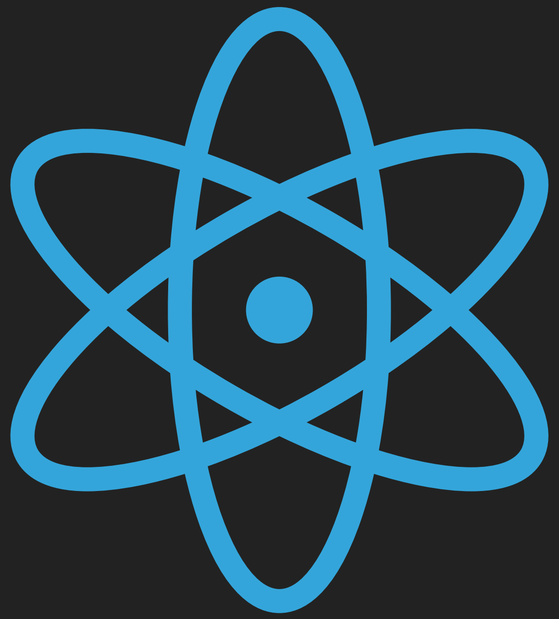
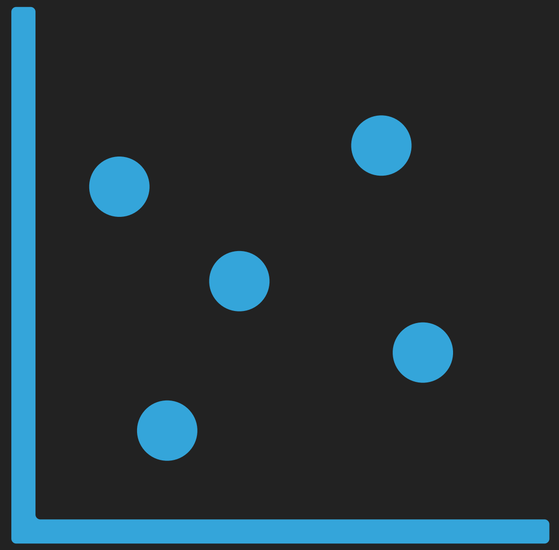
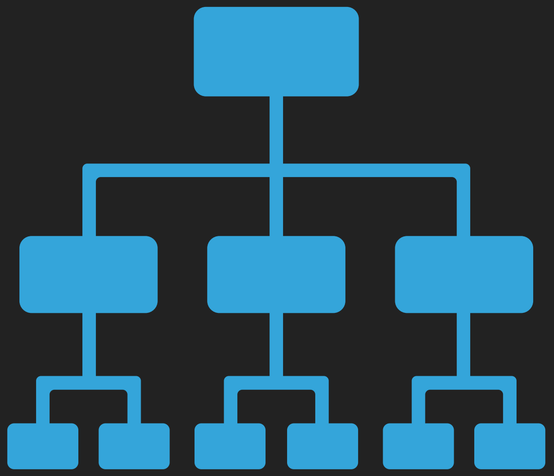
UNIVERSITY OF SUSSEX


Supervised Classification Tools

T. Stevenson

25 July 2019

Introduction

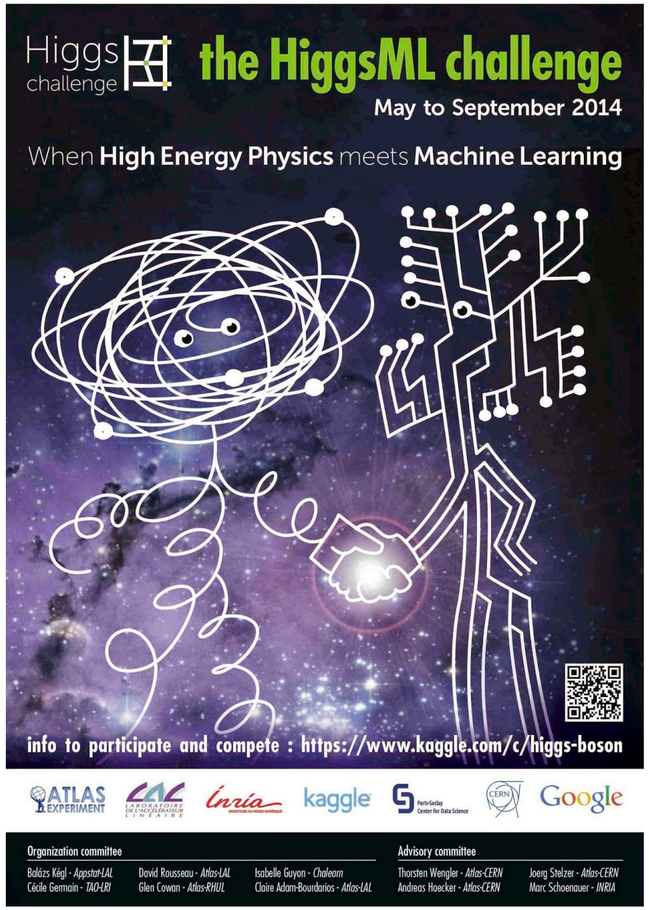
UNIVERSITY OF SUSSEX

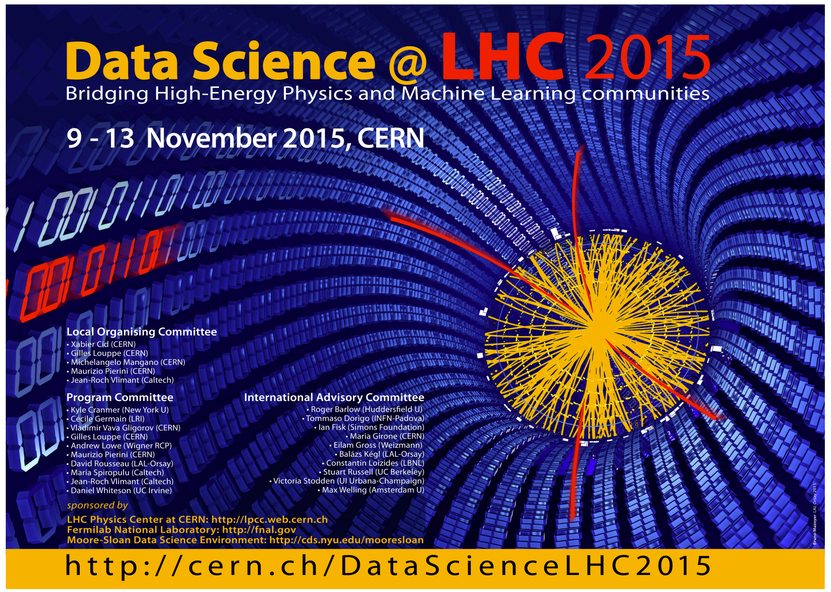
- What is Supervised Learning?
  - Utilise labelled events of input-output pairs
  - Objective is to learn the mapping between input information (a combination of multiple discriminating variables) and the desired output (either a category or a real value depending on the algorithm)
  - Apply trained technique to unseen data to predict output

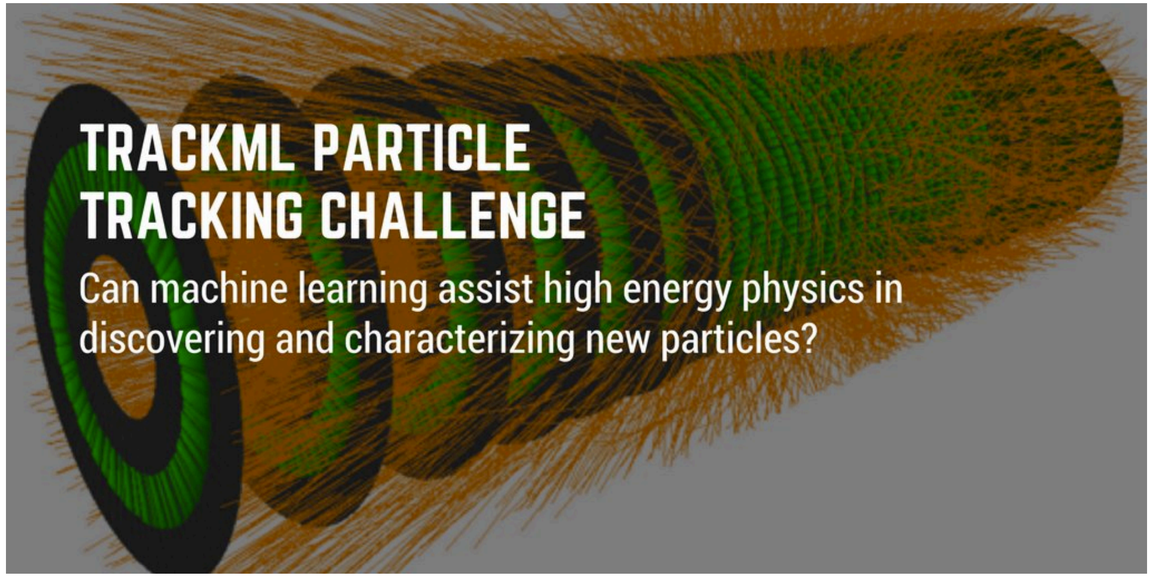
Introduction

- Supervised ML broadly split into two categories:
  - **Classification**:
    - Predicting the label of the data from a finite set
  - **Regression**:
    - Predicting a continuous target variable

Machine Learning Interest







Introduction - Toolkits

Introduction - Toolkits

Introduction

- Topics to be covered:
  - Linear Discriminant Analysis (Fisher Discriminant)
  - Naïve Bayes
  - Support Vector Machines
  - Random Forests and Boosted Decision Trees
  - Neural Networks and Logistic Regression
  - General Techniques

Code for Today

UNIVERSITY OF SUSSEX

github.com/TomStevo/SupervisedML/

- Set of jupyter notebooks
- Can be download or run in binder

LINEAR DISCRIMINANT ANALYSIS

Linear Discriminant Analysis

- Fisher discriminant is linear combination of input variables, $\underline{\mathbf{x}}$, to produce a single output, y
- Weights α_i have to be determined to maximise separation between signal and background
- β doesn't affect separation, just central value of output distribution

$$\mathbf{y} = \sum_{i=1}^{n} \alpha_i x_i + \beta,$$
$$= \underline{\boldsymbol{\alpha}} \cdot \underline{\mathbf{x}} + \beta$$

Linear Discriminant Analysis

- Separation defined as ratio of difference between means of signal and background distributions squared and sum of variances

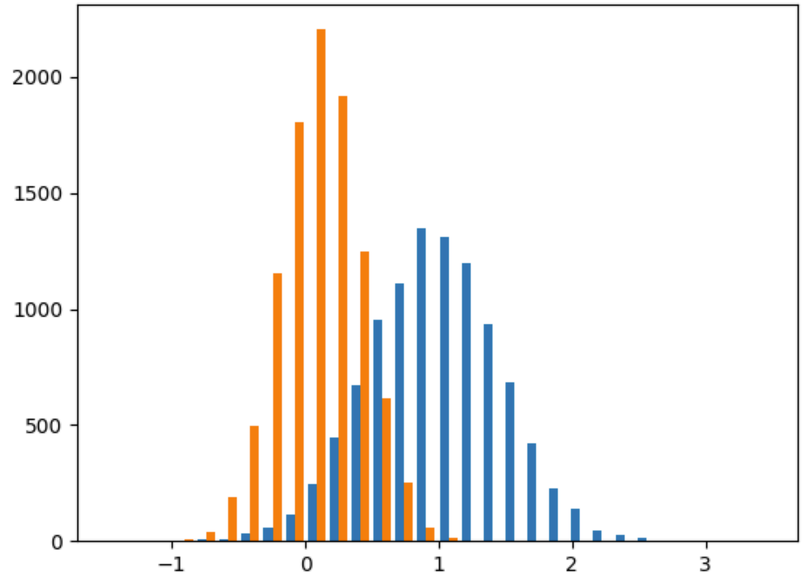
$$S = \frac{(\mu_s - \mu_b)^2}{\sigma_s^2 + \sigma_b^2}$$

- Can be shown that the maximum separation is found when

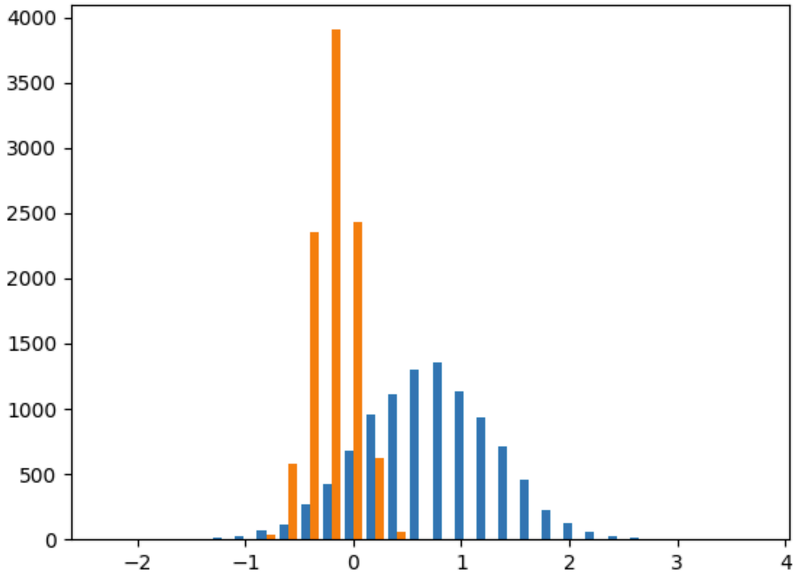
$$\alpha \propto W^{-1}(\underline{\mu}_s - \underline{\mu}_b)$$

where W is the covariance matrix of the input variables

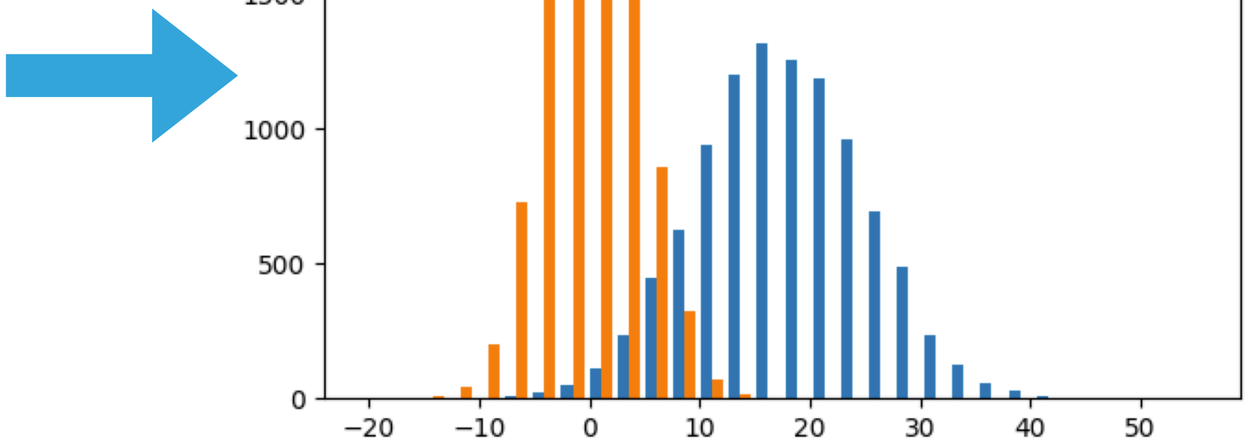
Linear Discriminant Analysis



x_1



x_2



y

- Notebook: LDA.ipynb

NAÏVE BAYES

$$\frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Naïve Bayes

- Application of Bayes' theorem with "naïve" assumption that the value of a particular feature is independent to the value of another feature

$$P = (y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)}$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid \mathbf{y})$$

- Can perform quite well despite oversimplified assumptions

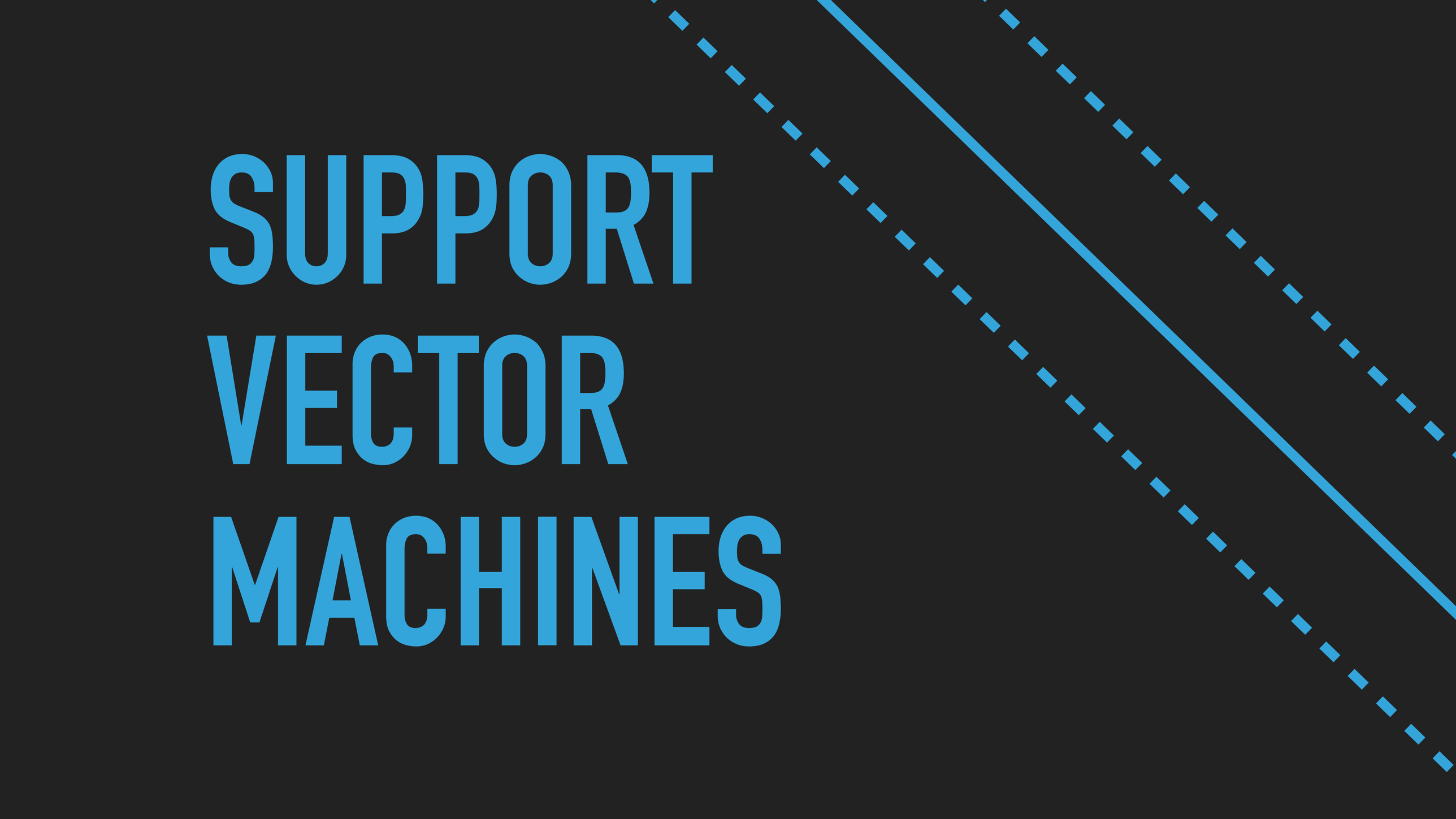
- Only require a small amount of training data to train

Naïve Bayes

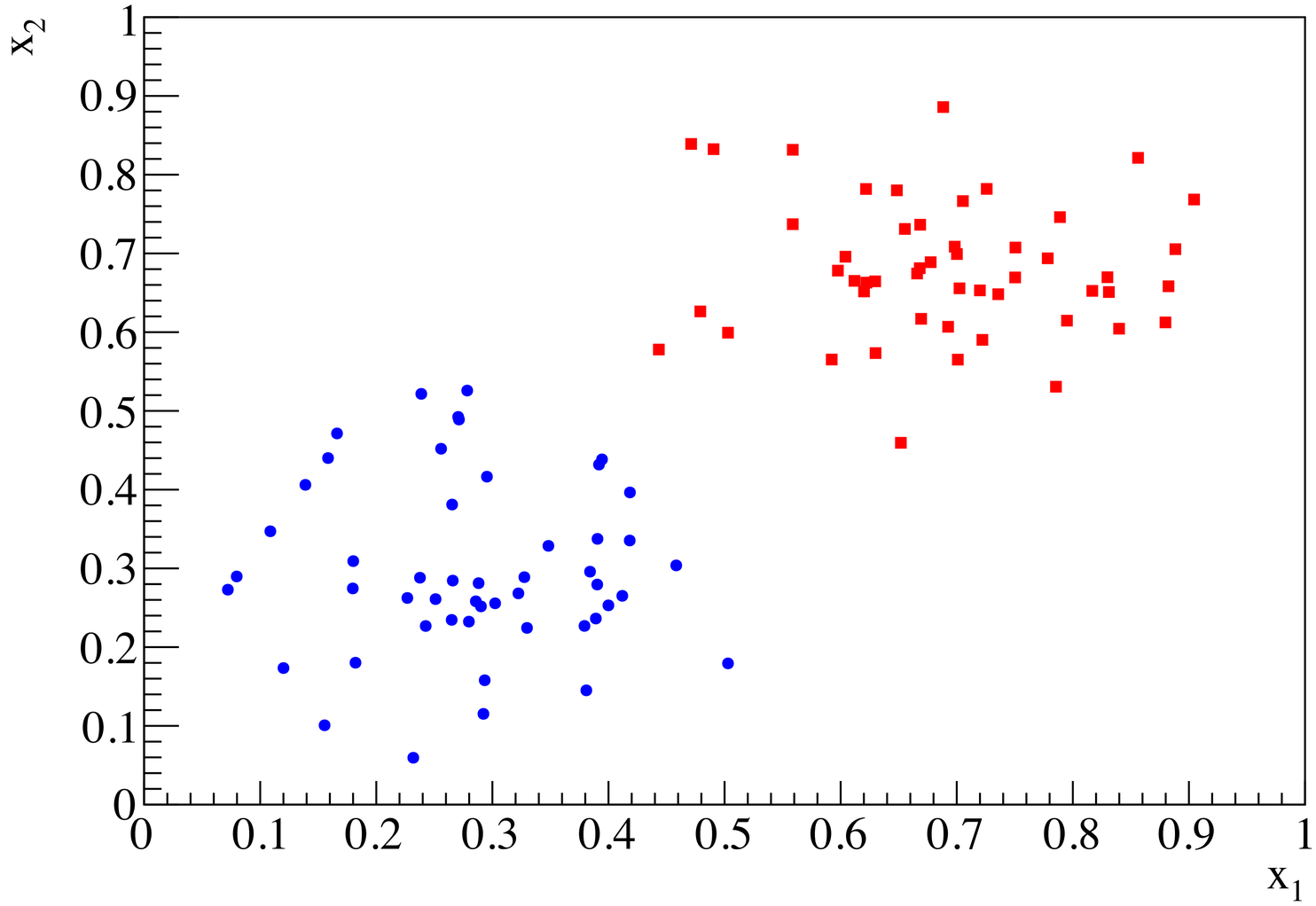
UNIVERSITY OF SUSSEX

- Notebook: NaiveBayes.ipynb

SUPPORT VECTOR MACHINES

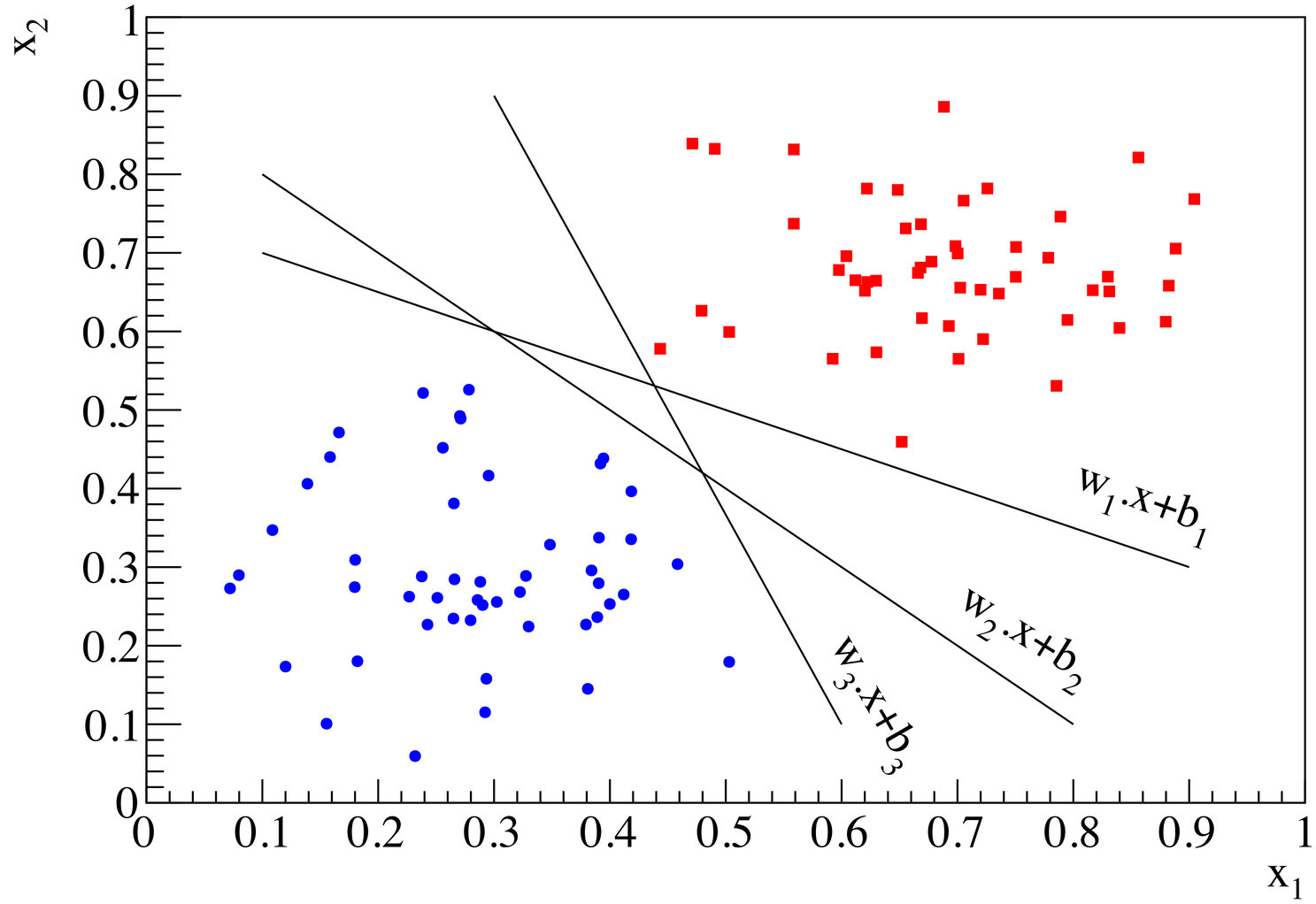


Support Vector Machines



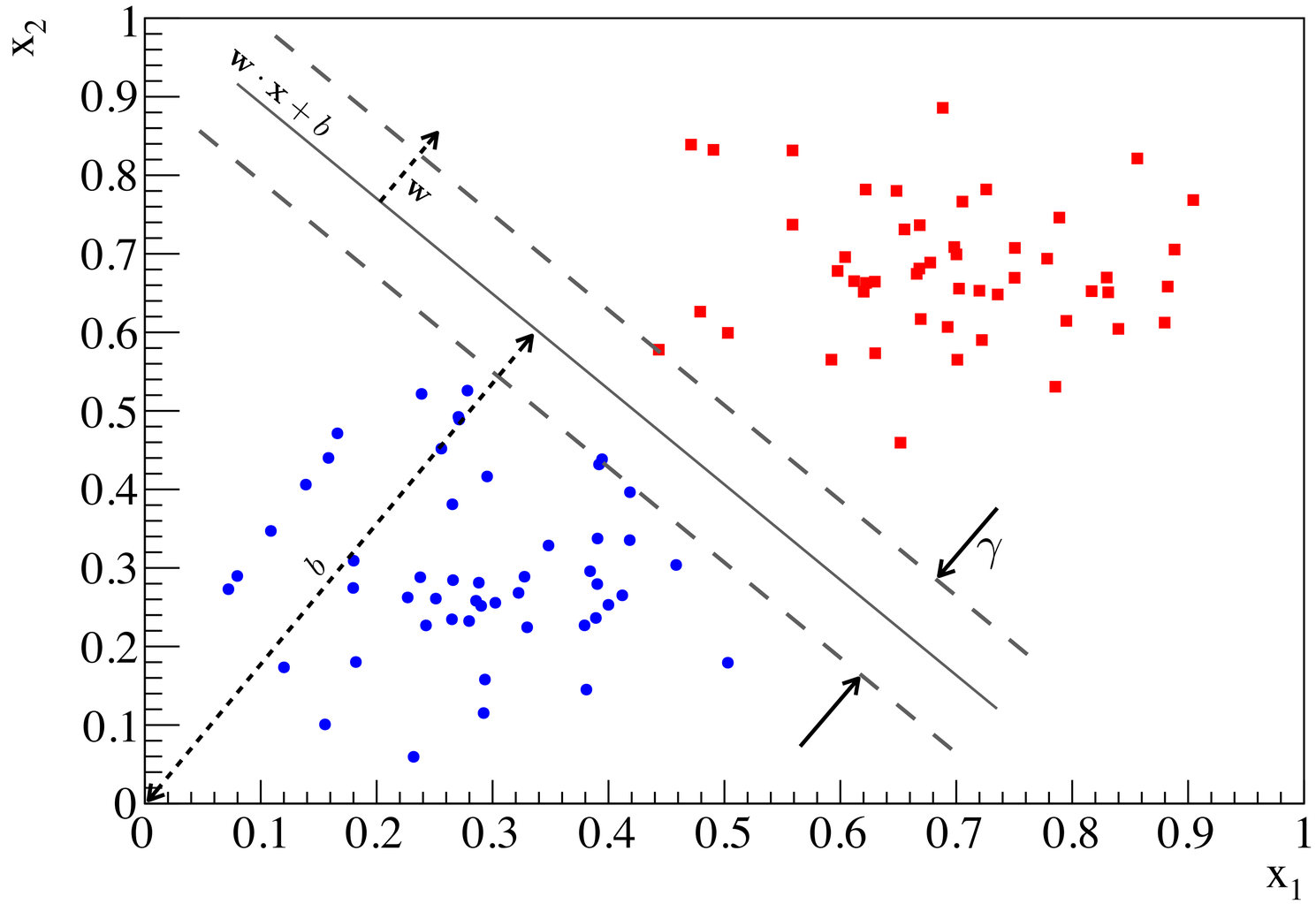
- Want to distinguish between the signal and the background

Support Vector Machines



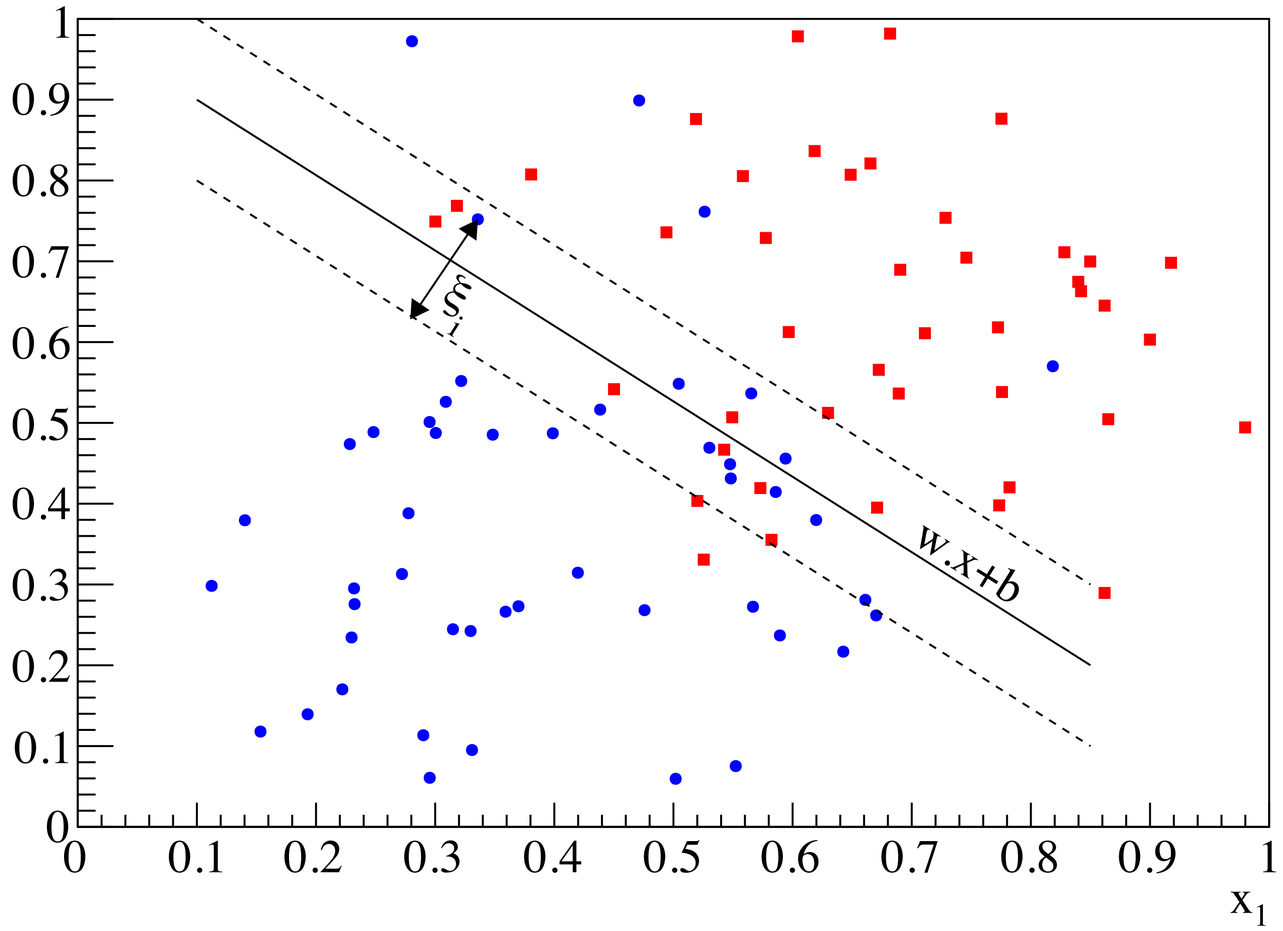
- Object to find separating hyperplane:
  - Arbitrary choice without further constraints

Support Vector Machines



- SVM achieves this by maximising the margin, γ:
  - Distance between hyperplane and points closest to the decision boundary, known as support vectors (SV)

Support Vector Machines



- Previous example assumes data are linearly separable
- Relax "hard margin" constraint by introducing misclassification
- Described by:
  - Slack (ξ_i) - distance from the hyperplane (defined by the margin) to the ith SV
  - Cost (C) - tuneable weight penalising misclassified points. Multiplies sum of slack parameters
- More useful for HEP problems.

Support Vector Machines

UNIVERSITY OF SUSSEX

- SVMs expressed in terms of inner product.
- Kernel functions $K(x,y)$ used in place of inner product.
- $K(x,y)$ maps the problem from input space, X, to potentially higher dimensional, implicit feature space, $\mathrm{F}=\{\phi(x)|x \in \mathrm{X}\}$ where data may then be separable:

$$K(x, y) = \langle \phi(x) \cdot \phi(y) \rangle$$

- Feature map, $\gamma(x)$, and the feature space do not need to be known, "kernel trick".
- Some properties required to be a proper kernel function.
  - Details and derivations can be found in "An Introduction to Support Vector Machines and Other Kernel-based Learning Methods" by Cristianini and Shawe-Taylor and references therein.

Support Vector Machines

UNIVERSITY OF SUSSEX

Polynomial

$$K(x,y) = (\mathbf{x}\cdot\mathbf{y}+c)^d = \left(\sum_{i=1}^{\dim(X)} \mathbf{x}_i\mathbf{y}_i + c\right)^d$$

Radial Basis Function (RBF)

$$K(x,y) = e^{-\Gamma\|x-y\|^2}$$

Multi-Gaussian

$$K(x,y) = \prod_{i=1}^{\dim(X)} e^{-\Gamma_i(x_i-y_i)^2}$$

Support Vector Machines

UNIVERSITY OF SUSSEX

- Notebook: SVM.ipynb

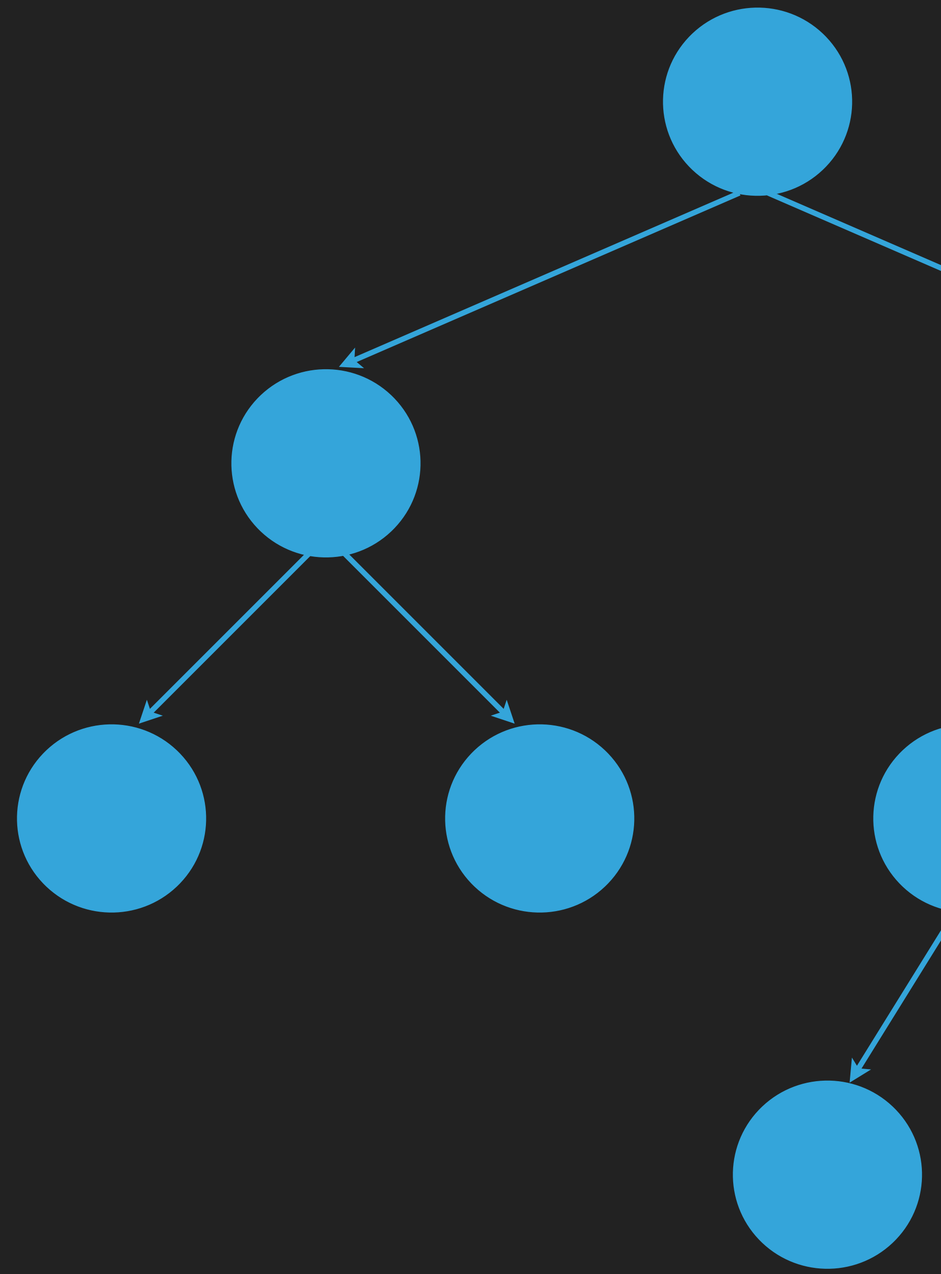
- Play around with some of the tuneable parameters (docs):
  - C : float, optional (default=1.0)
    - Penalty parameter C of the error term

  - kernel : string, optional (default='rbf')
    - rbf, poly, sigmoid, etc.

  - gamma : float, optional (default='auto')
    - Kernel coefficient for 'rbf', 'poly' and 'sigmoid'

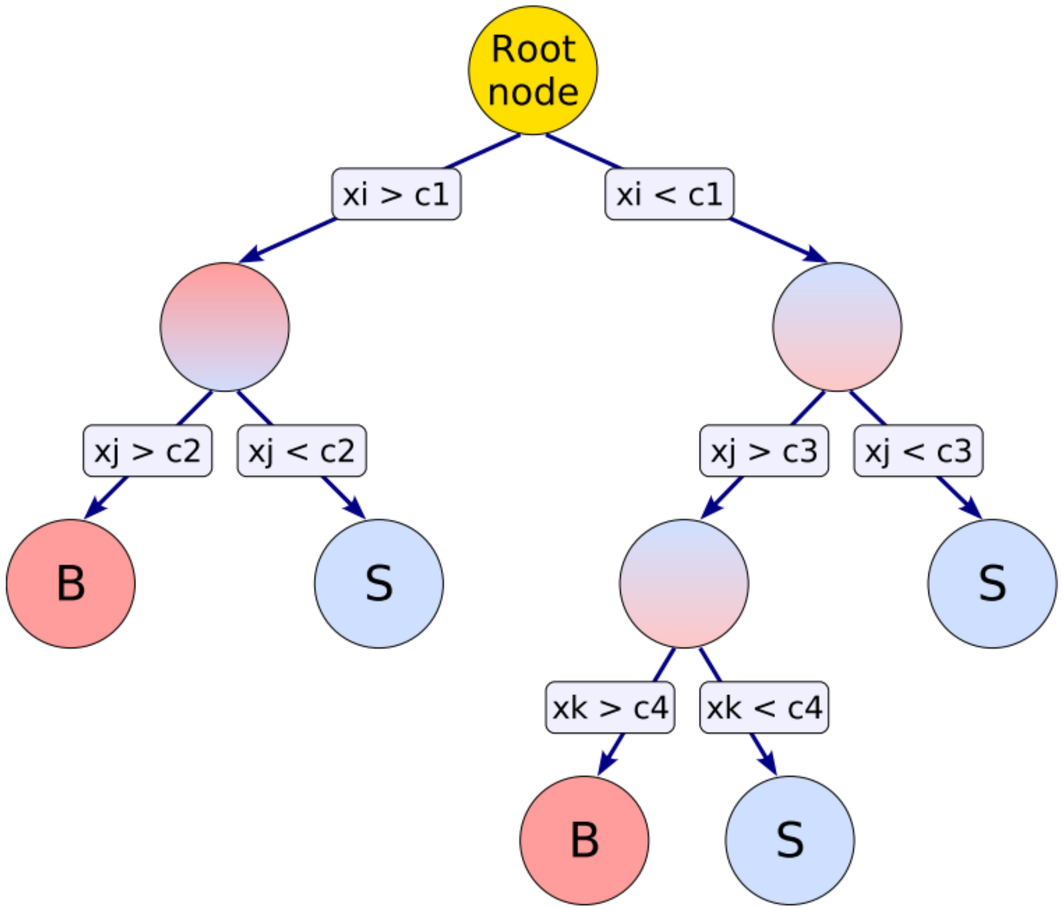
Note: Perform better on normalised data

DECISION TREES

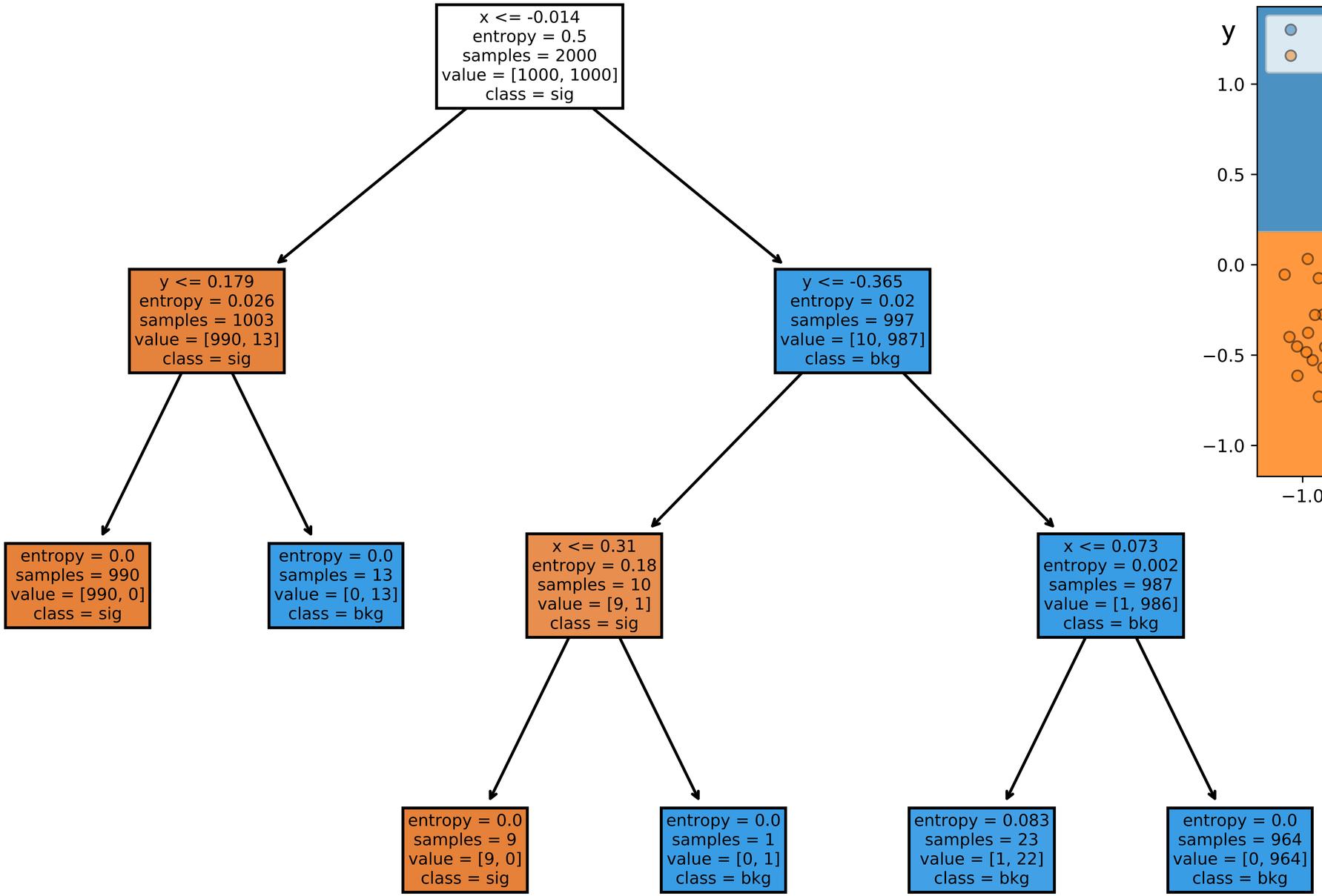


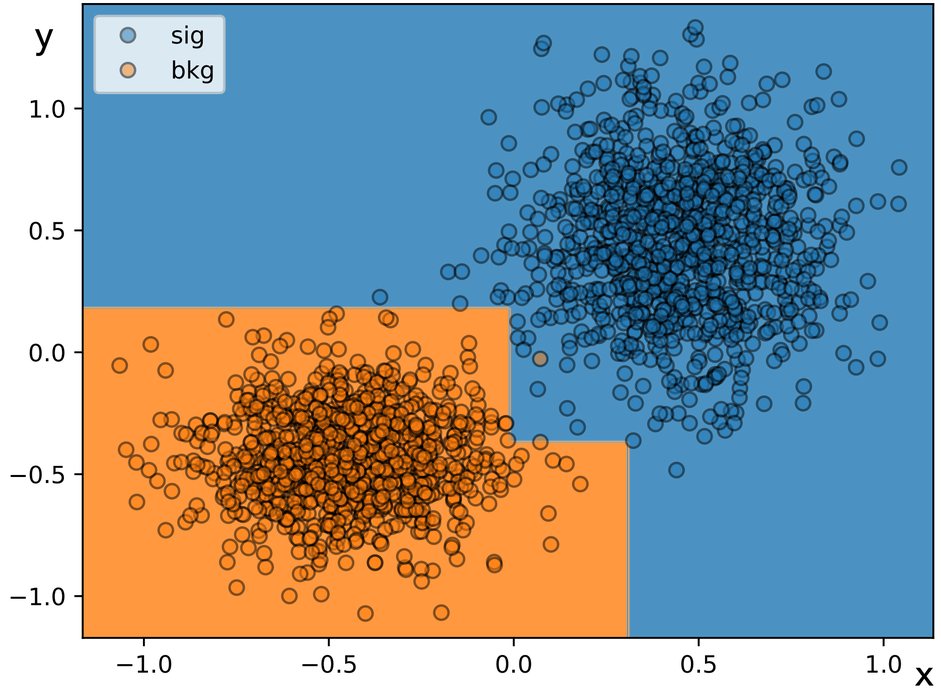
Decision Trees

- Decision Trees:
  - Structure of repeated binary cuts on individual variables
  - Optimised at each node to give best separation
    -
  - Route to each final node (leaf) represents different subset in hyperspace
    - Key advantage over cuts based



Decision Trees





More Decision Trees

UNIVERSITY OF SUSSEX

- DTs very susceptible to overtraining

- Use Random Forests to overcome this:
  - Combines many weaker learners via majority voting to form a stronger one
  - Bagging:
    - Trainings on resampled subsets
  - Boosting:
    - Iterative reweighting of misclassified events
    - Example on next slide

Random Forests

UNIVERSITY OF SUSSEX

- Trees are randomly "grown" from sample drawn with replacement (i.e. bootstrapping) from training set
  - Any data point can be sampled multiple times

- Rather than considering all features at a node, a random subset of the features is also considered

- The results of all trees are then averaged:
  - Aim to decrease the variance w.r.t. a single tree
  - Bias may slightly increase

UNIVERSITY OF SUSSEX

Adaptive Boosting

- Extremely popular boosting algorithm
- Iterative Process:
  - After each step, mis-classified events are weighted to increase their importance
    - Overall weight remains constant
  - "Weak learners" are combined in a weighted sum to give final output
- Has a tuneable parameter to adjust learning rate
- Works best on small individual trees

$$\alpha_i = \frac{1 - \mathrm{err}_i}{\mathrm{err}_i}$$

Misclassification rate

$$y_{\mathrm{Boost}}(\mathbf{x}) = \frac{1}{N_{\mathrm{trees}}} \cdot \sum_{i}^{N_{\mathrm{trees}}} \ln(\alpha_i) \cdot h_i(\mathbf{x})$$

Individual tree output

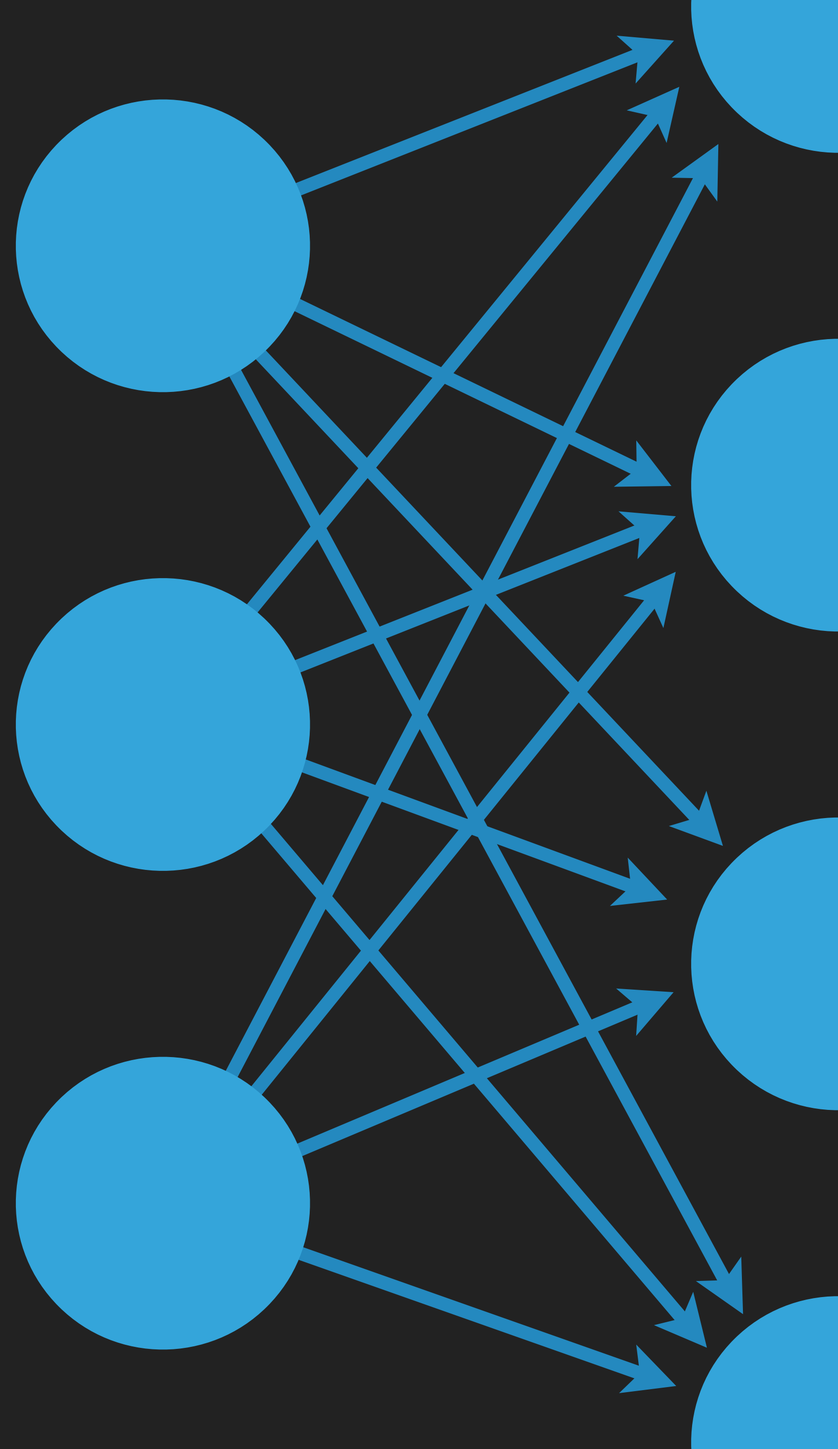
Boosted Decision Trees

UNIVERSITY OF SUSSEX

- Notebook: DecisionTrees.ipynb

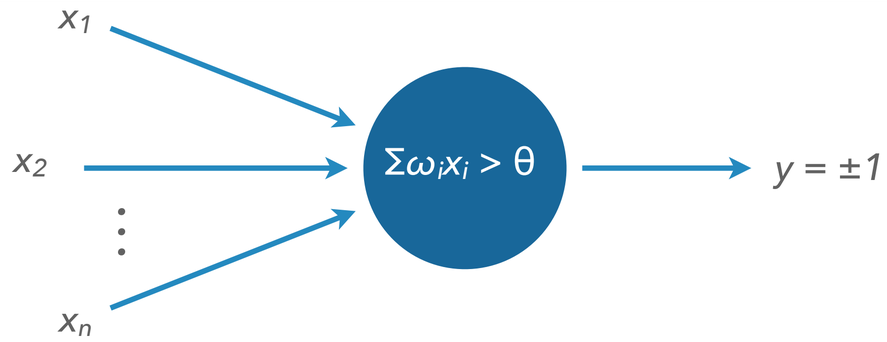
- Play around with some of the tuneable parameters:
  - DecisionTreeClassifier
  - AdaBoostClassifier
  - RandomForestClassifier
  - GradientBoostingClassifier

ARTIFICIAL NEURAL NETWORKS



Artificial Neural Networks

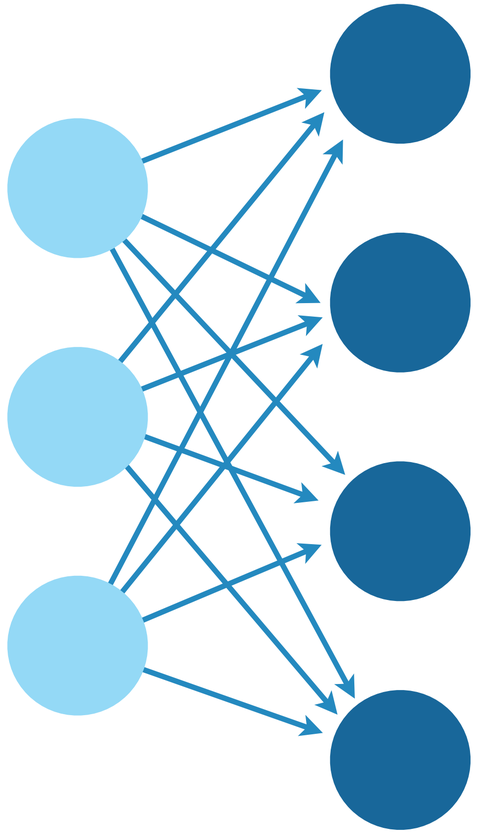
- Series of connected “perceptrons”:
  - Each takes *n* inputs
  - Compute weighted sum of inputs
  - Compare to reference threshold
  - Produce output, e.g. y = ±1



- Single perceptron defines separating hyperplane

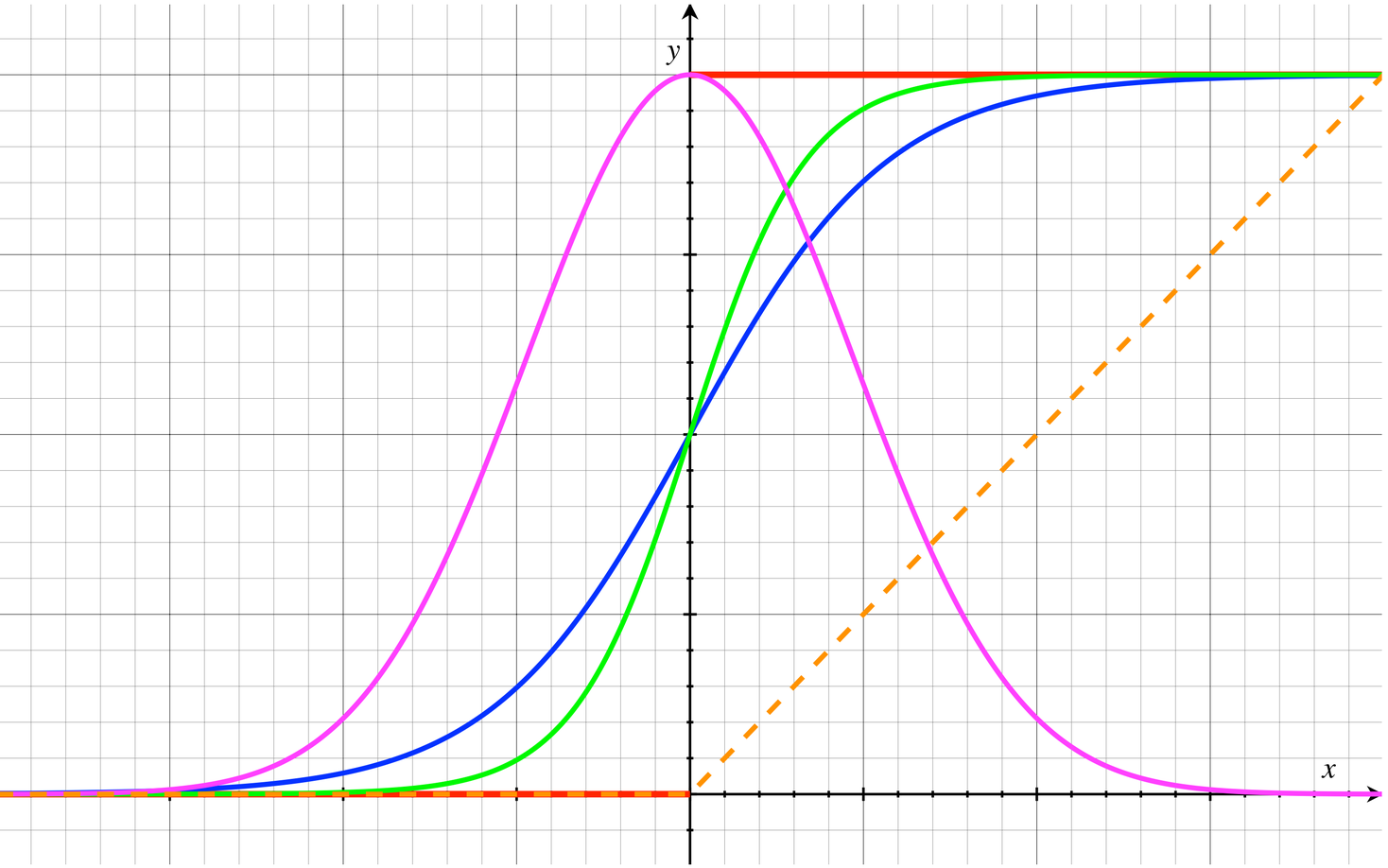
- Training optimises weights for each perceptron

- Useful to produce continuous output between two extremes:
  - Activation functions



UNIVERSITY OF SUSSEX

Activation Functions



Binary

$$y = \begin{cases} 0 & x < 0 \\ 1 & x > 0 \end{cases}$$

Logistic

$$y = \frac{1}{1 + e^{-x}}$$

Hyperbolic Tangent

$$y = \tanh x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

} "Sigmoid"

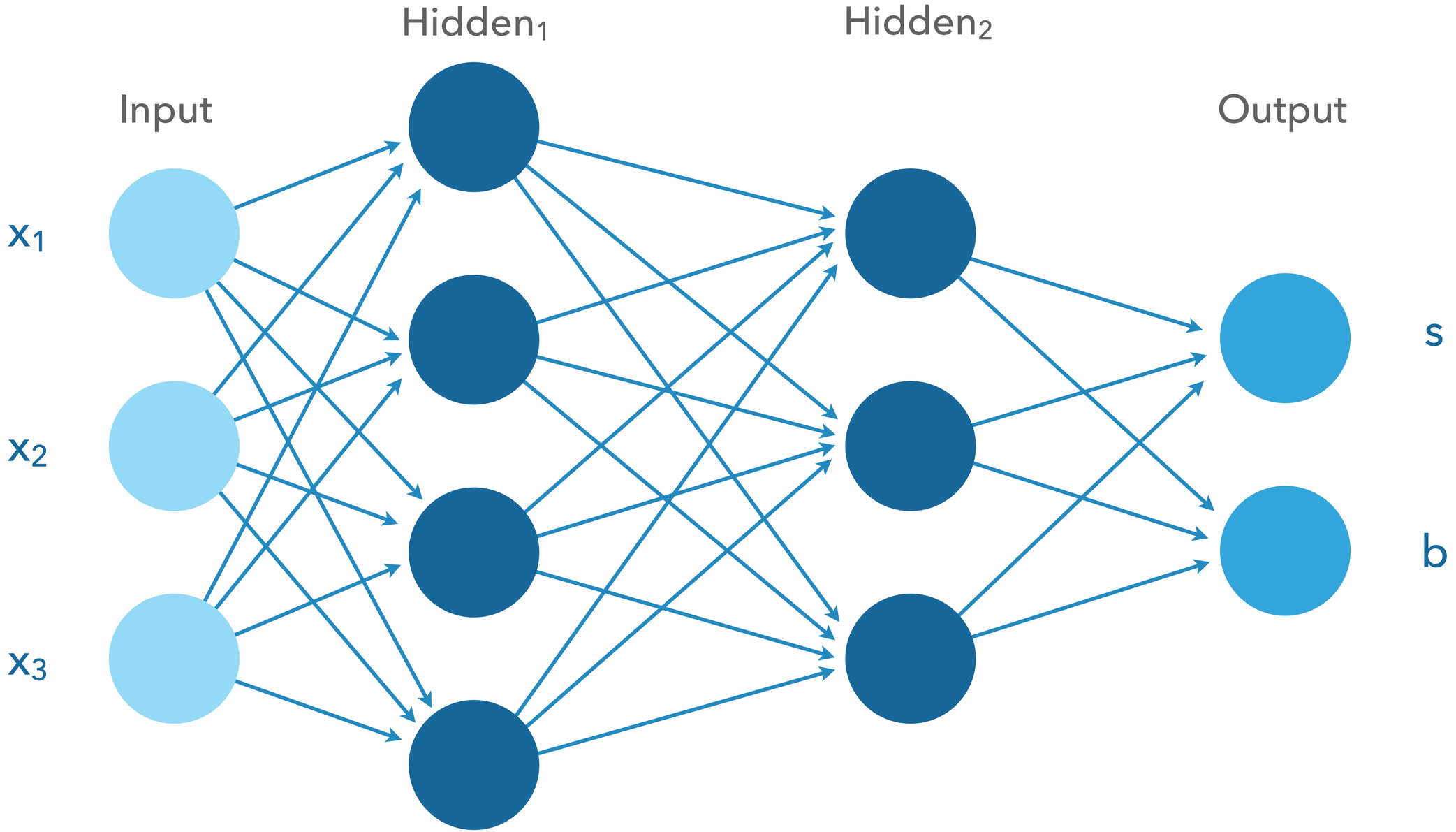
Radial Basis

$$y = e^{-x^2}$$

Rectified Linear Unit ("Relu")

$$y = \max(0, x)$$

Artificial Neural Networks



Artificial Neural Networks

- Notebook: NeuralNetwork.ipynb

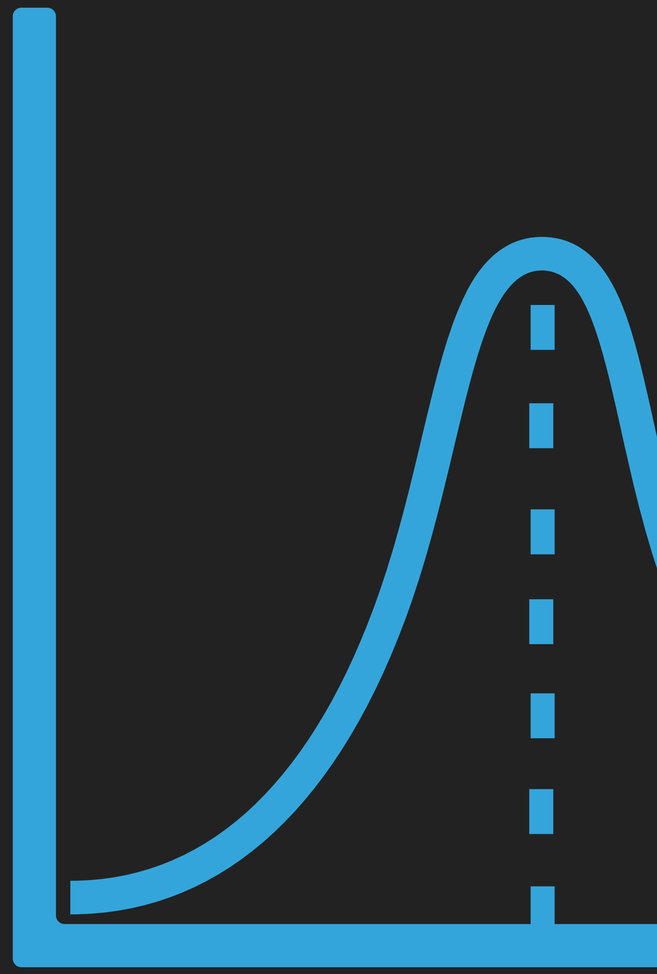
- Play around with some of the tuneable parameters (docs):
  - activation : {'identity', 'logistic', 'tanh', 'relu'}, default 'relu'

  - hidden_layer_sizes : tuple, length = n_layers - 2, default (100,)
    - The ith element represents the number of neurons in the ith hidden layer

  - alpha : float, optional, default 0.0001
    - L2 penalty (regularization term) parameter

Note: Perform better on normalised data

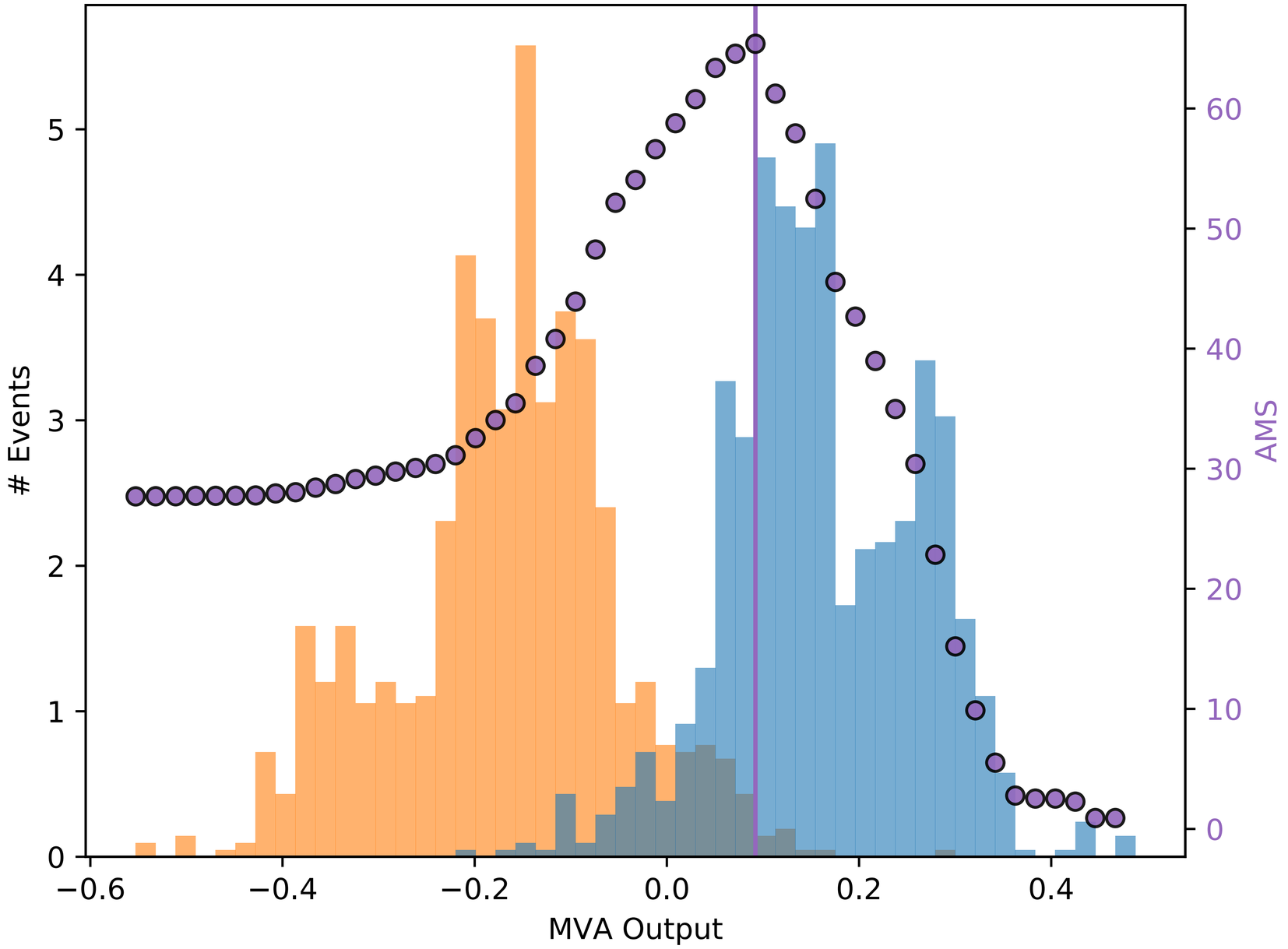
GENERAL TECHNIQUES



Using the MVA Output

- What do you do with the MVA output?
  - Use the output as input to fits
    - Thi
  - Can place a cut:
    - Fix a working point based on signal/background efficiency:
      - Predict places cut so 0.5 of signal is either side
      - ROC curve gives you idea of how this evolves
    - Optimise cut to maximise some metric
      - s/sqrt(b), AMS, etc.
      - see example next slide

Using the MVA Output



UNIVERSITY OF SUSSEX

Motivation

- Need confidence that the trained MVA is robust:
  - Performance on unseen samples accurately predicted.

- Validation techniques required for:
  - Model Selection:
    - Methods have at least one free parameter e.g.
      - BDT - #trees, min node size, etc.
      - SVM - kernel function, kernel parameters, cost, etc.
    - How are these parameters of models “optimally” selected?

  - Performance Estimation:
    - How does the chosen model perform?
    - Usually true error rate is used (misclassification rate for the entire dataset)

UNIVERSITY OF SUSSEX

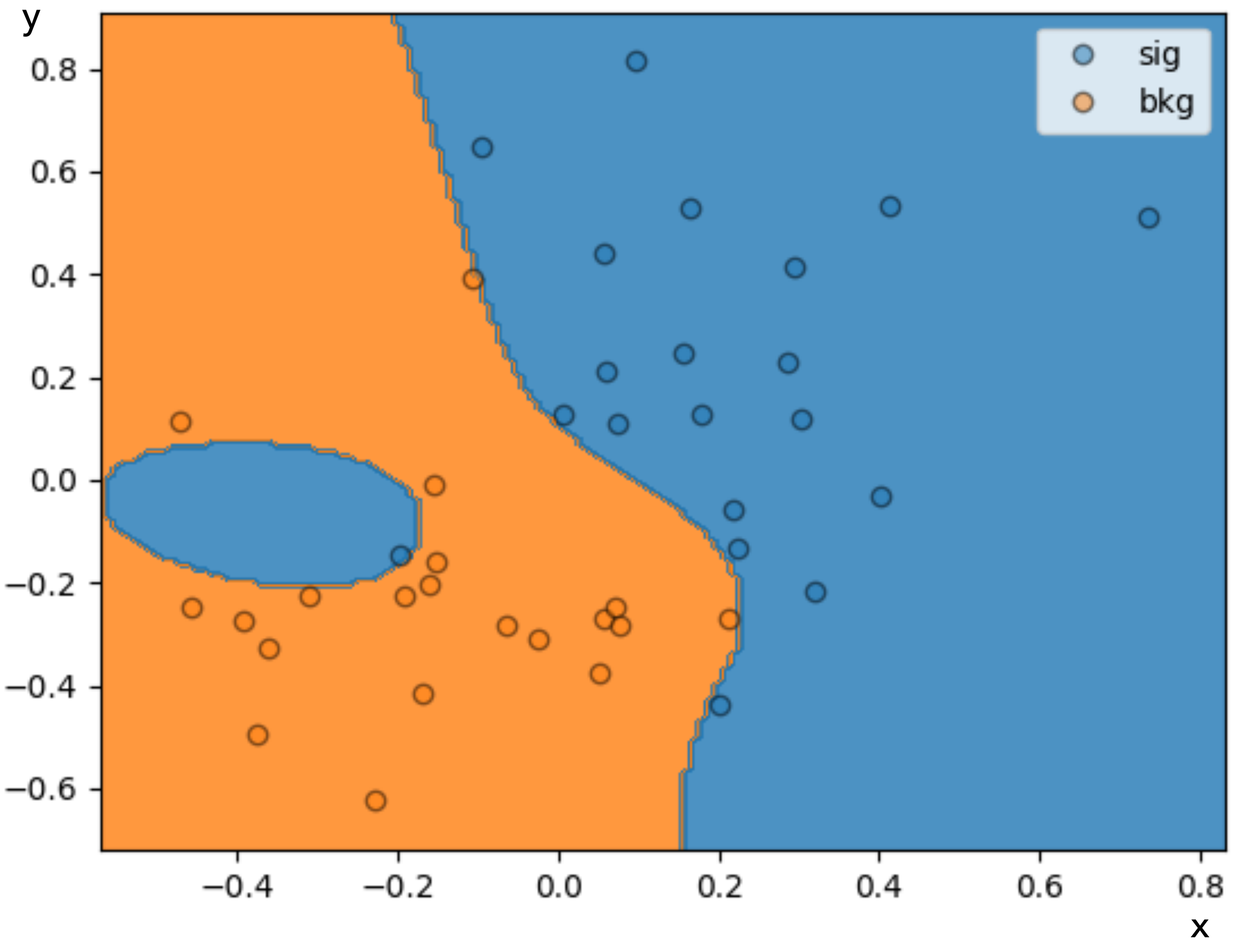
Motivation

- For an unlimited dataset these issues are trivial, iterate through parameters and find model with best value for figure of merit (FOM).
- In reality datasets are smaller than we would like.
- Naïvely use whole dataset to select and train classifier and to estimate error.
  - Leads to overfitting/overtraining as classifier learns fluctuations in the dataset and performs worse on unseen data.
  - Overfitting more distinct for classifiers with large number of tuneable parameters.
  - Also gives overly optimistic estimation of FOM.

Generalisation

The issue...

N=20

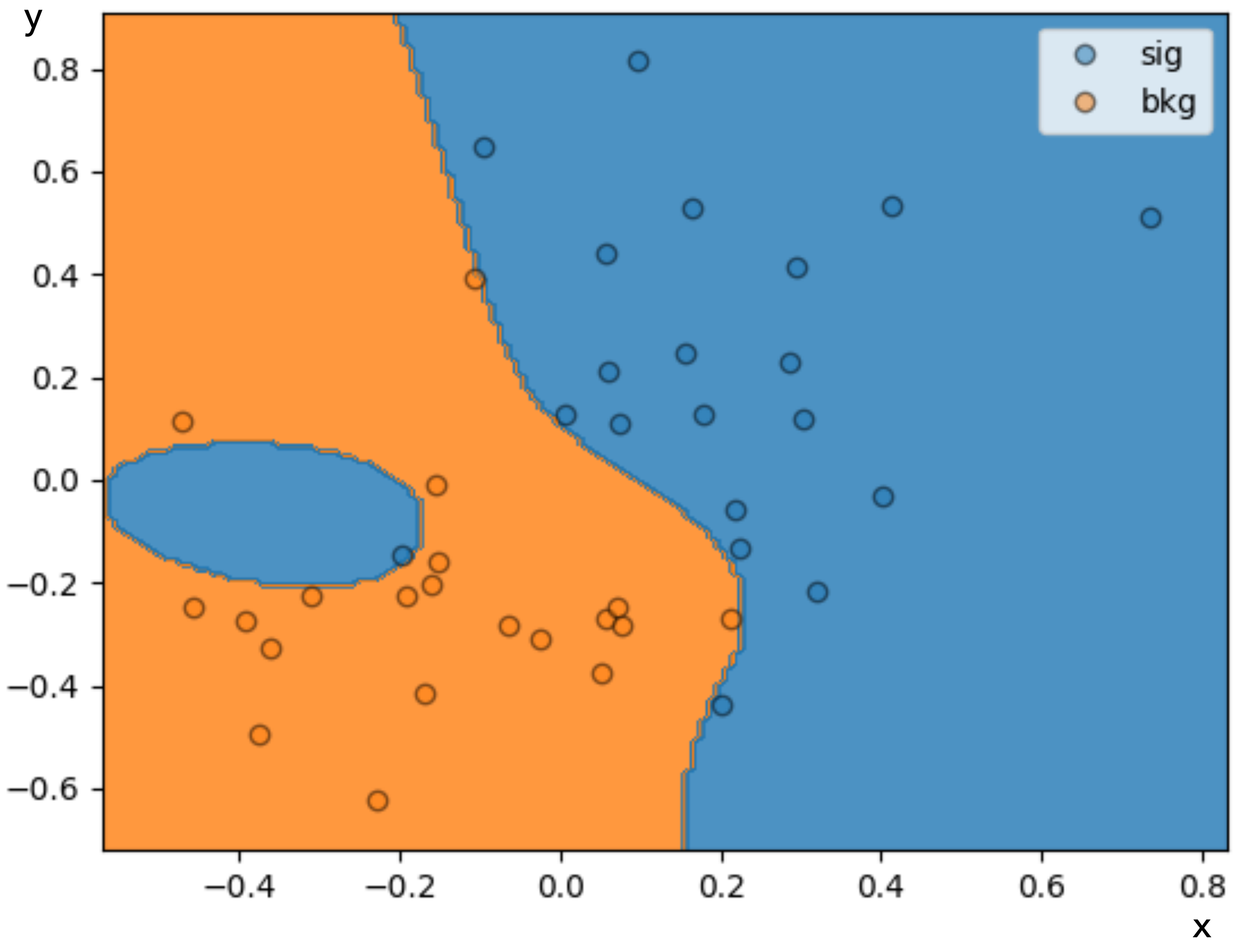


Generalisation

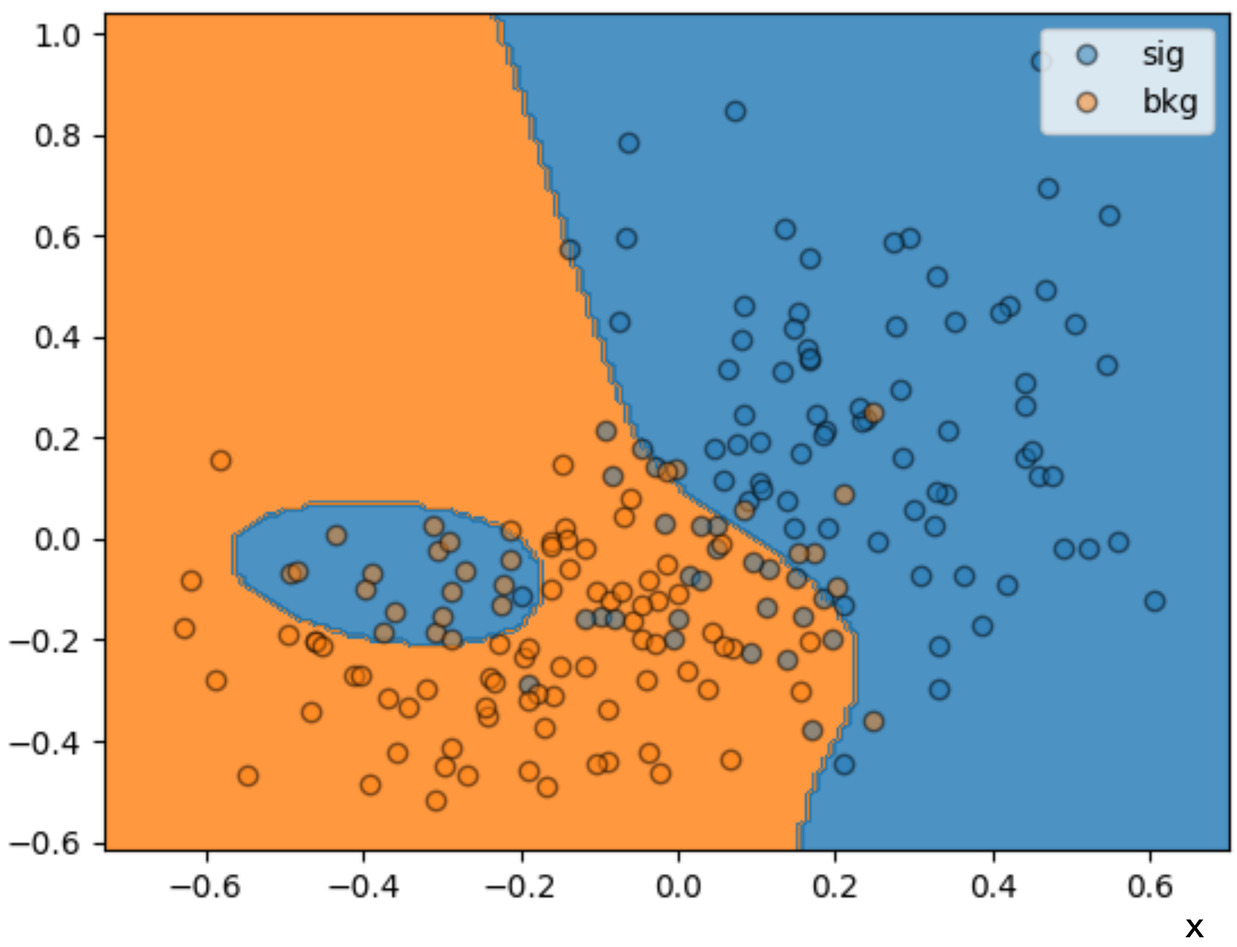
UNIVERSITY OF SUSSEX

The issue...

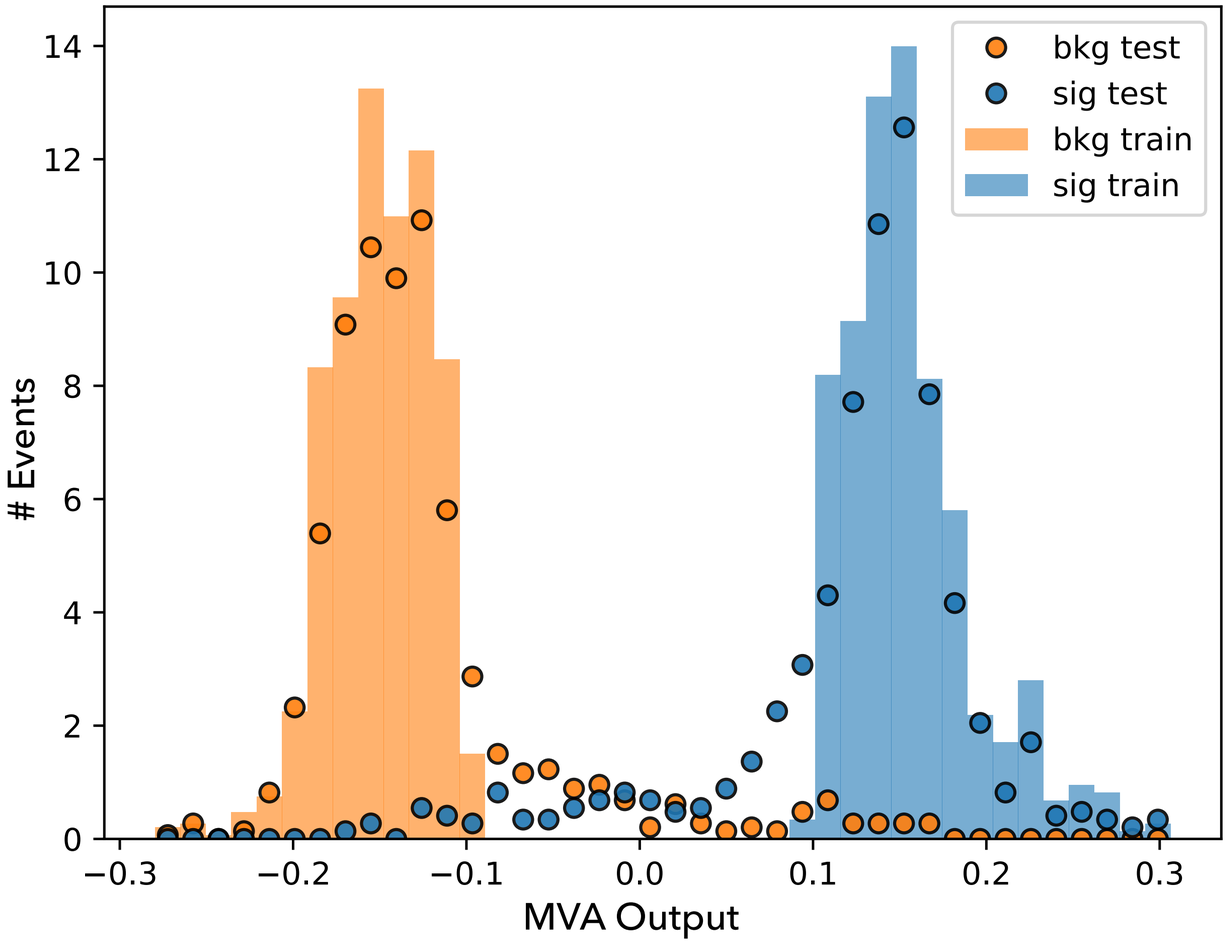
N=20



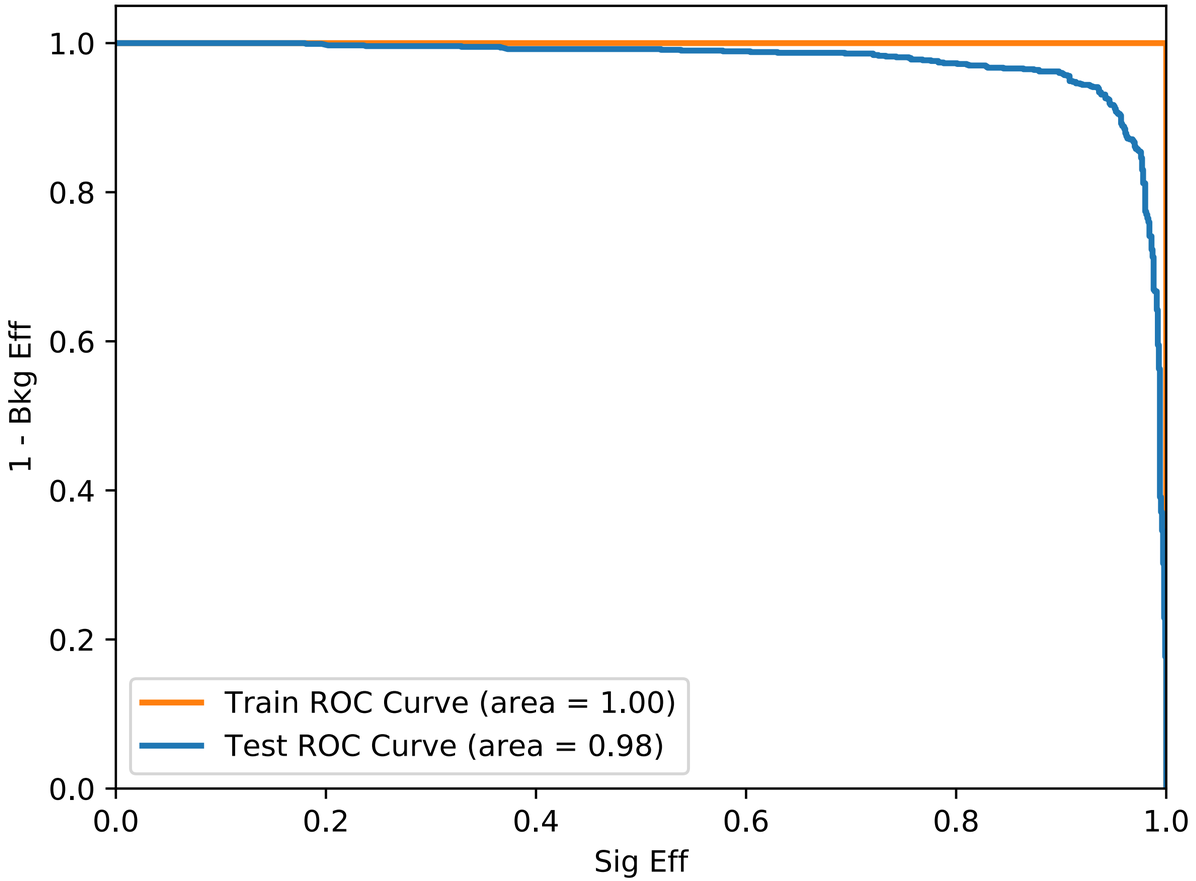
N=100



Generalisation



KSTestSig: pvalue=4.71e-05



UNIVERSITY OF SUSSEX

Dropout for Neural Networks

- Specific method to try to reduce overtraining in NNs

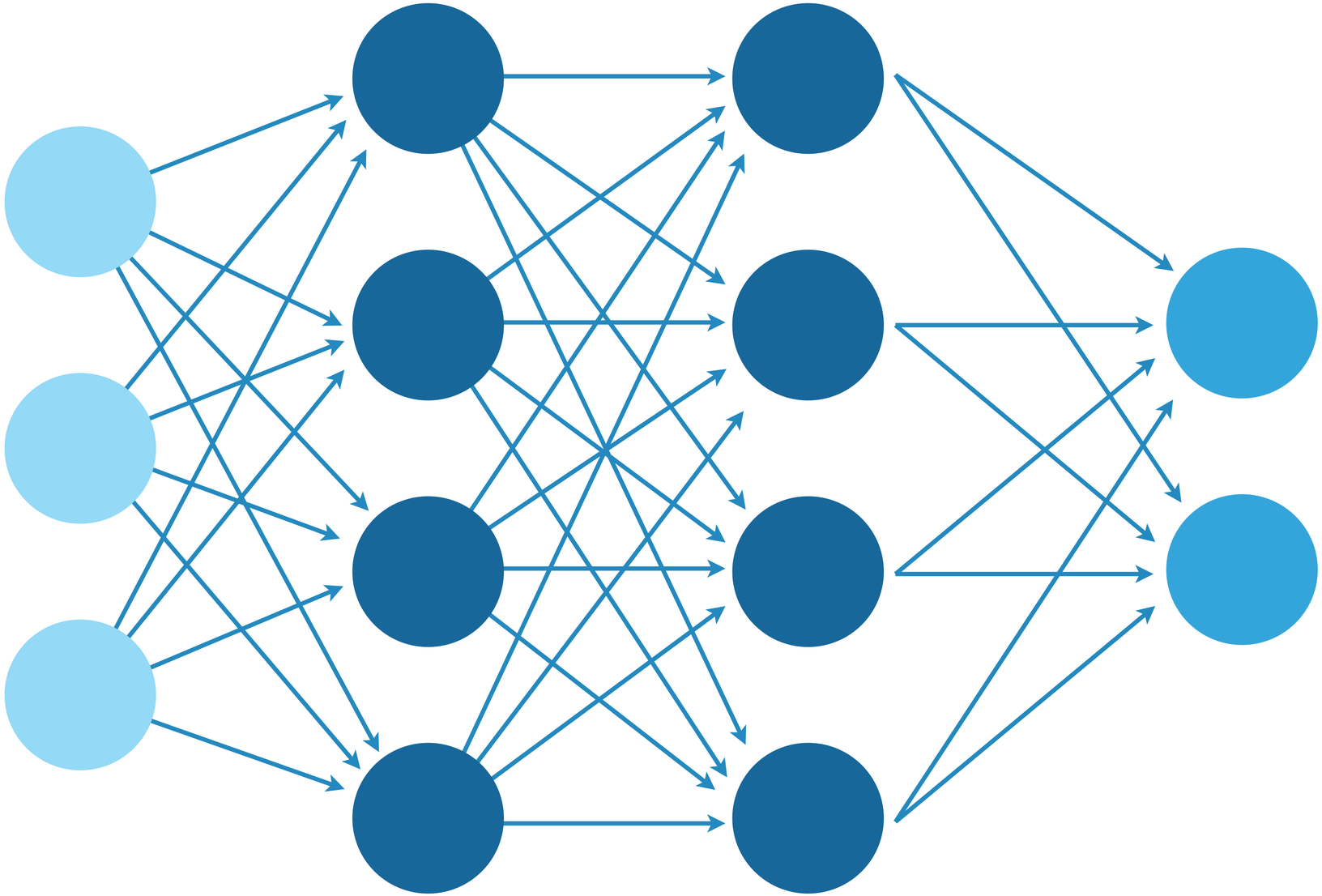
- Nodes are assigned a “survival” probability
- At each training step nodes are randomly dropped
  - Leaves a reduced network with a subset of the nodes
  - Edges connecting nodes are also removed
- For evaluation/testing use all nodes but weights by their probability

- Dropout will decrease time taken per training step but increase overall number of steps required to converge

- NN learns more robust features that exist across a larger number of the training batches

Generalisation

Dropout for Neural Networks

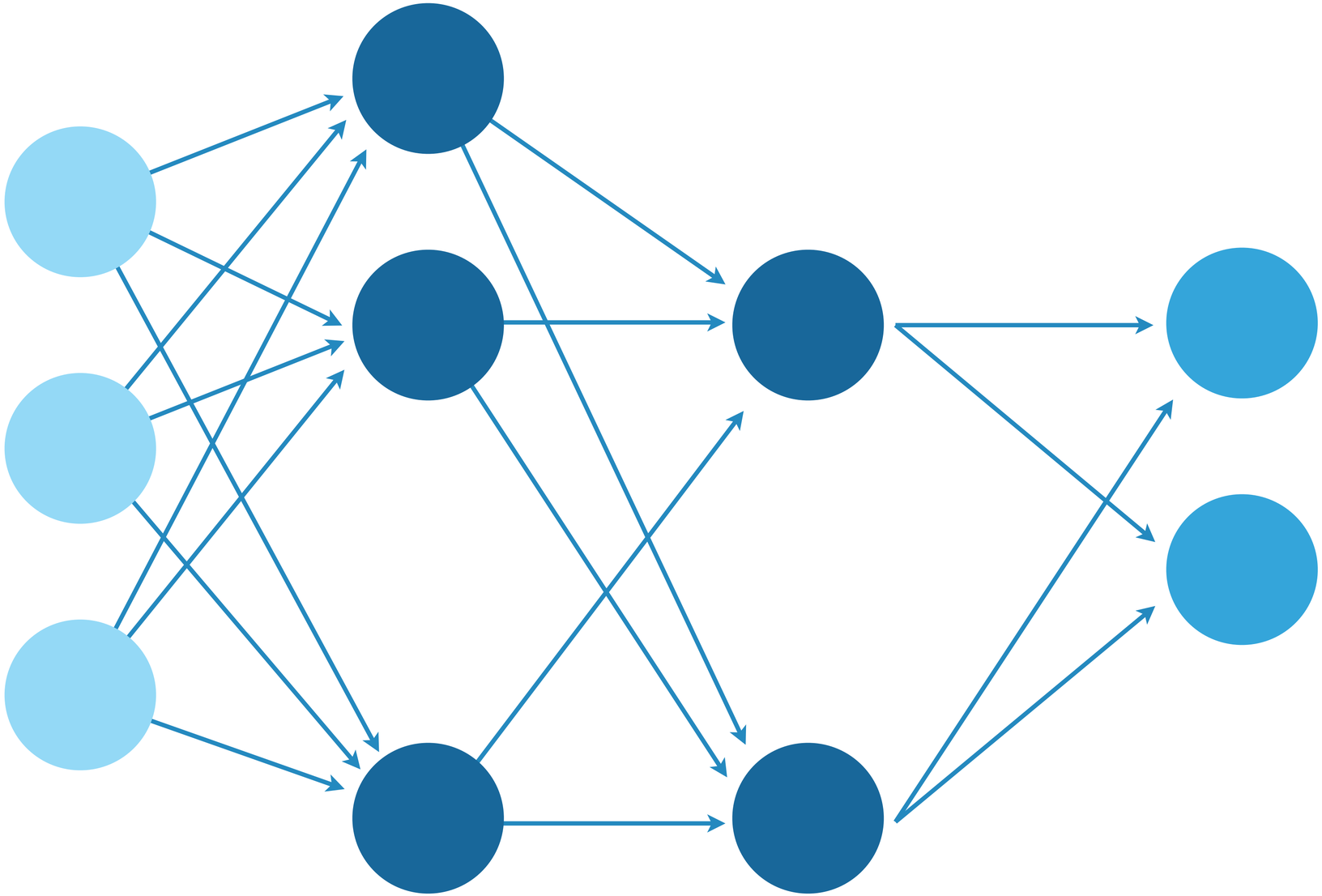


UNIVERSITY OF SUSSEX

Dropout for Neural Networks

Training Epoch n

Randomly drop nodes with prob = 1 - p

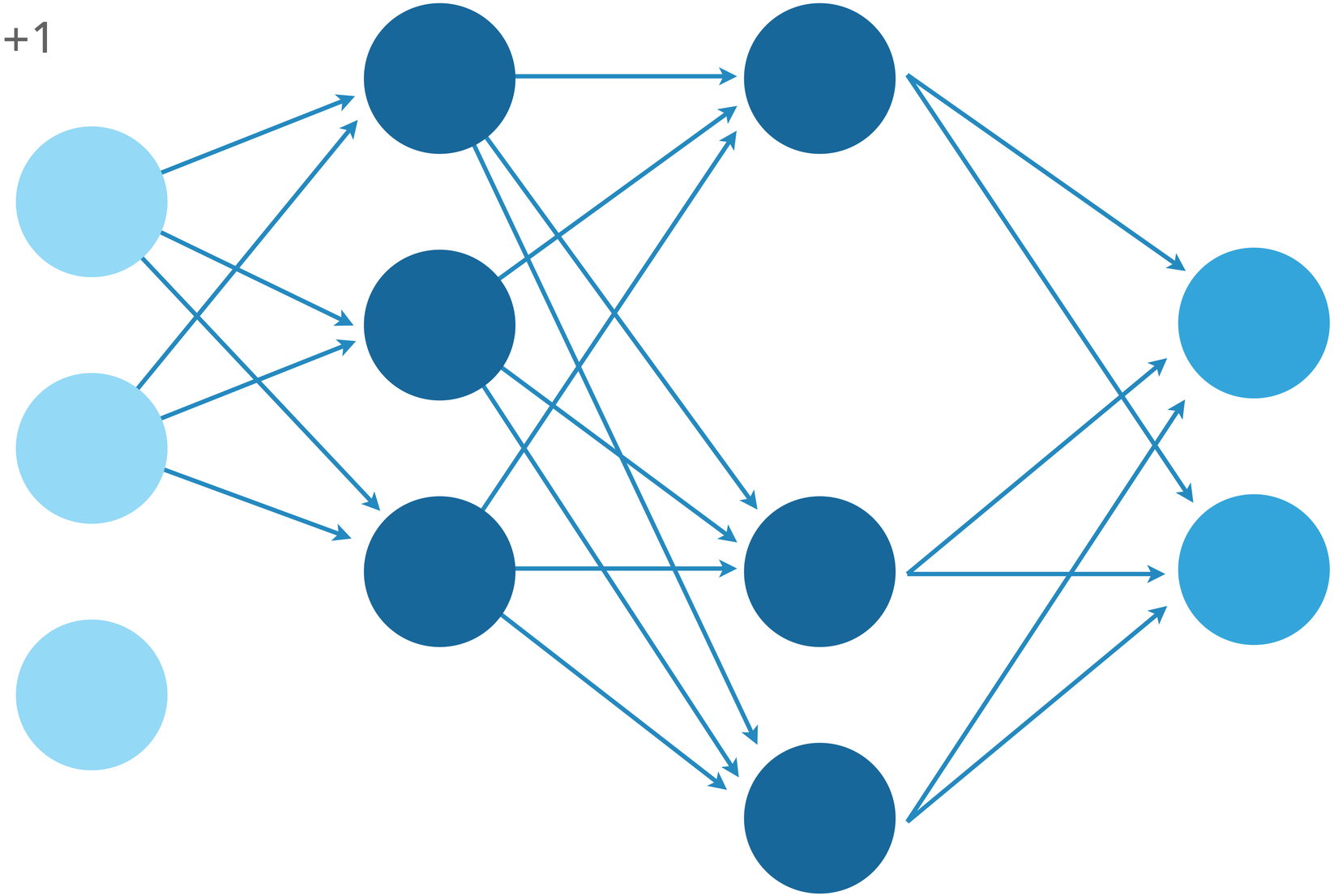


UNIVERSITY OF SUSSEX

Dropout for Neural Networks

Training Epoch n+1

Randomly drop nodes with prob = 1 - p

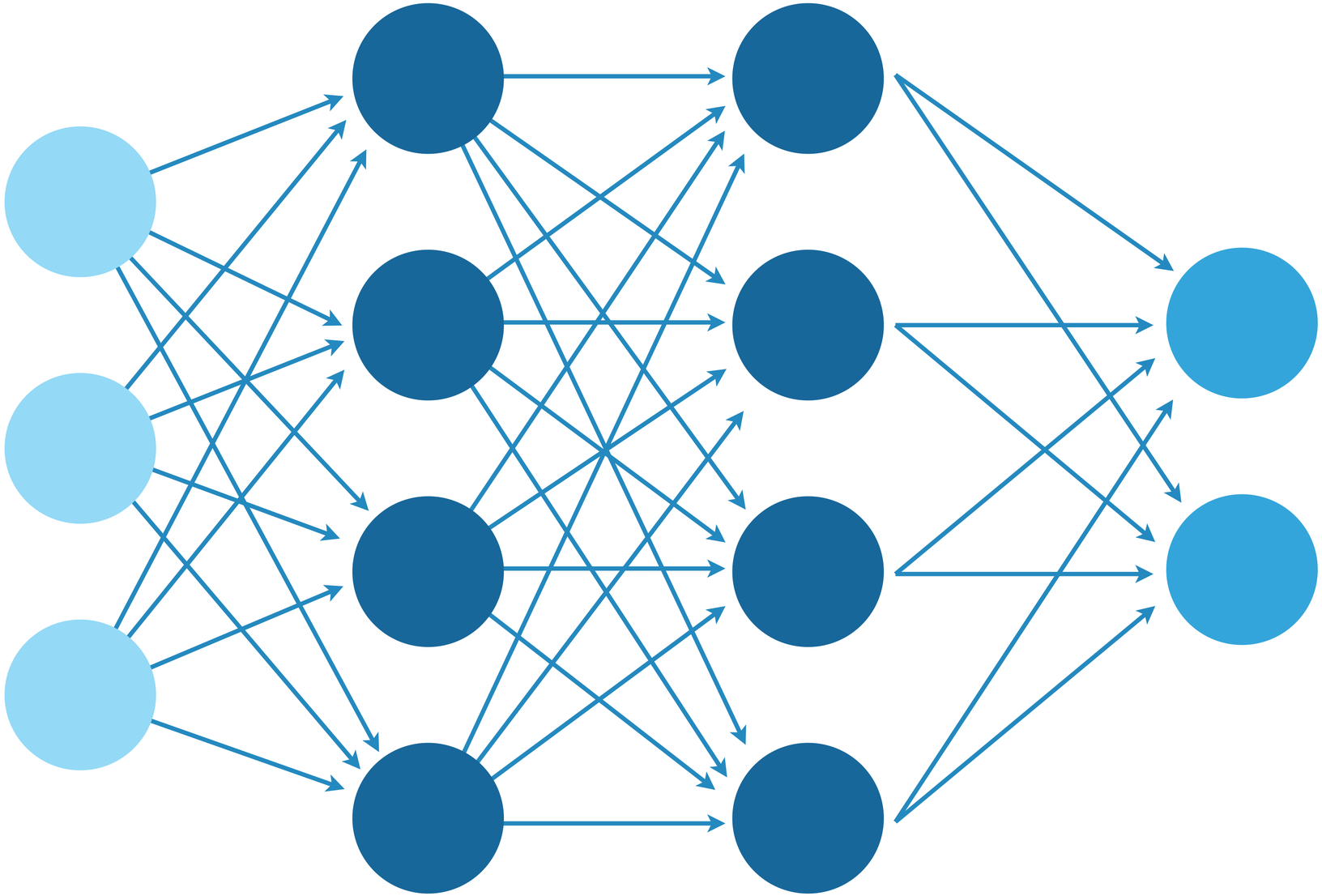


UNIVERSITY OF SUSSEX

Dropout for Neural Networks

Evaluation

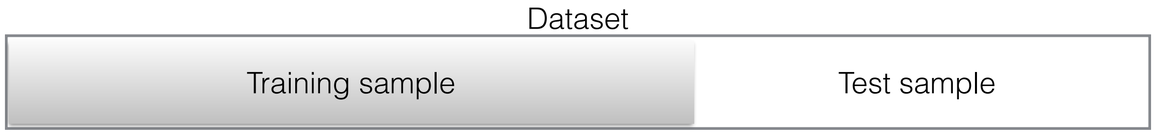
Use all nodes but weight with p



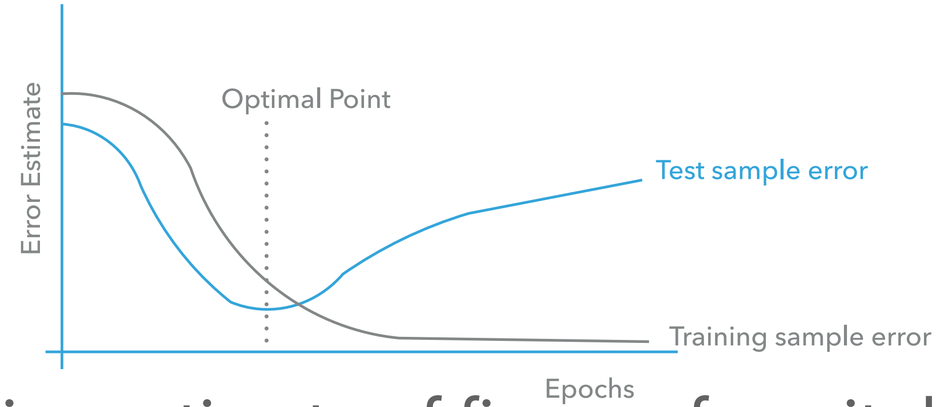
UNIVERSITY OF SUSSEX

Hold-Out Validation

- Potential way to overcome these issues is use hold-out technique, splitting the dataset into training and test subsamples.



- Can use these datasets to select "optimal" parameters, for example back-propagation for MLP.

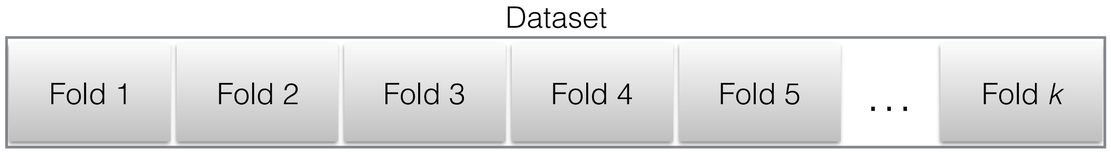


- Can give misleading estimate of figure of merit depending on how the data is split.

UNIVERSITY OF SUSSEX

k-Fold Cross-Validation

- May not be able to reserve a large portion of data for testing.
- Use k-fold cross-validation:



  - Split dataset into k randomly sampled independent subsets (folds).
  - Train classifier with k-1 folds and test with remaining fold.
  - Repeat k times.

- Advantage of using the whole dataset for testing and training.
- True value of FOM is then estimated using average.

UNIVERSITY OF SUSSEX

k-Fold Cross-Validation

- How many folds???

- Large number of folds:
  - Good estimate of average error rate (bias of the estimator is small).
  - Variance of the estimator is large.
  - Computational time is long.

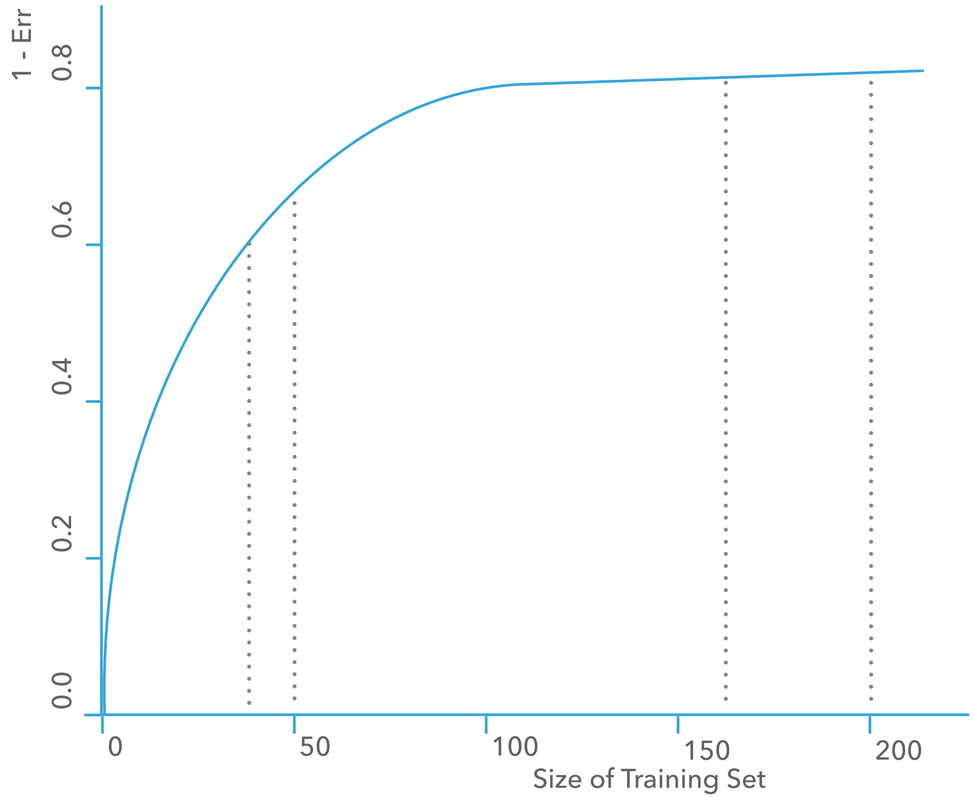
- Small number of folds:
  - Poor estimate of average error rate (bias of the estimator is large).
  - Variance of the estimator is small.
  - Computational time is relatively short.

- In reality choice is motivated by the size of the dataset, i.e. sparse dataset need extreme of leave-one-out method to train on as much data as possible.

UNIVERSITY OF SUSSEX

k-Fold Cross-Validation

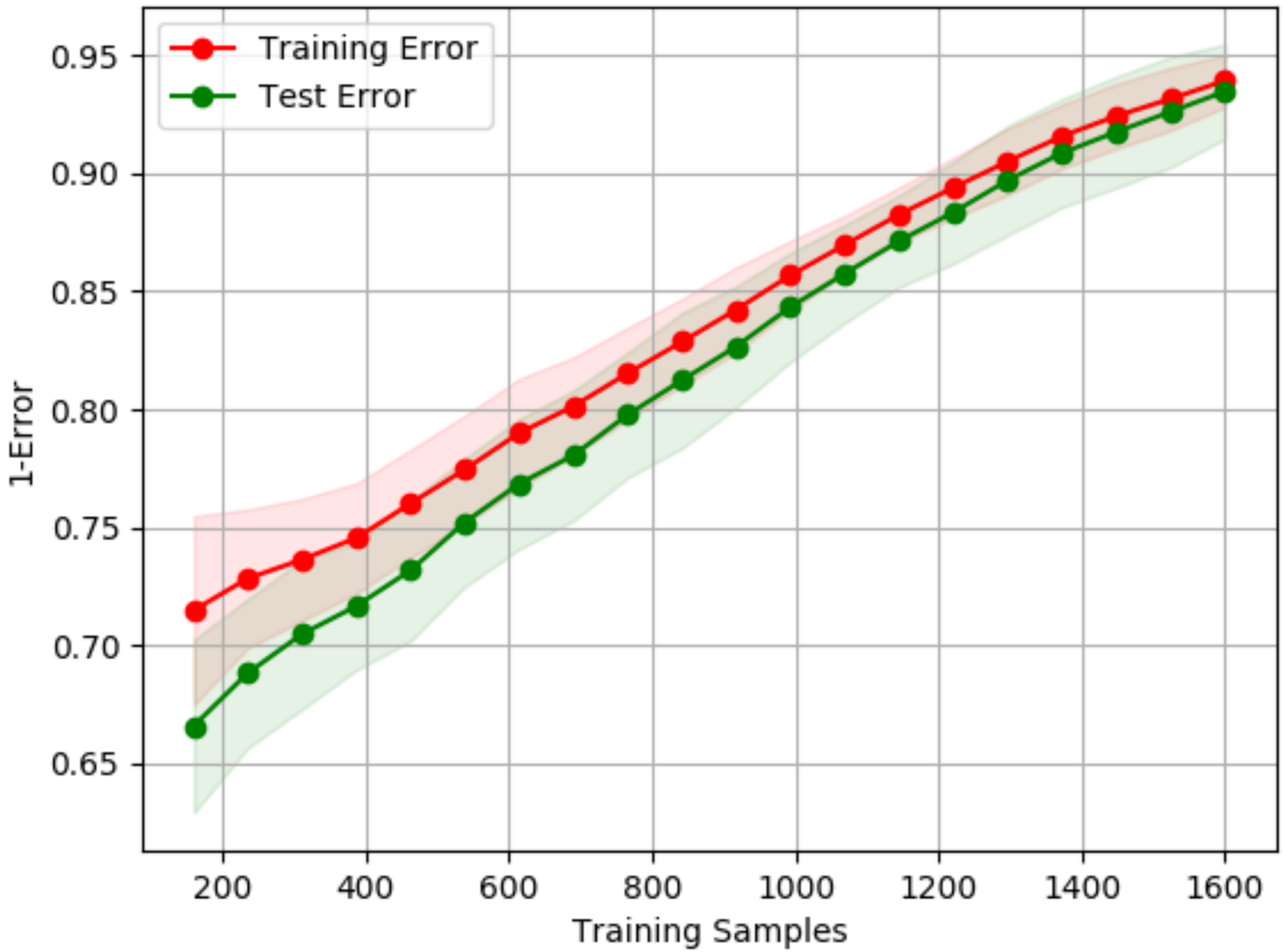
- Hypothetical example:
  - For sample size of 200, 5 fold CV will estimate the error with similar performance on training set of 160 to that of the full sample.
  - However for sample of 50, 5 fold CV will give a larger error than not using CV.



UNIVERSITY OF SUSSEX

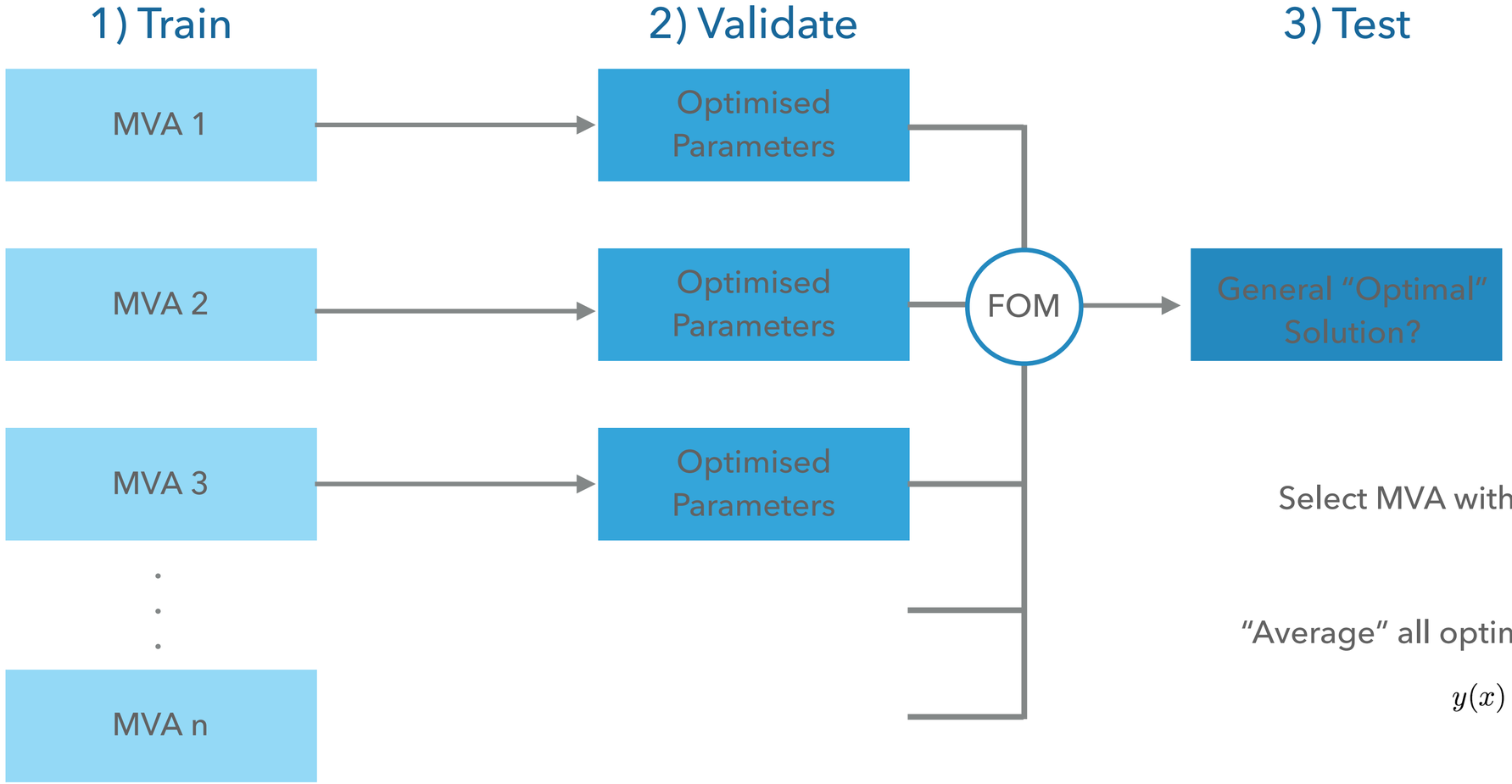
k-Fold Cross-Validation

- Hypothetical example:
  - For sample size of 200, 5 fold CV will estimate the error with similar performance on training set of 160 to that of the full sample.
  - However for sample of 50, 5 fold CV will give a larger error than not using CV.



Generalisation

- Ideally 3 statistically independent datasets.



Select MVA with best validation FOM?

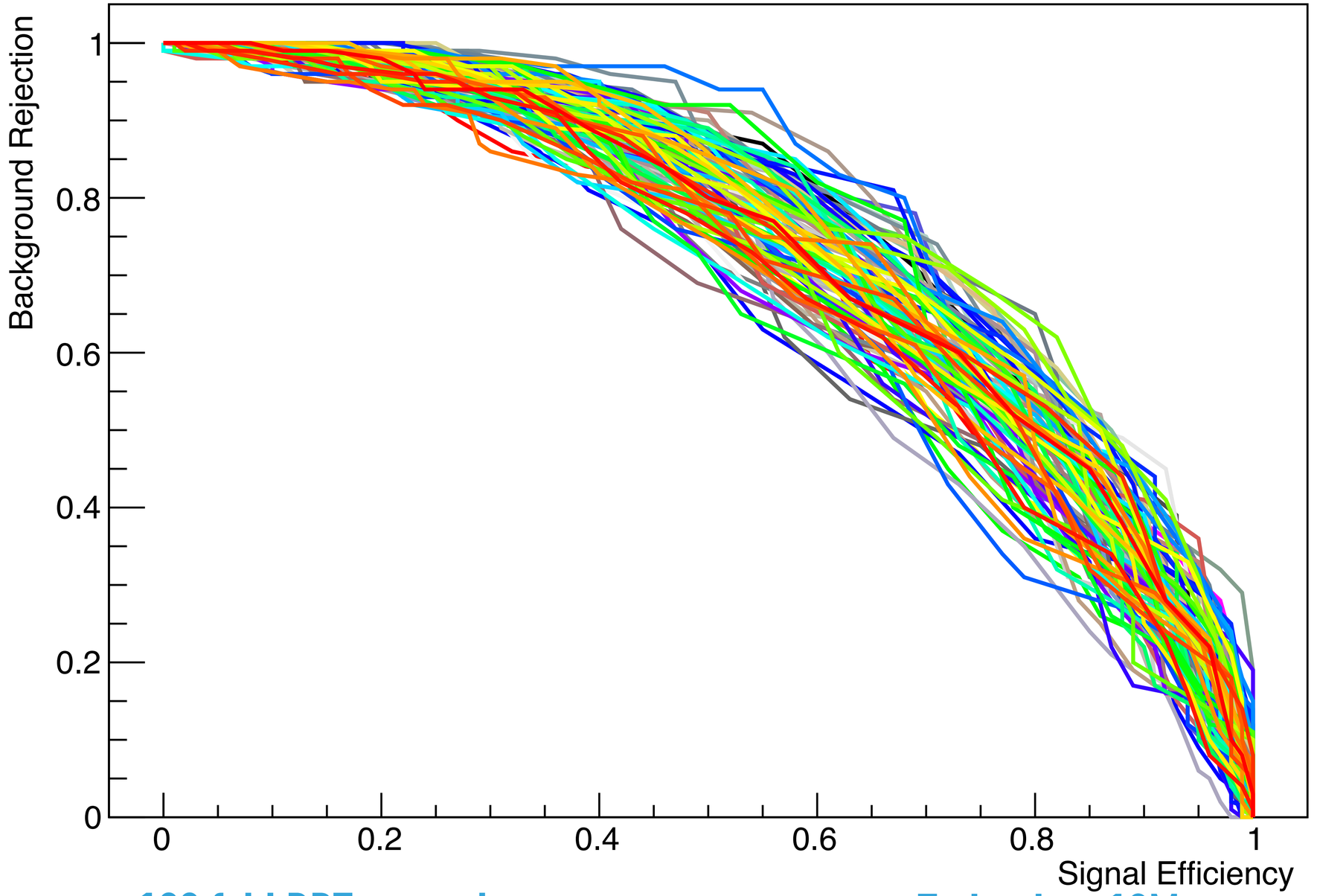
or

"Average" all optimised MVA trainings?, i.e.

$$y(x) = \frac{1}{n}\sum_{i=1}^{n} y_i(x)$$

- "Best" performing MVA doesn't necessarily give the desired output.
- Take aggregated output of final trained MVAs on test sample in some form of average.

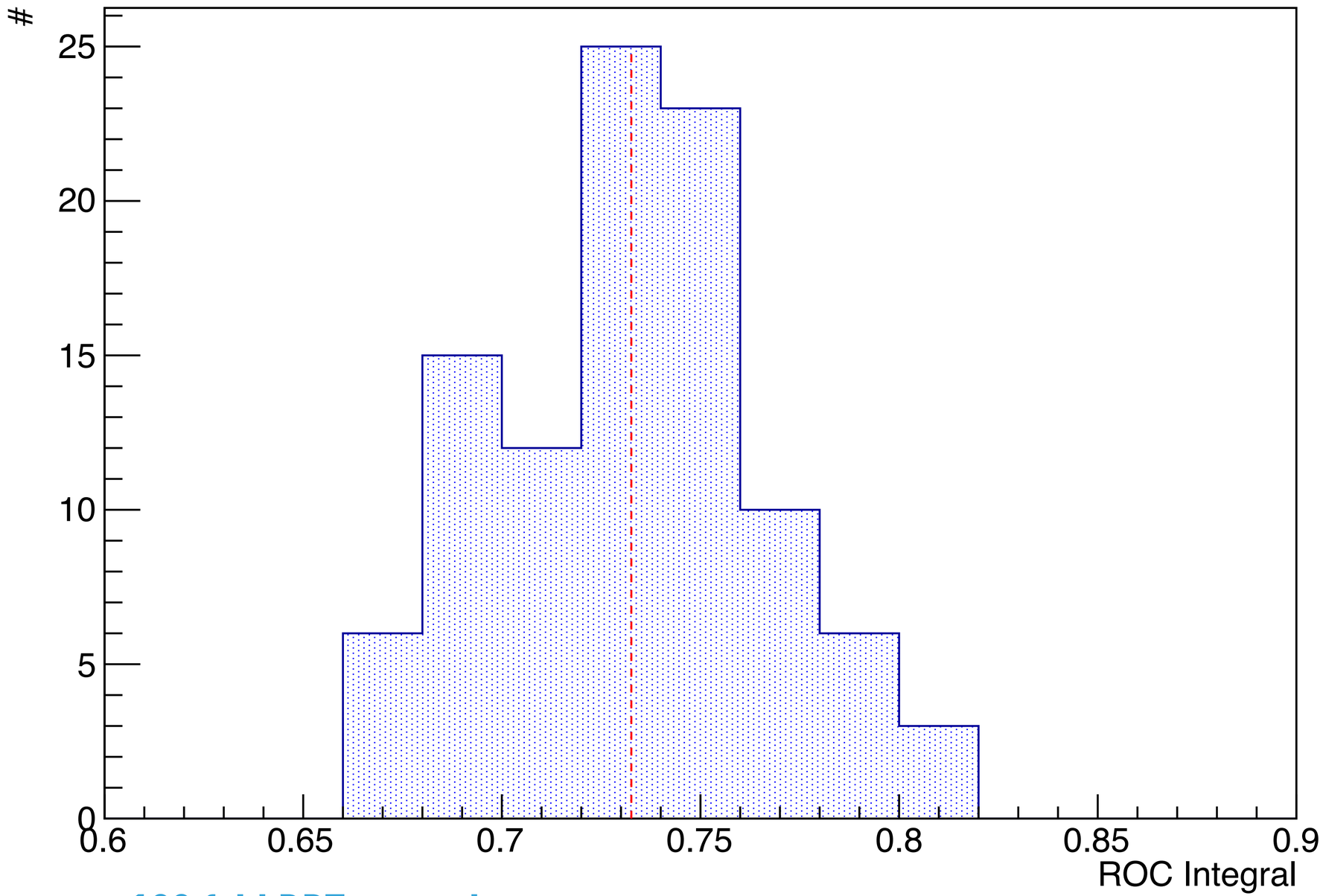
Generalisation



100 fold BDT example

Trained on 10M events

Generalisation



100 fold BDT example

General Techniques

UNIVERSITY OF SUSSEX

- Notebook: GeneralTechniques.ipynb

Committee Machines

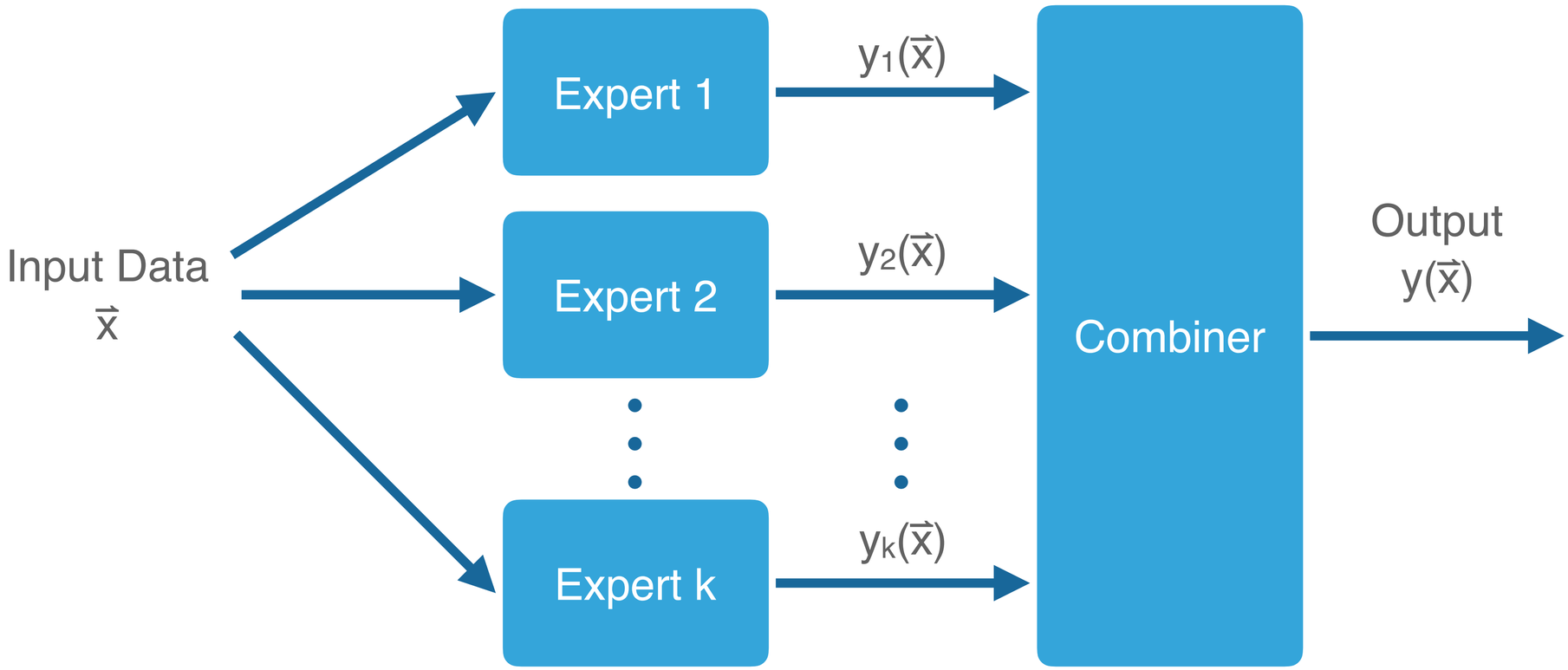
- Normal workflow involves keeping best method (based on performance of independent validation set):
  - Two main disadvantages:
    - Train many different candidates which are ultimately thrown away
    - Best performing on validation set may not have best performance on test set

- Can be overcome by combining methods to form a committee machine
  - Can perform better than single method
  - Little additional computational effort
  - Attempting to reduce variance at little/no cost to bias

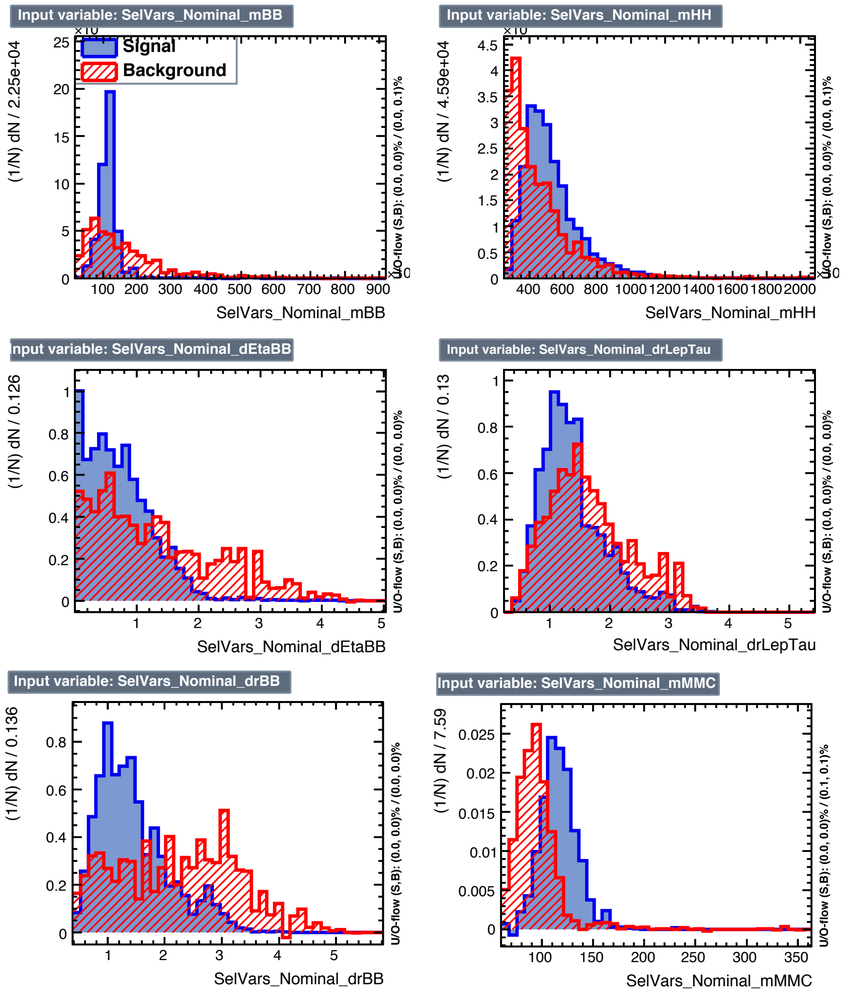
Committee Machines

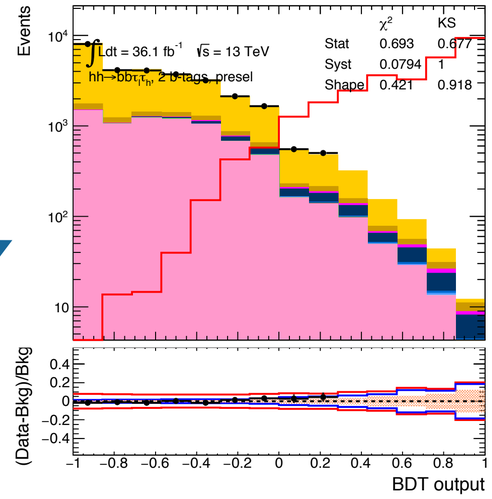
- Types:
  - Ensemble averaging
    - Combine outputs of different experts through linear combination
  - Boosting
    - Weak learners combined to make stronger one
  - Mixture of experts
    - Individual experts outputs non-linearly combined using gating networks

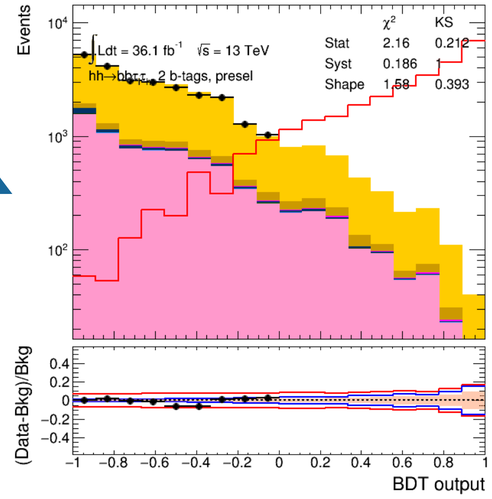
Committee Machines

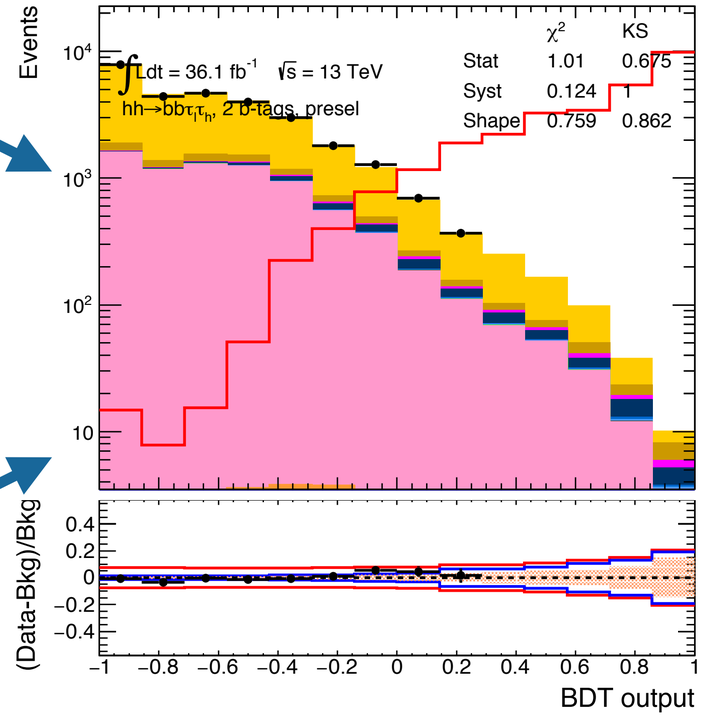


Committee Machines

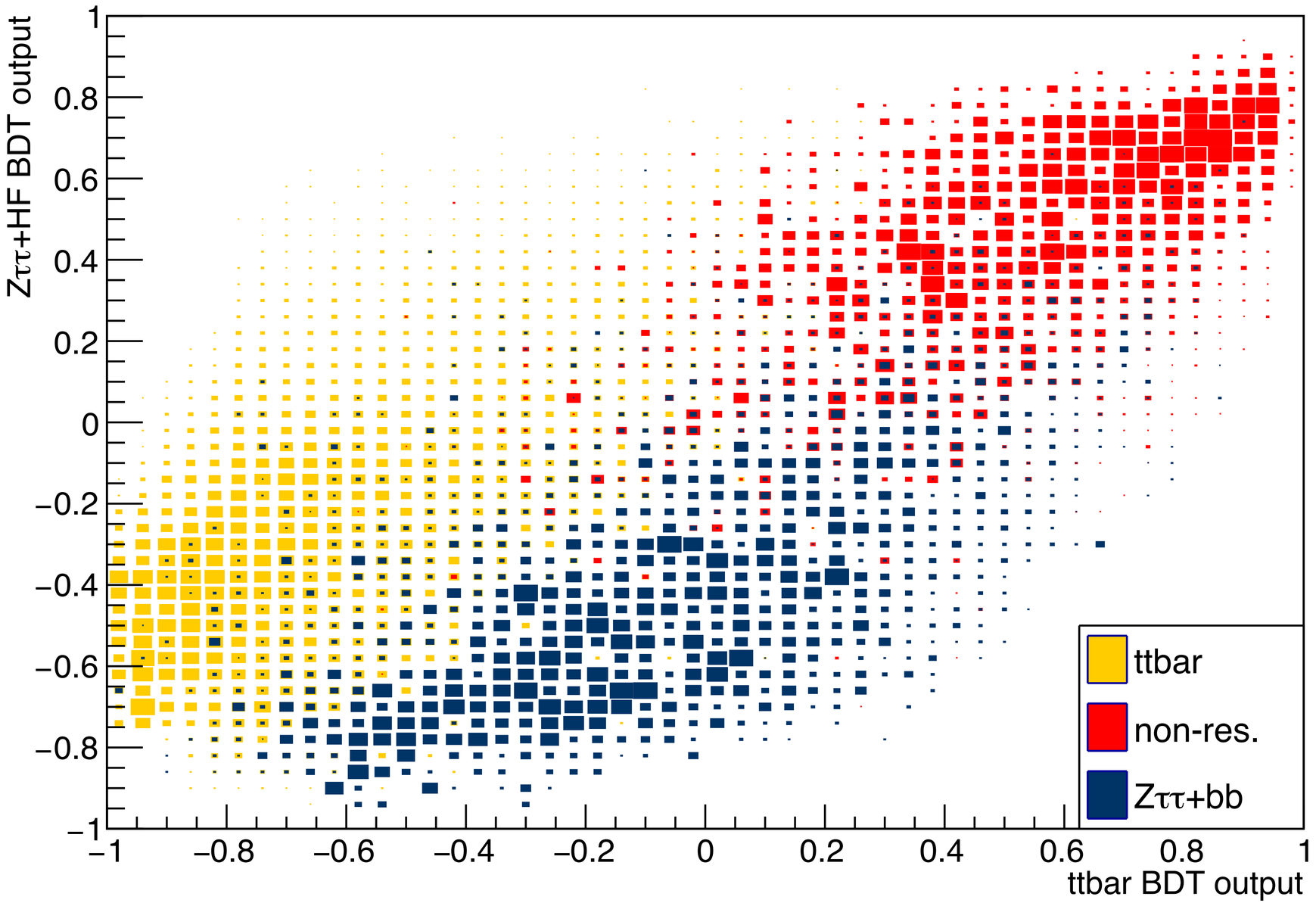








Committee Machines



Summary

UNIVERSITY OF SUSSEX

- Supervised classification techniques used to learn mapping from input variables to output labels
- Covered basics of some popular algorithms
  - Very large and constantly developing field
- Examples to get you started
  - Many tools available for tackling problems

THANKS!!!

and

Questions?