing your data: Also called “Data Wrangling”, before you start to use your algorithm
- Modern algorithms are resource-intensive; think about your impact on the environment when designing and using algorithms

# Overview

- The Process (Big Picture)
- Algorithms
- Data Wrangling
- The Optimisation Problem And What It Means
- Tools and Methods
- Resources
- Closing Remarks

# The Process (Big Picture)

- **Broadly speaking we follow a sequential process**

| |
|---|
| Select data, identifying features of interest |
| Select algorithm, optimisation metric, optimisation algorithm, convergence stopping criteria |
| Train, Test, Validate, Cross-Validation, … |
| Apply |
| Publish |

**Algorithms can use supervised or unsupervised learning**

**Particle physics tends to use the former: with control data or Monte Carlo simulated data samples used to develop models, which are then applied to an analysis**

**Unsupervised algorithms learn directly from the data (e.g. K-nearest neighbours, KNN), which we won't touch upon here.**

**e.g. see the Elements of Statistical Learning by Hastie, Tibshirani and Friedman for an overview.**

# The Process (Big Picture)

- **Broadly speaking we follow a sequential process**

| |
|---|
| Select data, identifying features of interest |
| Select algorithm, optimisation metric, optimisation algorithm, convergence stopping criteria |
| Train, Test, Validate, Cross-Validation, … |
| Apply |
| Publish |

- **Feature = input variable, e.g.** $|p|, p_x, p_y, p_z, E, \ldots$
- **i.e. some measurement or reconstructed quantity of a particle or event level quantity**
- **A detector "image" is a set of features (each segment is a feature)**
- **Can combine images and features in a multimodal analysis**
- **Generative AI uses natural language processing; so not limited to numerical data**

# The Process (Big Picture)

- **Broadly speaking we follow a sequential process**

| |
|---|
| Select data, identifying features of interest |
| Select algorithm, optimisation metric, optimisation algorithm, convergence stopping criteria |
| Train, Test, Validate, Cross-Validation, … |
| Apply |
| Publish |

- Lots of algorithms to choose from
- The optimisation metric is called the loss function or objective function in the literature, e.g. $L_2$ norm, softmax (likelihood inspired), or some custom loss
- Fitting in particle physics relies on a robust, but old variable metric minimisation algorithm (use the inverse of the Hessian to find the optimal solution)
- Modern AI uses faster optimisation tools: Adam Optimiser is the current preferred optimiser

# The Process (Big Picture)

- **Broadly speaking we follow a sequential process**

| |
|---|
| Select data, identifying features of interest |
| Select algorithm, optimisation metric, optimisation algorithm, convergence stopping criteria |
| Train, Test, Validate, Cross-Validation, … |
| Apply |
| Publish |

- Is the algorithm tuned to statistical fluctuations in the data?
- Using independent samples to that are used for training to verify performance builds confidence in the results
- Systematic errors are not considered, and uncertainty quantification is an active area of AI research
- Cross-validation can give a sense of the variation in performance resulting from statistical variations in the underlying data sampling

# The Process (Big Picture)

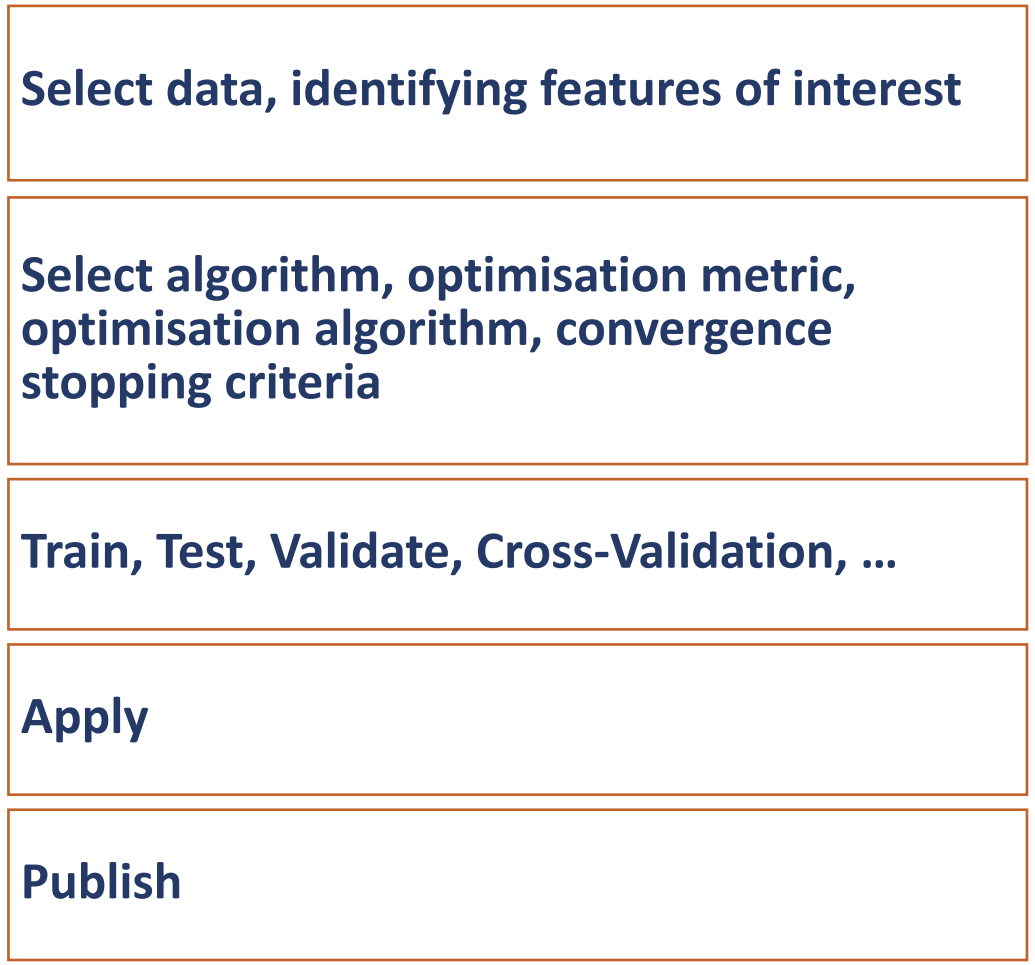
- Broadly speaking we follow a sequential process

| |
|---|
| Select data, identifying features of interest |
| Select algorithm, optimisation metric, optimisation algorithm, convergence stopping criteria |
| Train, Test, Validate, Cross-Validation, … |
| Apply |
| Publish |

- The final step is processing your data based on your understanding of the model that you've built.
- How confident are you that the model performs as expected on your real data, compared with the test, training, and validation samples?
- Are there systematic effects on the input features that you need to explore?

# The Process (Big Picture)

- **Broadly speaking we follow a sequential process**




- **Congratulations on publishing your work, but did you describe the algorithm used? How it was trained? Did you think about if the model should become part of the HEPdata or other public release of information?**
- **Without details of the algorithm, there is little chance of someone being able to replicate the analysis with an independent sample of data.**
- **Not publishing the model closes off the prospect for transfer learning to be used in the community for similar analysis challenges.**

# Algorithms

- I think of AI algorithms as broadly falling into a few useful categories:
  - "Vanilla" AI:
    - Up to the 2010's, built around hyperplanes: $y = w \,.\, x + \beta$
    - Bayesian networks: using causal data to build a model (sometimes referred to as Smart Learning to counter the deep learning trend).
  - Deep learning:
    - Extending MLPs to wide or deep structures, with lots of data and an extended numbers of training epochs
    - Convolutional Neural Networks - inspired by the visual cortex of cats
    - Expansions into intricate structures, starting with models like AlexNet and building on that
    - NLP algorithms are historically simple and adapted to deep learning applications
  - Generative AI:
    - Transformer networks

See my notes on linear discriminants for more details

# Algorithms: Vanilla ($y = w \,.\, x + \beta$)

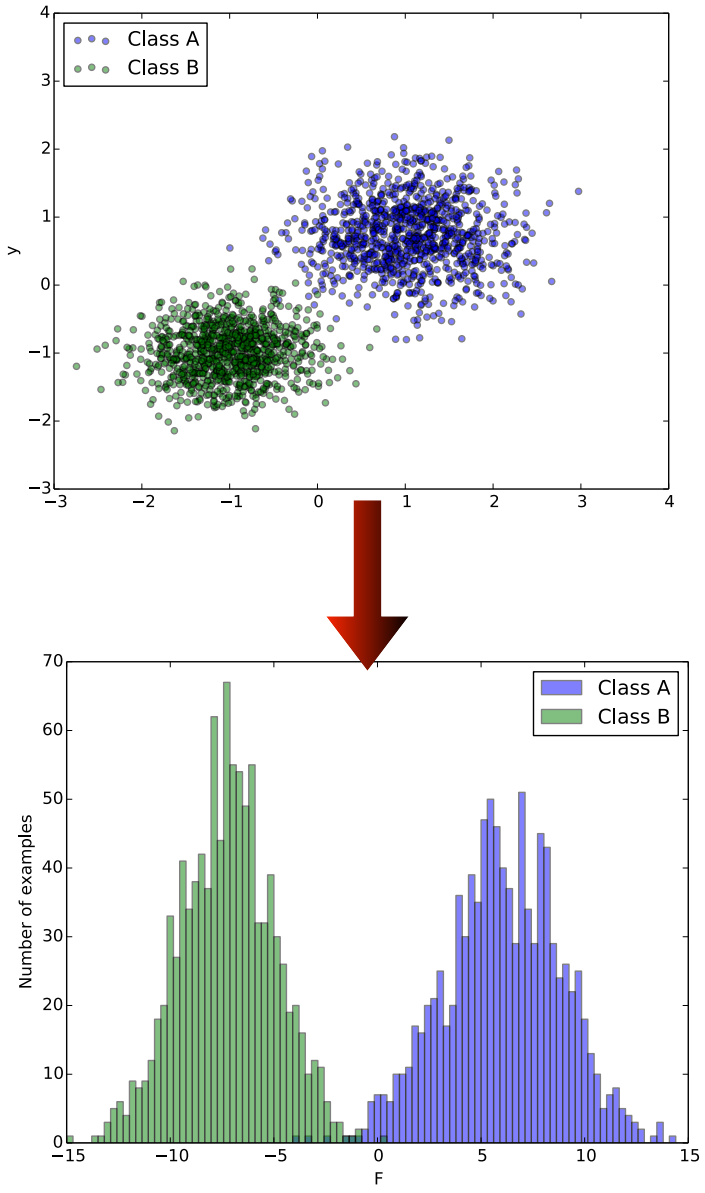
- Linear discriminants like the Fisher discriminant:
  - Align the data along a common axis to reduce dimensionality to 1D, used by CLEO and the B Factories to cut on data or fit data to suppress $q\bar{q}$ ($q = u, d, s, c$) background from $B\overline{B}$ events
  - Straightforward algorithm with coefficients computed analytically

$$F = \alpha^T x + \beta$$

$$\alpha \propto W^{-1}(\underline{\mu}_A - \underline{\mu}_B)$$

$\beta$ is your choice

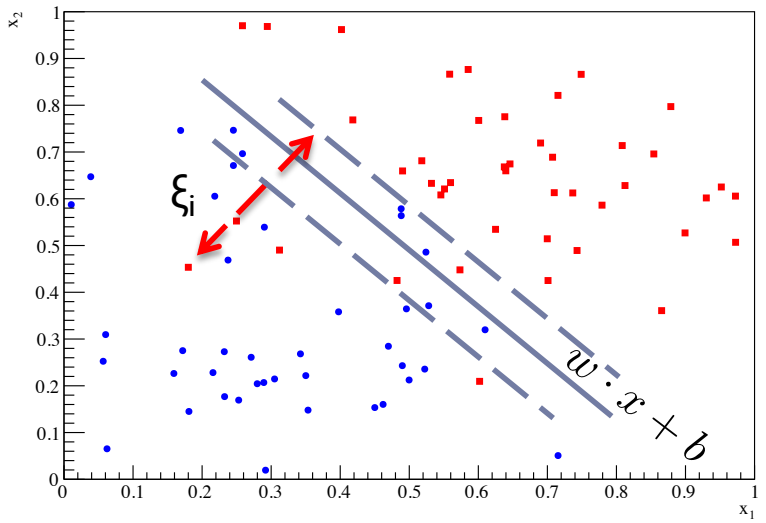
$W$ computed from the sum of covariances



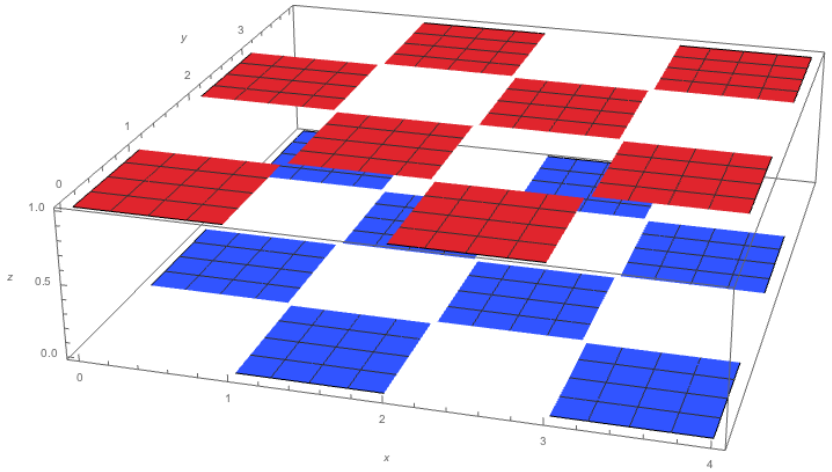
See my notes on SVMs for details.

# Algorithms: Vanilla ($y = w \,.\, x + \beta$)

- Support Vector Machines:
  - Use the equation of the hyperplane to cut the data into categories of signal vs noise using a hard or soft margin for error
  - Choice of Kernal function and how data are prepared affects optimisation
  - Not good for large data samples; curse of dimensionality limits application
  - Implicitly map from the input feature space to an infinite-dimensional dual space to solve the separation problem.




e.g. 4x4 checker board example

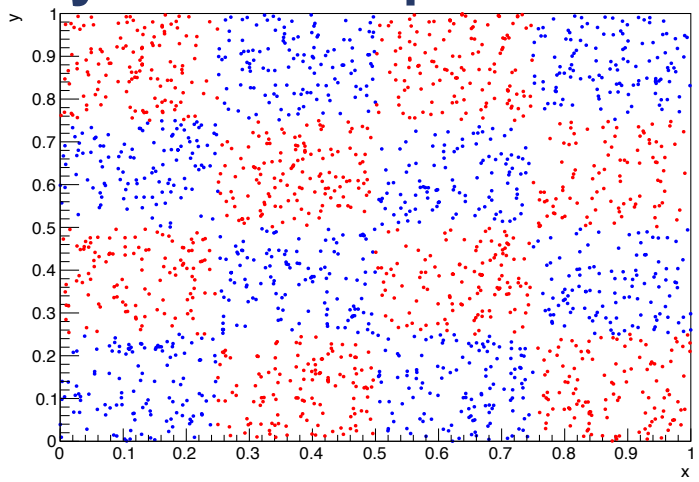


See my notes on SVMs for details.

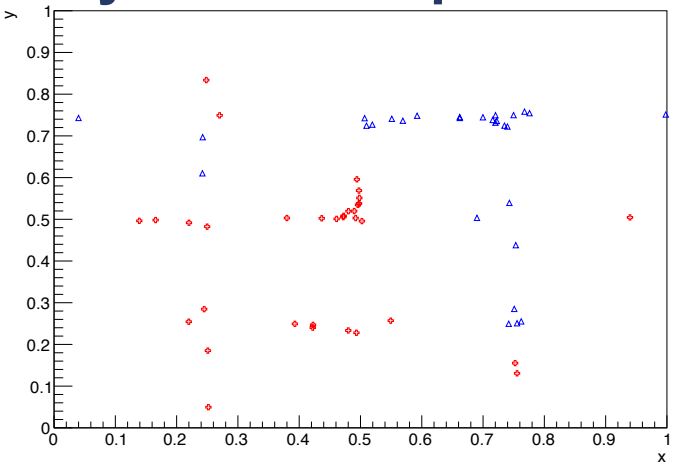
# Algorithms: Vanilla ($y = w \,.\, x + \beta$)

- Support Vector Machines:
  - Use the equation of the hyperplane to cut the data into categories of signal vs noise using a hard or soft margin for error
  - Choice of Kernal function and how data are prepared affects optimisation
  - Not good for large data samples; curse of dimensionality limits application
  - Implicitly map from the input feature space to an infinite-dimensional dual space to solve the separation problem.

Correctly classified points



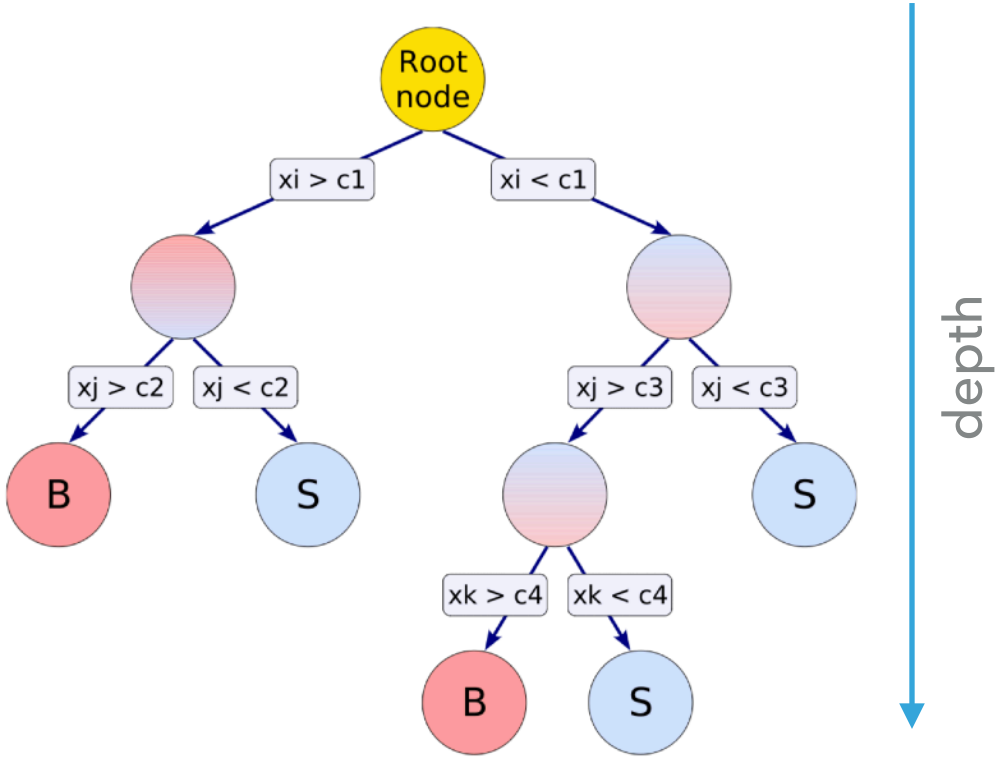

Incorrectly classified points



See my notes on BDTs for details.

# Algorithms: Vanilla ($y = w \,.\, x + \beta$)

- Decision Trees, and related algorithms:
  - Use the equation of the hyperplane to cut the data into categories of signal vs noise to create a weak learner
  - Variants result in the creation of ensembles of weak learners
  - Can use “boosting” to encorage the algorithm to correct its mistakes
  - Premise is that an ensemble of weak learners is a strong learner
  - Algorithms used as based on trial and error - what works well in CS literature on problems is taken as being good for other problems.

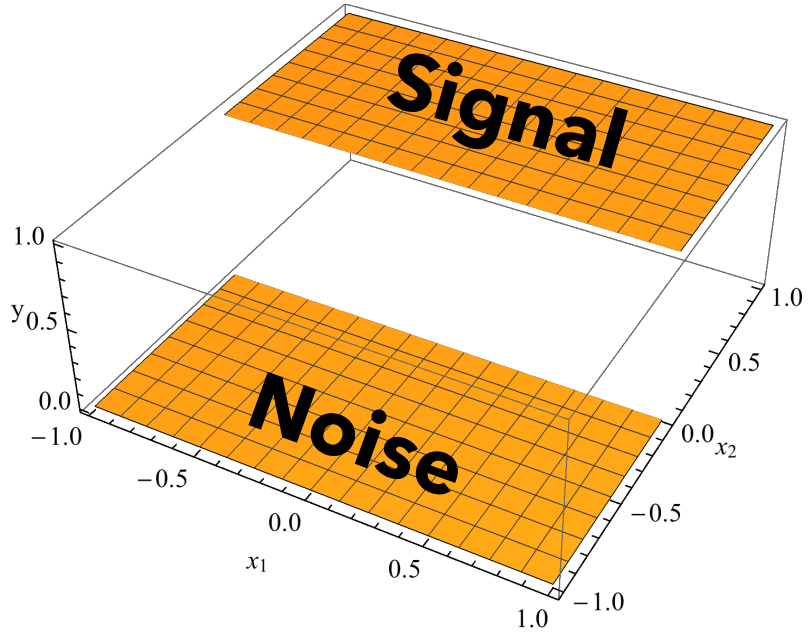


See my notes on NNs for more details

# Algorithms: Vanilla ($y = w \,.\, x + \beta$)

- Neural Networks:
- Use the equation of the hyperplane as an argument to functions that divide up the data into signal and noise like regions, or a measure of where an example sits on that spectrum
  - Single neurons are built around activation functions: $y = f(w \,.\, x + \beta)$
  - Networks are built from layers of neurons
  - Multilayer perceptrons are not easy to interpret or understand

e.g. heavy side function
(binary activation function)




If $w_1 x_1 + w_2 x_2 > \beta$

Output = 1

else

Output = 0

$w_1 = 0$

$w_2 = 1$

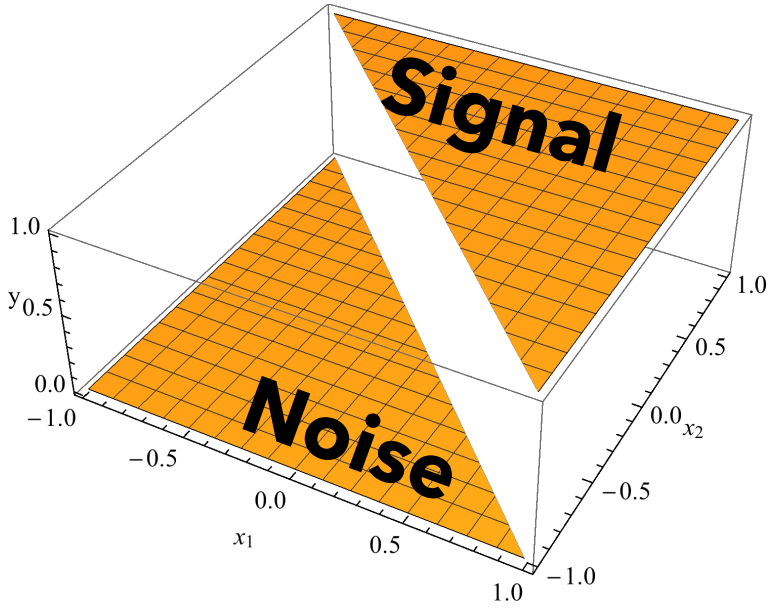
$\beta = 0$

See my notes on NNs for more details

# Algorithms: Vanilla ($y = w \,.\, x + \beta$)

- Neural Networks:
- Use the equation of the hyperplane as an argument to functions that divide up the data into signal and noise like regions, or a measure of where an example sits on that spectrum
  - Single neurons are built around activation functions: $y = f(w \,.\, x + \beta)$
  - Networks are built from layers of neurons
  - Multilayer perceptrons are not easy to interpret or understand

e.g. heavy side function
(binary activation function)




If $w_1 x_1 + w_2 x_2 > \beta$

Output = 1

else

Output = 0

$w_1$ = 1

$w_2$ = 1

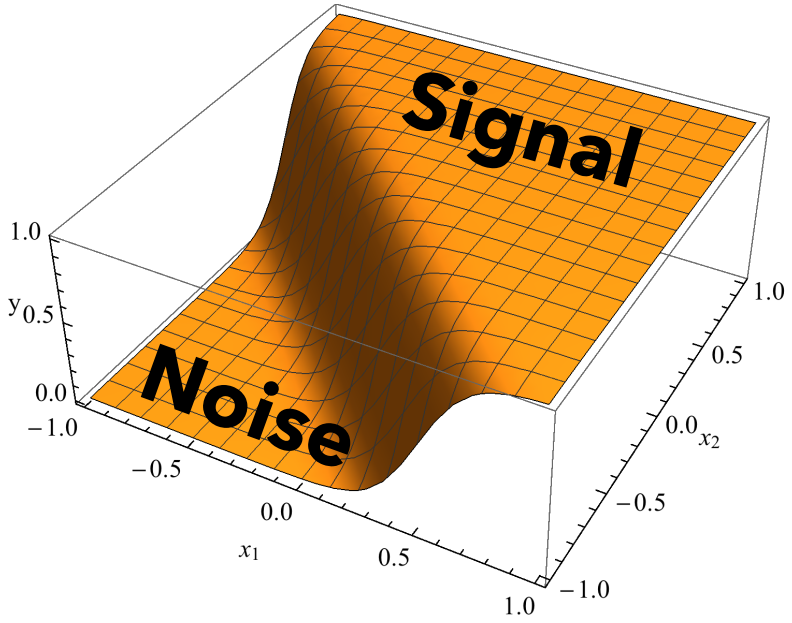
$\beta$ = 0

See my notes on NNs for more details

# Algorithms: Vanilla ($y = w \,.\, x + \beta$)

- Neural Networks:
- Use the equation of the hyperplane as an argument to functions that divide up the data into signal and noise like regions, or a measure of where an example sits on that spectrum
  - Single neurons are built around activation functions: $y = f(w \,.\, x + \beta)$
  - Networks are built from layers of neurons
  - Multilayer perceptrons are not easy to interpret or understand

e.g. logistic (sigmoid) activation function




$$y = \frac{1}{1 + e^{w_1 x_1 + w_2 x_2 + \beta)}}$$

$w_1$ = 10

$w_2$ = 10

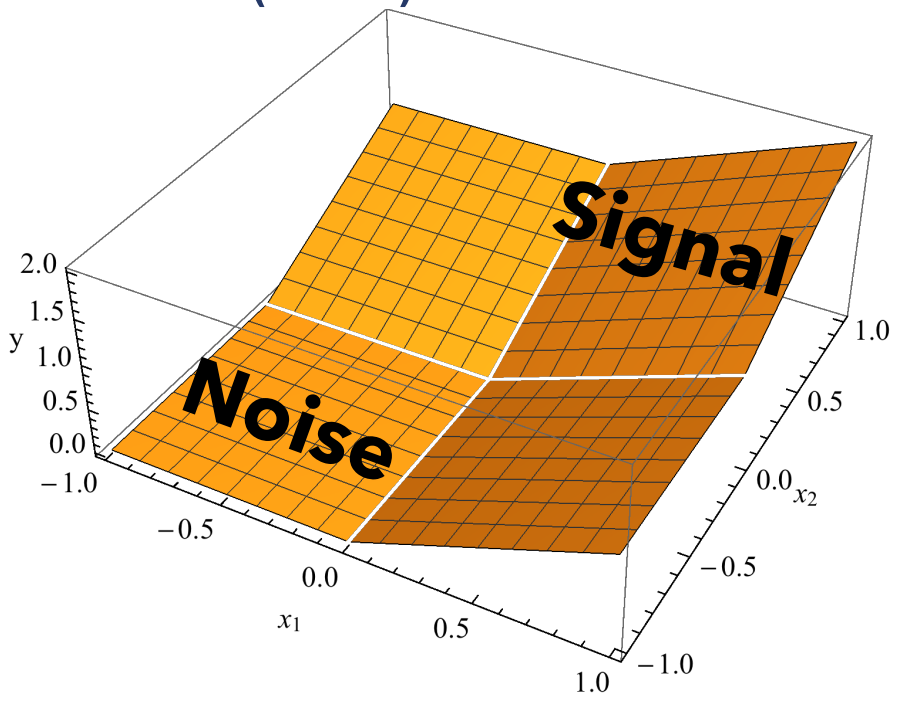
$\beta$ = -5

See my notes on NNs for more details

# Algorithms: Vanilla ($y = w \,.\, x + \beta$)

- Neural Networks:
- Use the equation of the hyperplane as an argument to functions that divide up the data into signal and noise like regions, or a measure of where an example sits on that spectrum
  - Single neurons are built around activation functions: $y = f(w \,.\, x + \beta)$
  - Networks are built from layers of neurons
  - Multilayer perceptrons are not easy to interpret or understand

e.g. Rectified Linear Unit (ReLU) activation function



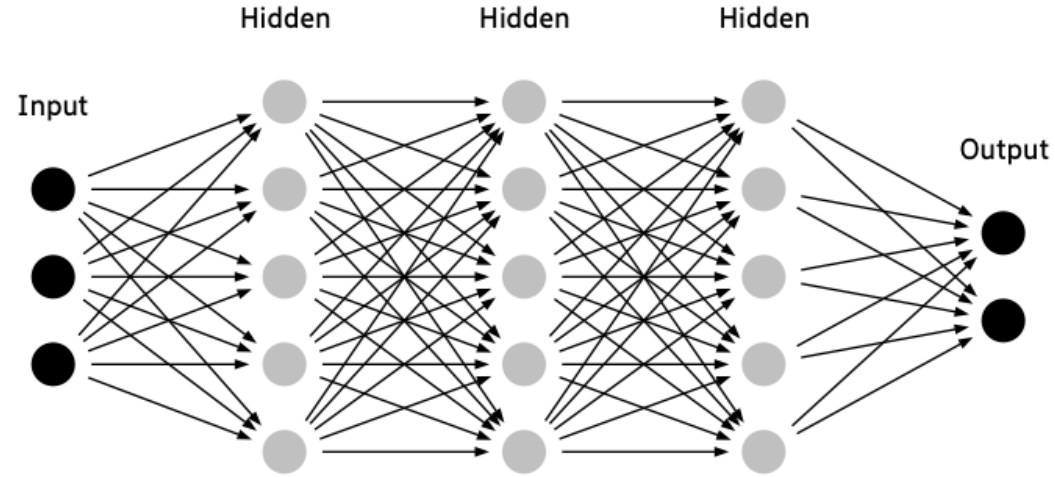

If $x < 0, y = 0$
Else $x = wy$

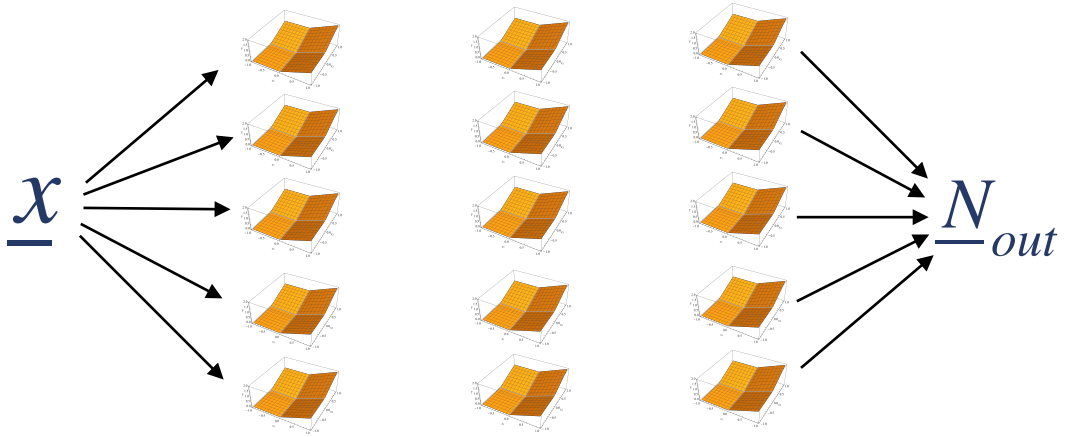
$w_1 = 1$
$w_2 = 1$
$\beta = 0$

See my notes on NNs for more details

# Algorithms: Vanilla ($y = w \,.\, x + \beta$)

- Neural Networks:
- Use the equation of the hyperplane as an argument to functions that divide up the data into signal and noise like regions, or a measure of where an example sits on that spectrum
  - Single neurons are built around activation functions: $y = f(w \,.\, x + \beta)$
  - Networks are built from layers of neurons
  - Multilayer perceptrons are not easy to interpret or understand



See my notes on DL for details

# Algorithms: Deep Learning

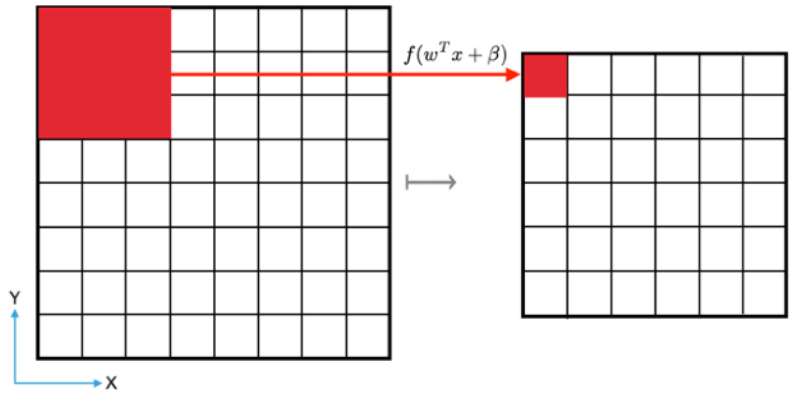
- Deep Networks (MLP style)
  - No concrete definition of a deep network structure in the CS literature.
  - Can be a wide network or a narrow one with many layers
  - Can have millions of weights to learn using the optimisation algorithm
  - Critical to have enough data to learn the parameters of the model
    - If you have <1,000 events, probably best to use an SVM (don't use SVMs for much larger samples)
    - If you have <50,000 events then a BDT normally works well
    - If you have 100,000s events then a deep network can work well*

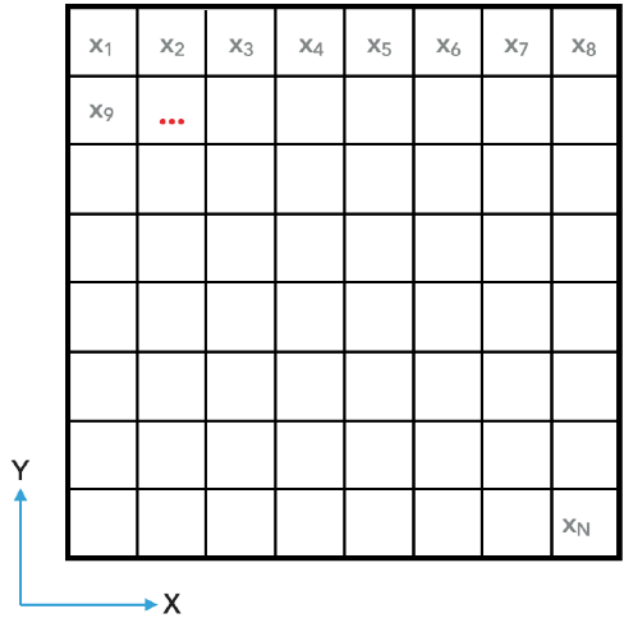
*caveat, image processing using convolutional neural networks (see later) can be done well with lower numbers of images than this

See my notes on DL for details

# Algorithms: Deep Learning

- Convolutional Networks
  - Inspired by the visual cortex of cats[1]
  - Designed for image processing
  - Uses $y = w \,.\, x + \beta$ as a numerical convolution




- Map the image colour values to [0, 1]
- Greyscale is a 2D set of numbers
- Colour is a 3D set of numbers (3rd dimensions I r, g, b)
- Define a convolution filter size (e.g. 3x3)
- Define a step size (e.g. 1 pixel step) referred to as stride
- Specify how many convolution filters to learn
- Optimise the filter weights

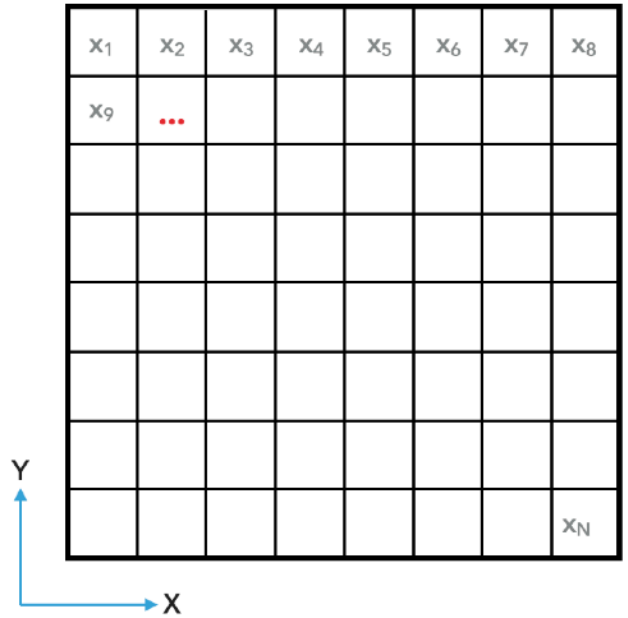


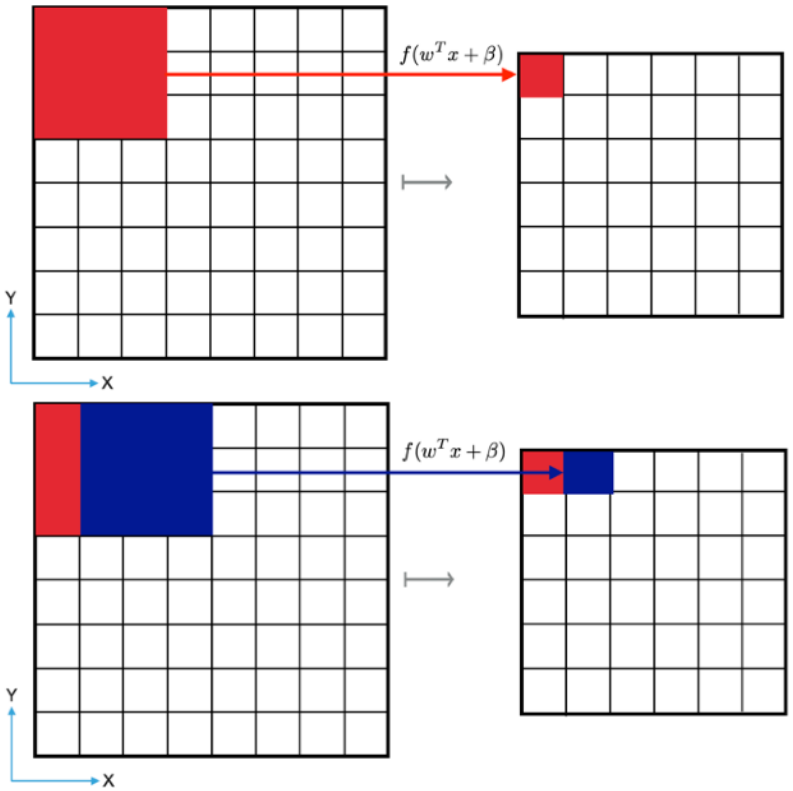

[1] K Fukushima, Bio. Cybernetics **36** p193-202, 1980.

See my notes on DL for details

# Algorithms: Deep Learning

- Convolutional Networks
  - Inspired by the visual cortex of cats[1]
  - Designed for image processing
  - Uses $y = w \,.\, x + \beta$ as a numerical convolution




- The new image generated is smaller than the original, unless …
- Can pad the original image with zeros before learning

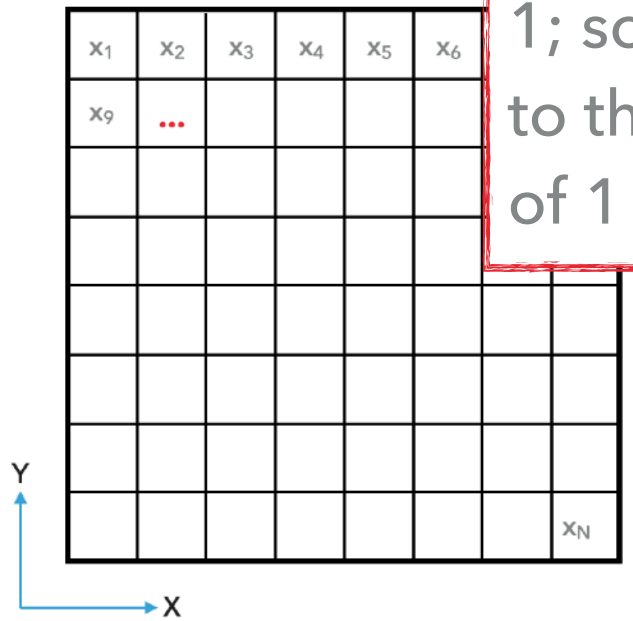


[1] K Fukushima, Bio. Cybernetics **36** p193-202, 1980.

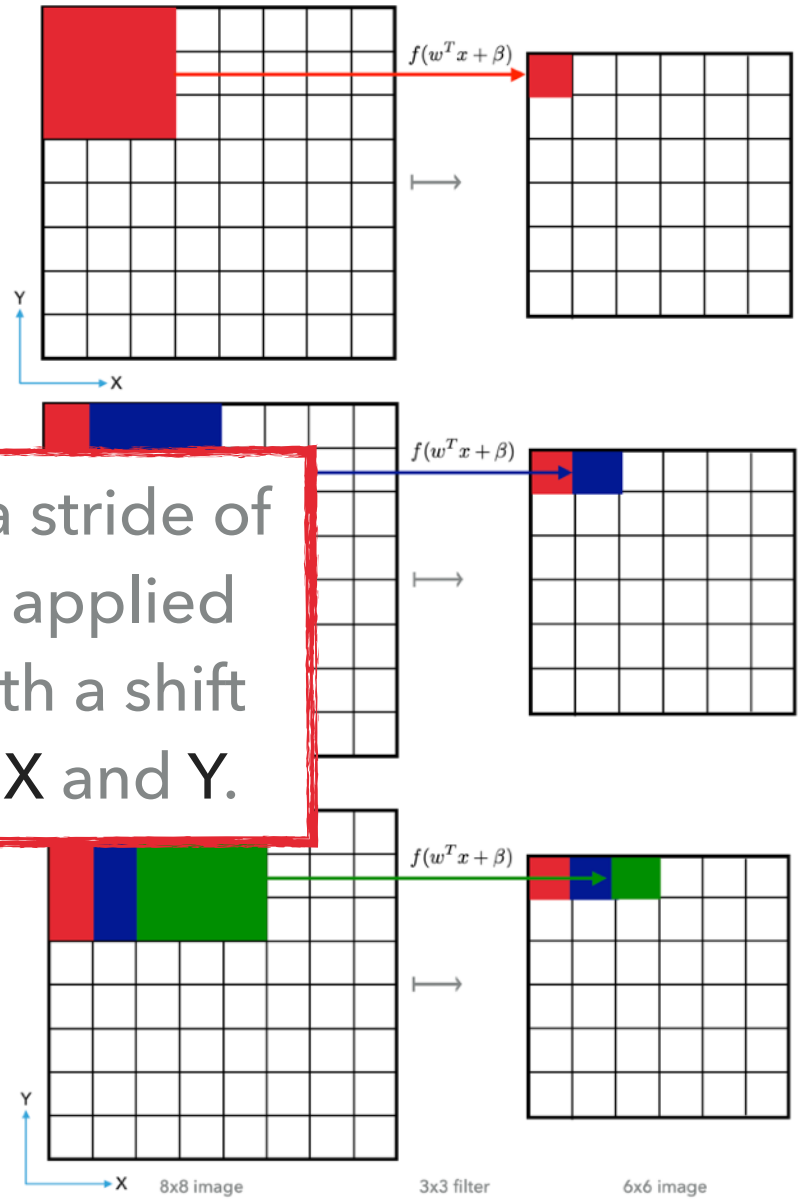
See my notes on DL for details

# Algorithms: Deep Learning

- **Convolutional Networks**
  - **Inspired by the visual cortex of cats[1]**
  - **Designed for image processing**
  - **Uses $y = w \,.\, x + \beta$ as a numerical convolution**




This illustration uses a stride of 1; so the conv filter is applied to the input image with a shift of 1 pixel at a time in X and Y.



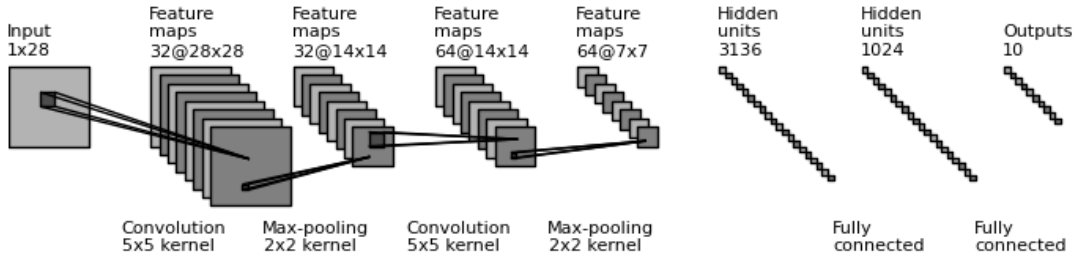



[1] K Fukushima, Bio. Cybernetics **36** p193-202, 1980.

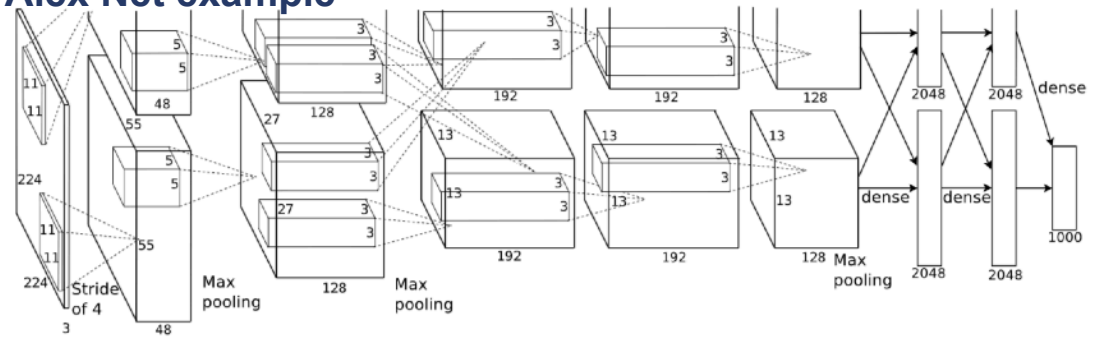
See my notes on DL for details

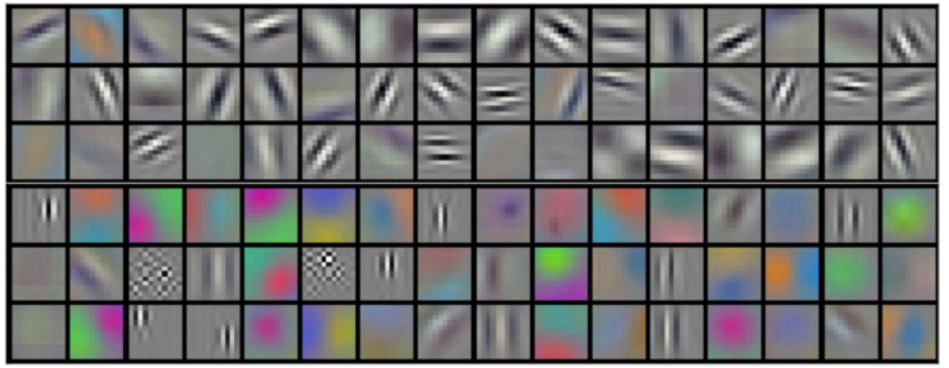
# Algorithms: Deep Learning

- Convolutional Networks
  - Typical structure has convolution layers
  - Pooling layers are a downsampling of the image size (e.g. blurring)
  - Often pass from image based layer structures to fully connected MLP-like structures in the network
  - One or more outputs as a logits or one hot vector
- Lots of other kinds of model developed and tested against benchmark data sets in the literature
- Convolution filters tune on shape and colour patterns




Alex Net example





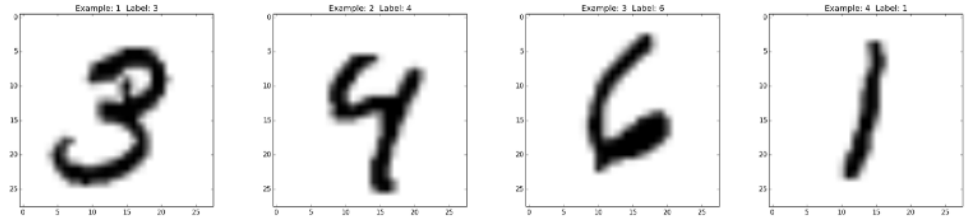
Krizhevsky et al., Neural Information Processing Systems conference proceedings.

See my notes on DL for details

# Algorithms: Deep Learning

- Lots of benchmark data, for example MNIST (handwritten numbers)
- CFAR (different categories of object)
- Etc.

MNIST examples



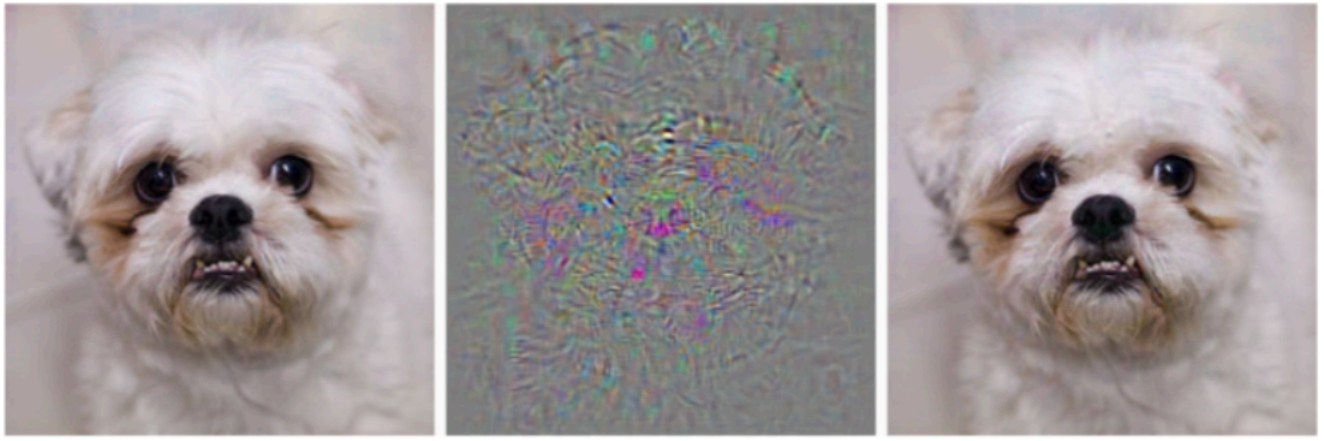

For data to help develop an understanding of these algorithms, see for example

- UC Irvine repository of data: https://archive.ics.uci.edu/datasets
- Kaggle: https://www.kaggle.com/datasets
- MNIST: https://www.kaggle.com/datasets/hojjatk/mnist-dataset/data (standard data set found in TensorFlow and other packages)
- CFAR 10/100: https://www.cs.toronto.edu/~kriz/cifar.html

See my notes on DL for details

# Algorithms: Deep Learning

- Adversarial Examples: Can I generate more training data by adding noise?
  - No… Szegedy et al. found that small perturbations in the example were sufficient to lead to example mis-classification.
    - The inclusion of some training example that has a small amount of noise added to it resulting in a change in classification is counter to the aim of obtaining a generalised performance from a network.



Correctly classified image | Perturbation of image | Incorrectly classified resultant image

From Szegedy et al,

Easy to see why with an $L_2$ loss function

$L_2$ measure a distance in a Euclidian space

Lots of small perturbations in a large dimensional space, can resut in a large change in $L_2$



Szegedy et al, ICLR, abs/1312.6199
Goodfellow et al, CoRR, abs/1412.6572, arXiv:1406.2661

See my notes on DL for details

# Algorithms: Deep Learning

- Generative Adversarial Networks
  - Use the features of the adversarial example to develop a new kind of model, optimising a “min-max game” of 2 distinct networks

Model = Discriminator + Counterfitter

  - Generative AI approach that can be used for simulation of realistic data from the counterfitter part of the model (initialised with white noise)
  - Aim to learn how to fool the discriminator by evolving the counterfit generated example images to a high standard

e.g. see Wang et al., arXiv:2308.08468

# Algorithms: Deep Learning

- Physics Inspired Neural Networks
  - The network is constructed from the perspective of solving differential equations, rather than arbitrary activation functions
  - Logic: for systems with underlying physics content (quantum mechanics, etc.) tailor the structure of the algorithm to use relevant terms that can be fit in optimisation.

**PDE Nondimensionalization**

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u}$$

$$\frac{\partial \mathbf{u}^*}{\partial t^*} + (\mathbf{u}^* \cdot \nabla^*)\mathbf{u}^* = -\nabla^* p^* + \frac{1}{Re}\nabla^{*2}\mathbf{u}^*$$

**Network Architecture**

- Fourier Feature Network
- Random Weight Factorization
- Exact Boundary Condition

**Training Algorithm**

- Loss Balancing
- Causal Training
- Curriculum Training

e.g. see Wang et al., arXiv:2308.08468

# Algorithms: Deep Learning

- Physics Inspired Neural Networks
  - Worth exploring for problems with underlying physics content
    - e.g. beam dynamics

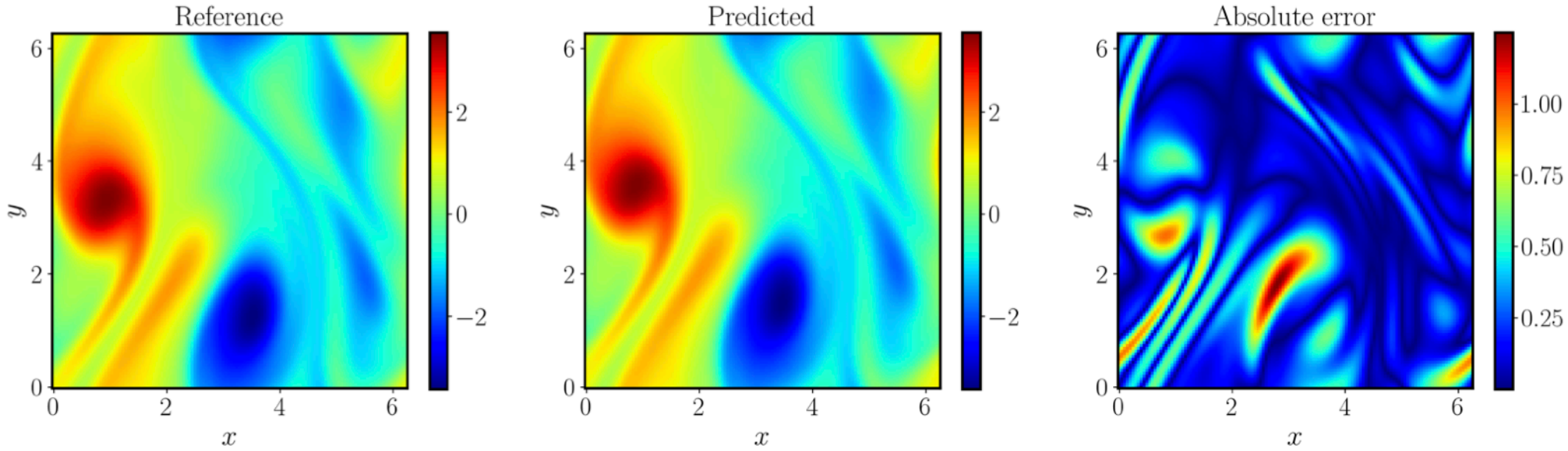



Figure 16: *Navier-Stokes flow in a torus:* Comparison of the best prediction against the reference solution at the last time step. The animation is provided in `https://github.com/PredictiveIntelligenceLab/jaxpi`.

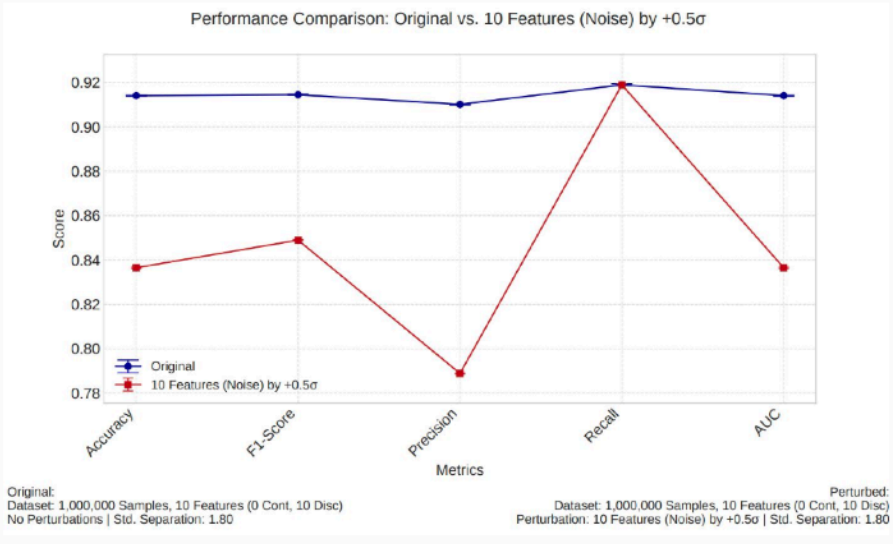
# Algorithms: Reflections and a brief note on Generative AI

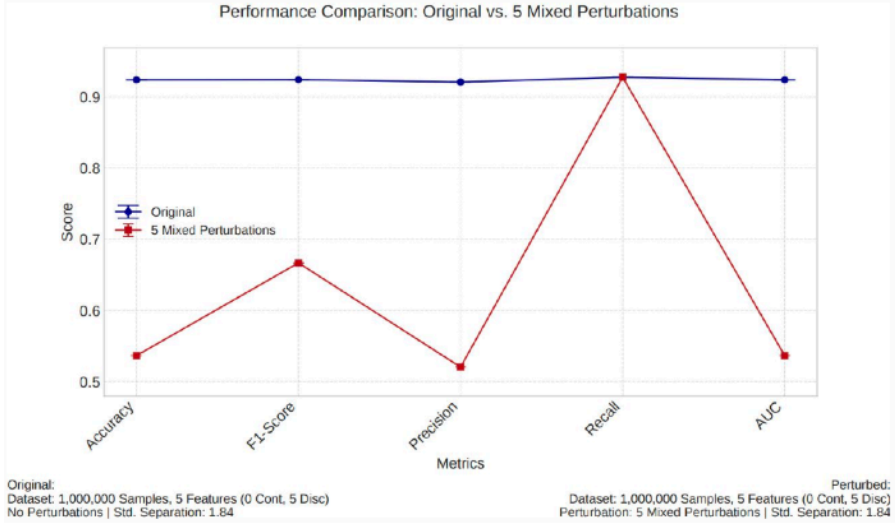
- Neural Nets are arbitrary function approximators:
  - Prone to overtraining
  - Most algorithms don't incorporate causal information; so poor uncertainty quantification
  - BEWARE: Systematic shifts of input can quickly kill the algorithm performance

**E. Cary MSc thesis**

- Simple MLP
- Gaussian input features
- <1$\sigma$ systematic shifts

**Clear demonstration that systematic shifts or drift can lead to performance collapse of networks: Uncertainty Quantification is important (and poorly studied)**



# Algorithms: Reflections and a brief note on Generative AI

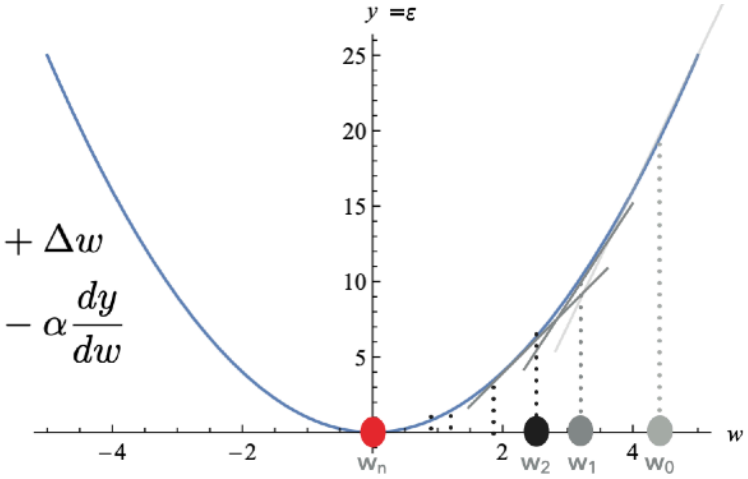
- Gen AI algorithms can be used to generate code, solve problems, create images etc.
- The outputs are not trustworthy from a scientific rigour perspective
- As with industry, we should use with caution and question everything in the outputs.
  - e.g. companies are shifting toward using generative AI for code generation. They find two possible outcomes:
    - People who know how to code become much more effective and the companies become more profitable
    - People who don't know how to code result in projects taking much longer and are losing money for those companies (those graduates won't last long in industry).
  - Let's do the former - not the latter!

# Data Wrangling

- Know your data:
  - Are the features well-behaved?
  - Do you understand the context of each input feature?
  - Are there systematic trends in performance with time (e.g. detector calibration drift)?
  - Is the data complete?  Any NaN's or missing information in features you want to use?
  - Have you cleaned your data?
  - Have you transformed your data to speed up optimisation?

# The Optimisation Problem And What It Means

- Weights are randomly initialised, typically in the range [0, 1]
- A loss function/objective function is defined
- The objective function is minimised using some iterative algorithm; typically Gradient descent inspired: *assumes* parabolic minima

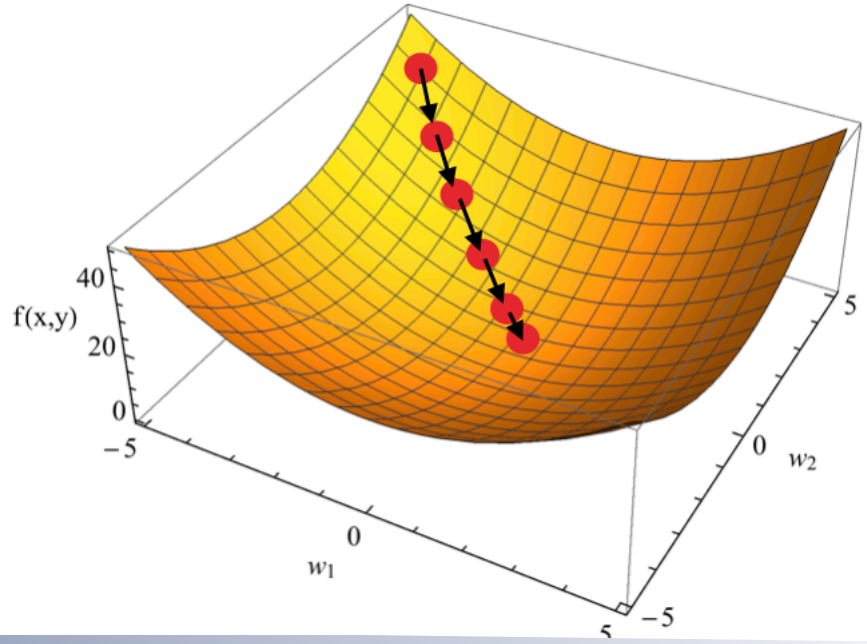



$$w_{i+1} = w_i + \Delta w$$
$$= w_i - \alpha \frac{dy}{dw}$$

$$\Delta y = \Delta w \frac{dy}{dw}$$
$$= -\alpha \left(\frac{dy}{dw}\right)^2$$

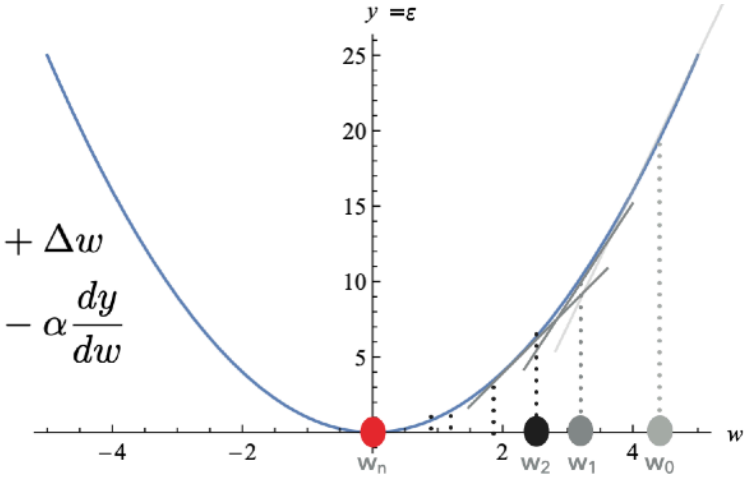
α is the learning rate: a small positive number

Choose $\Delta w = -\alpha \frac{dy}{dw}$ to ensure Δy is always negative.



# The Optimisation Problem And What It Means

- Weights are randomly initialised, typically in the range [0, 1]
- A loss function/objective function is defined
- The objective function is minimised using some iterative algorithm; typically Gradient descent inspired: *assumes* parabolic minima

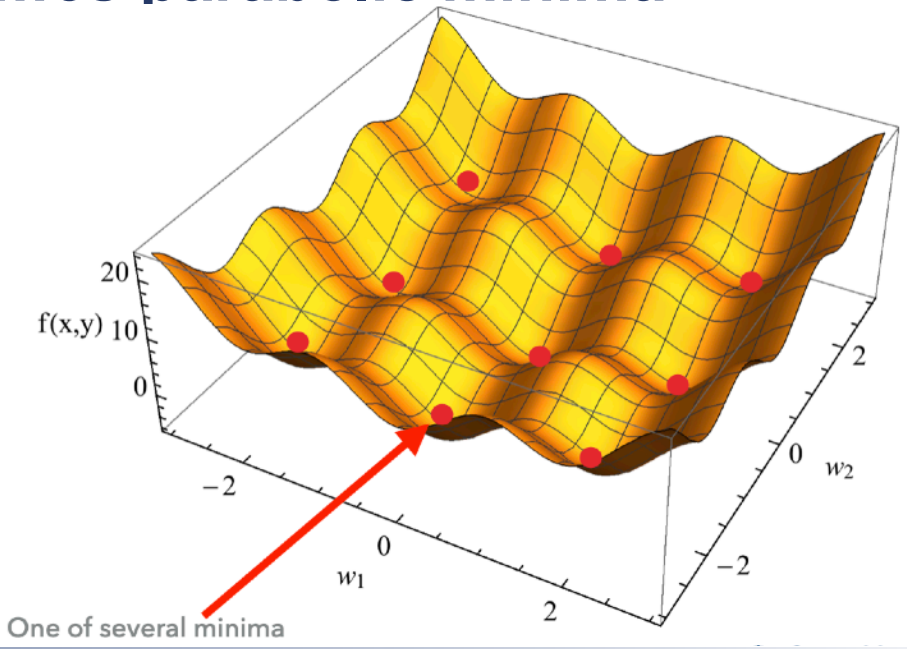



$$w_{i+1} = w_i + \Delta w$$
$$= w_i - \alpha \frac{dy}{dw}$$

$$\Delta y = \Delta w \frac{dy}{dw}$$
$$= -\alpha \left(\frac{dy}{dw}\right)^2$$

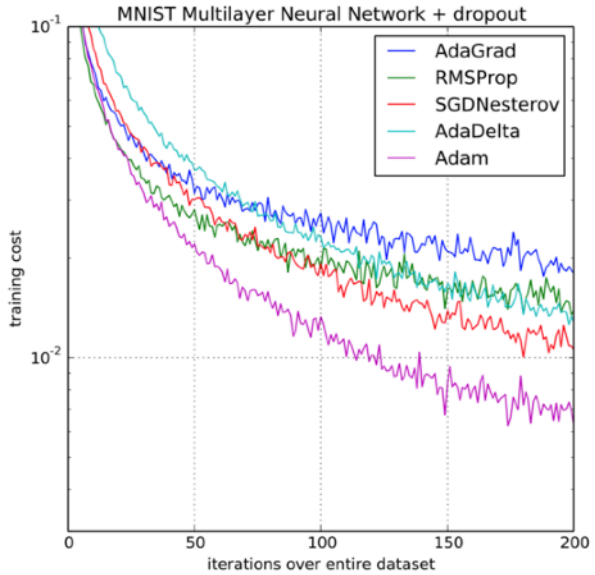
α is the learning rate: a small positive number

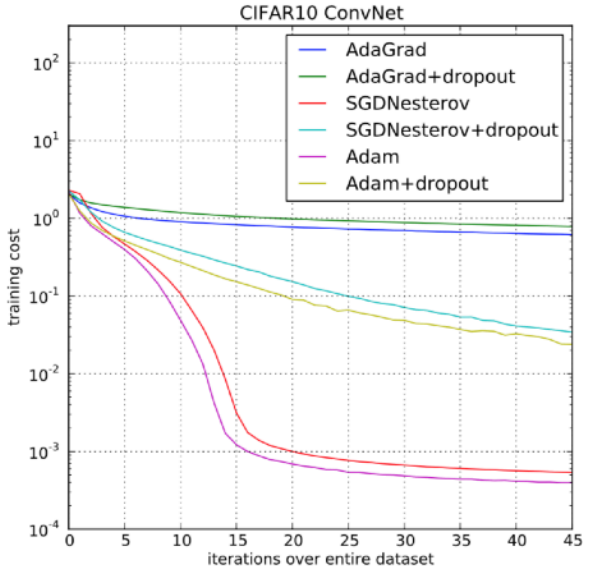
Choose $\Delta w = -\alpha \frac{dy}{dw}$ to ensure Δy is always negative.



# The Optimisation Problem And What It Means

- Lots of different algorithms available
- Trial and error used to identify those that work
- The ADAM optimiser is the alg. of choice currently: ADAptive Moment estimation based on gradient descent
- Motivated by tests using MNIST and CFAR10 optimisation of deep learning models



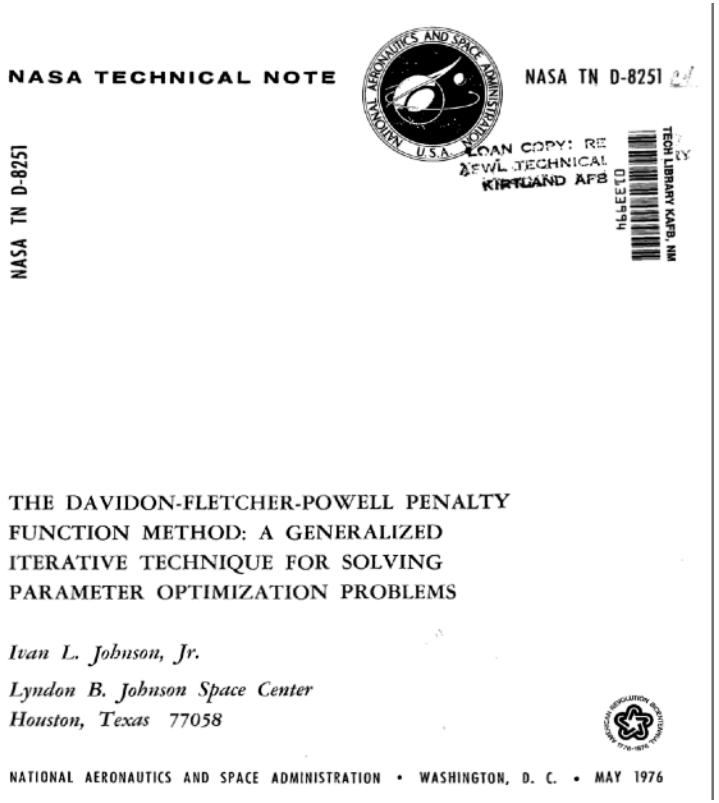




- Fast
- Robust (for minimisation), but can not guarantee the global minimum is found
- Don't care about errors
- Available in modern ML model building frameworks

Kingma and Ba, Proc. 3rd Int Conf. Learning Representations, San Diego, 2015

# The Optimisation Problem And What It Means

- c.f. Particle Physics Optimiser of choice
- Variable Metric Matrix Optimisation method from the late 50's/early 60's encoded in MINUIT
  - Davidon-Fletcher-Powell algorithm
  - Supersceded quickly by other algs. but still widely used.
- Very slow in comparison to ADAM
- But robust (based on 40 year old benchmark tests)
- Errors determined once the estimator of the minimum located


NASA TECHNICAL NOTE

NASA TN D-8251

NASA TN D-8251

LOAN COPY: RE
AFWL TECHNICAL
KIRTLAND AFB

TECH LIBRARY KAFB, NM

THE DAVIDON-FLETCHER-POWELL PENALTY FUNCTION METHOD: A GENERALIZED ITERATIVE TECHNIQUE FOR SOLVING PARAMETER OPTIMIZATION PROBLEMS

*Ivan L. Johnson, Jr.*

*Lyndon B. Johnson Space Center*
*Houston, Texas 77058*

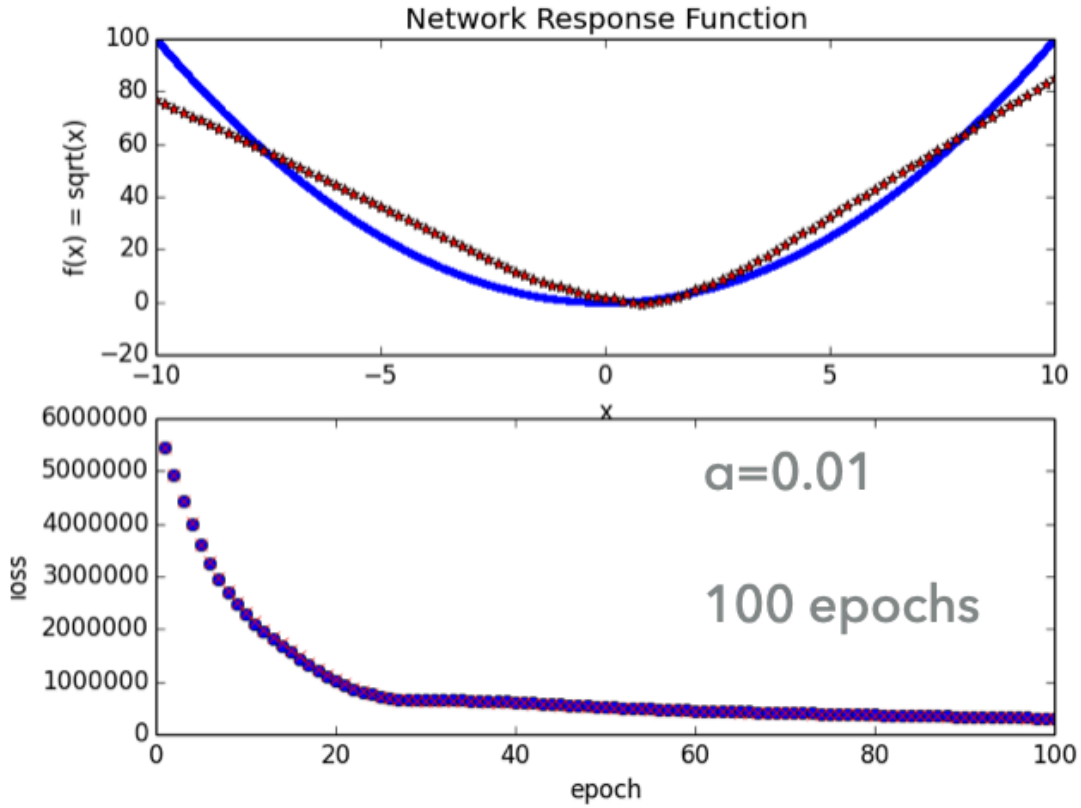
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • MAY 1976

**ML optimisation very different to MINUIT - neither guarantee convergence to global minimum, MINUIT used to determine errors on the parameters of interest and nusiance parameters.**

https://ntrs.nasa.gov/api/citations/19760017876/downloads/19760017876.pdf

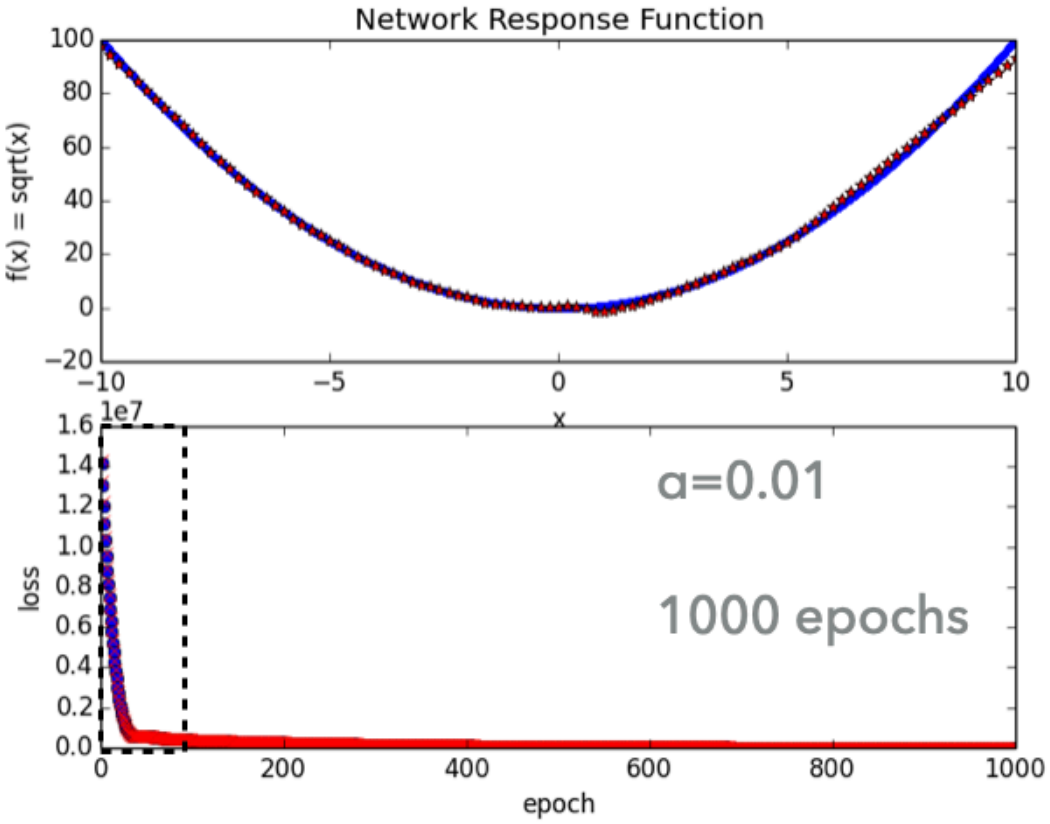
# The Optimisation Problem And What It Means

- Is the result sensible?
  - Hard to say for sure…
- Checks and concerns
  - Use test/train validate statistically independent samples to compare performance to gain confidence
  - Cross validation is a method that can help understand variance in the model predictive power
  - Is the result biased?
    - e.g. see the review by Mikolajczyk-Barela and Grochowski: https://arxiv.org/abs/2308.11254



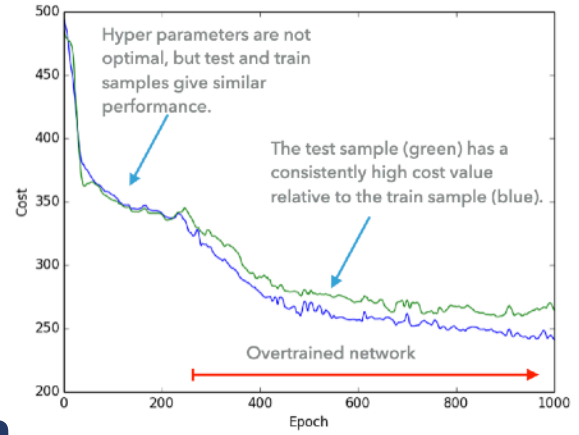
# The Optimisation Problem And What It Means

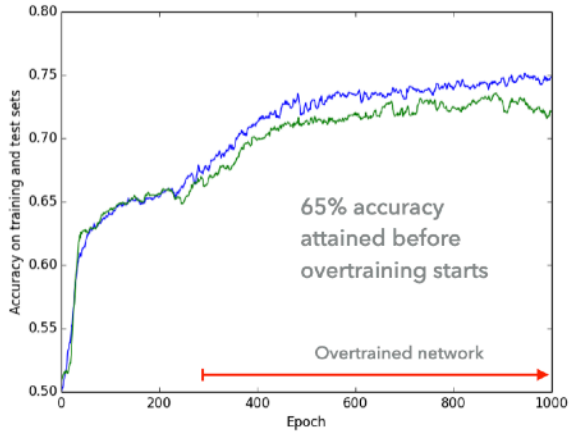
- Is the result sensible?
  - Hard to say for sure…
- Checks and concerns
  - Use test/train validate statistically independent samples to compare performance to gain confidence
  - Cross validation is a method that can help understand variance in the model predictive power
  - Is the result biased?
    - e.g. see the review by Mikolajczyk-Barela and Grochowski: https://arxiv.org/abs/2308.11254

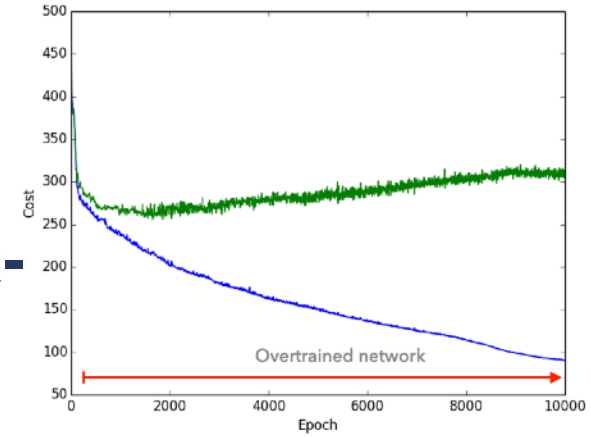


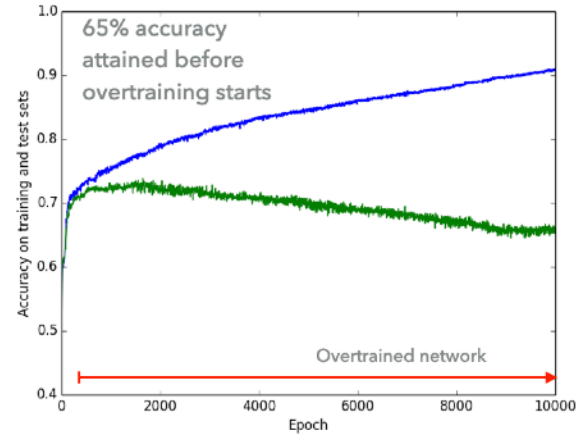
# The Optimisation Problem And What It Means

- Is the result sensible?
  - Hard to say for sure…
- Checks and concerns
  - Use test/train validate statistically independent samples to compare performance to gain confidence
  - Cross validation is a method that can help understand variance in the model predictive power
  - Is the result biased?
    - e.g. see the review by Mikolajczyk-Barela and Grochowski: https://arxiv.org/abs/2308.11254



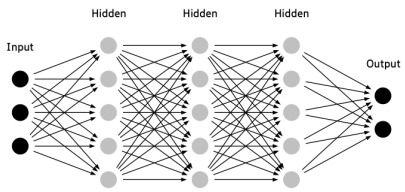



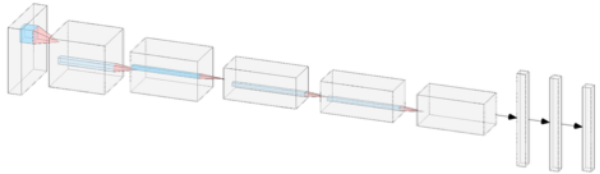



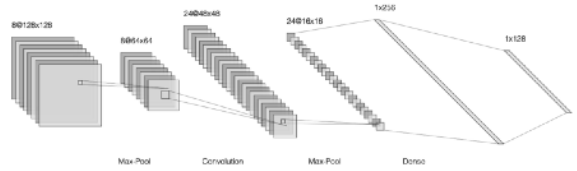
# Tools And Methods

- Lots out there to use
- As with everything in science, expect to find mistakes*
- Different implementations and technologies provide stand alone tools
- How do these interface into your analysis or the basf2 framework?

Kaggle has a page on tools for drawing networks in different styles:
https://www.kaggle.com/discussions/getting-started/253300






*e.g. the TMVA overtraining check pots use an incorrect calculation of the KS probability. This is known to the TMVA and ROOT developers, but they have not corrected the mistake in over a decade. Do your homework to validate the code is doing something sensible!

# Tools And Methods

- ROOT:
  - TMVA - fully integrated toolkit with a range of old and modern algorithms
  - Benefits from using a strongly typed language that is compilable (fast implementation and catch errors at compile time)

- Python:
  - pyROOT provides access to TMVA
  - FastBDT
  - sklearn
  - TensorFlow
  - KERAS
  - pyTORCH
  - …

Wide range of toolkits:
- Some are for dedicated algorithms (FastBDT/svmlib)
- Some are toolkits/frameworks for building algorithms:
  - TensorFlow
  - KERAS
  - pyTORCH
  - Etc.

# Tools And Methods

- Deep learning algorithms benefit from complex models that need large datasets to train properly

- Handing the data I/O and memory storage can be a problem:
  - e.g. ATLAS lost 6 months figuring out how to train a deep network algoritim while the worked to figure out how to stage loading data

- Belle II recommends the use of ONNX to integrate models into basf2.
  - This was discussed recently in a talk by Nikolai Krug in the Belle II Belle II Software Developers Meeting on the 21st August.
  - See this link for the talk.
  - See this link for more information on ONNX.

I have more notes on machine learning online at: https://pprc.qmul.ac.uk/~bevan/teaching.html.

# Resources

- Lots of material out there, as usual most of it is not very good. Some useful books on the topic include:
  - Bishop: Neural Networks
  - Goodfellow: Deep Learning
  - Friedman, Tibshirani, and Hastie: The Elements of Statistical Learning
  - Online resources e.g. Kaggle, DataCamp
  - Etc.

An old paper that I recommend everyone read is Yann Le Cun's Efficient Backprop paper in *Neural Networks Tricks of the Trade*.

Yes it is ancient (1998)

… but it is so relevant in designing model workflows that are energy and resource efficient

# Closing Remarks

- Rely on a benchmark simple algorithm as a baseline
- Learn the limitations of the algorithms you use to get a feel for when something is wrong
- Understand your data;
  - can you apply a preselection to simplify the problem?
  - What is the question you want to ask of the data with the ML algorithm?
  - Does your question make sense?
  - How will you verify that the algorithm is probably doing what you want?
  - etc.

# Closing Remarks

- Think about data and resources:
  - Do you have enough data to benefit from a complex algorithm?
  - Are you selecting a sensible algorithm that is a good fit to your problem?
  - Are your computing resources sufficient to learn a model?
  - Is a complex algorithm justified, given the potential performance gain and impact on the environment?
  - Is your data biased?
  - etc.

# Thank you for listening