Queen Mary
University of London

DR ADRIAN BEVAN

MULTIVARIATE ANALYSIS AND ITS USE IN HIGH ENERGY PHYSICS

2) DECISION TREES

Lectures given at the department of Physics at CINVESTAV, Instituto Politécnico Nacional, Mexico City
28th August - 3rd Sept 2018

LECTURE PLAN

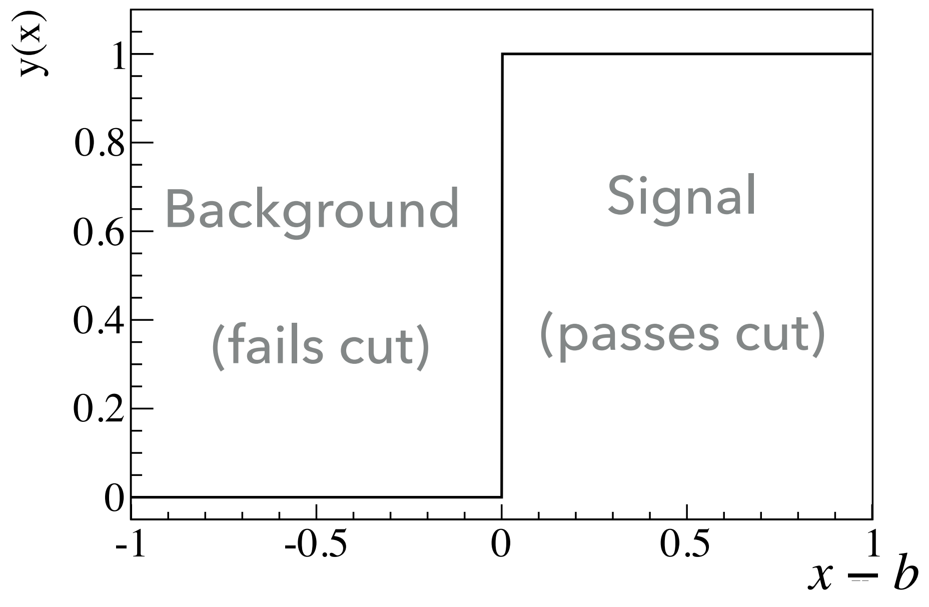
- Introduction
- Decision Trees
  - Boosting
  - Bagging
- Random Forests
- Examples
- Decision trees (reflection)
- Summary
- Suggested reading

INTRODUCTION

- The cut based and linear discriminant analysis methods are limited.
- The underlying concepts of applying Heavyside function constraints on data selection and on the use of a decision boundary definition (a plane in hyperspace) of the form of the dot product $\alpha^T x + \beta$ can be applied in more complicated algorithms.
- Here we consider extension to the concept of rectangular cuts to decision trees as a machine learning algorithm.
  - We will have to introduce the concepts of classification and regression; and methods to mitigate mis-classification of data.
  - The issue of overtraining is something we will come back to when discussing optimisation.

DECISION TREES

- Consider a data sample with an N dimensional input feature space X.
- X can be populated by examples from two or more different species of event (also called classes, categories or types).
- Consider the two types and call them signal and background, respectively*.
- We can use a Heavyside function to divide the data into two parts:
  - We can use this to distinguish between regions populated signal and background:



For an arbitrary cut position in x we can modify the Heavyside function

$$H(x) = \frac{1}{2}(1 + sign(x - b))$$

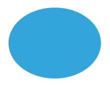
where b is the offset (bias) from zero.

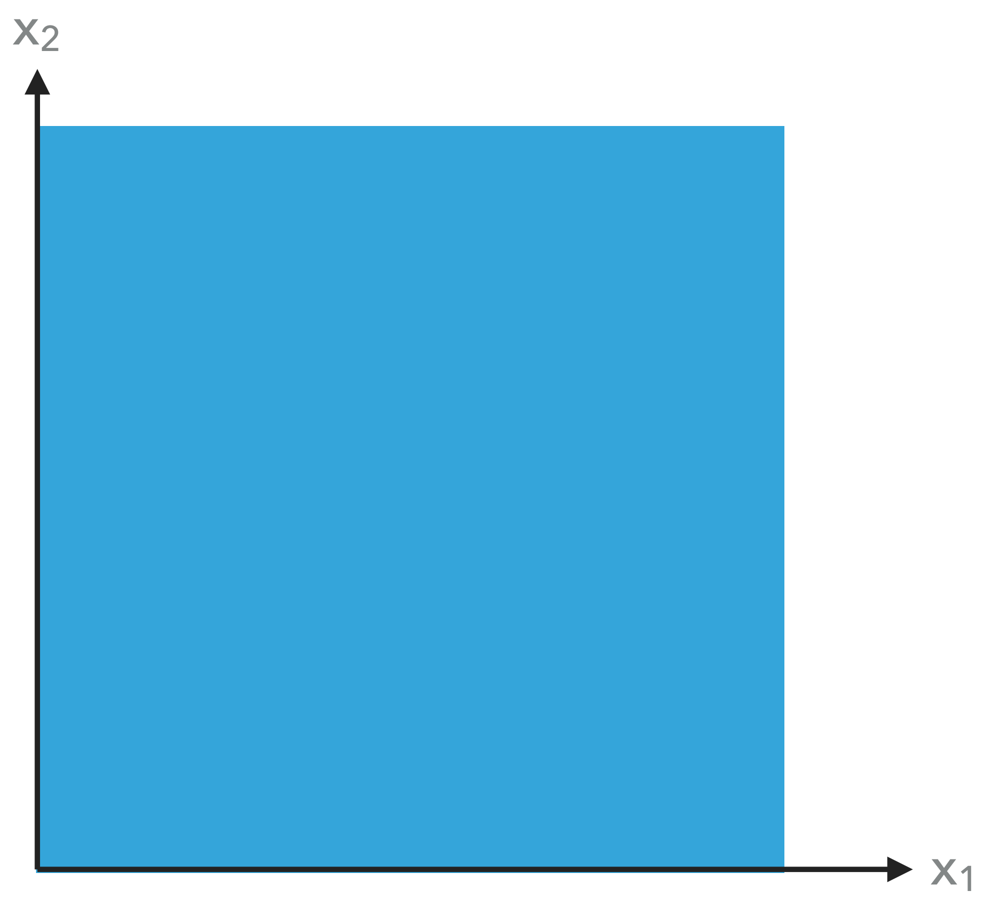
*Can generalise the problem to an arbitrary number of types.

DECISION TREES

- Decision trees divide the data feature space into a set of hypercubes that are classified as signal (+1) or background (-1) like.
    - Each region can be fitted with a constant to represent the data in that region.
    - We can recursively continue to sub-divide the data until some minimum number of examples are left in each sub-division.

Can describe the data as the root node.



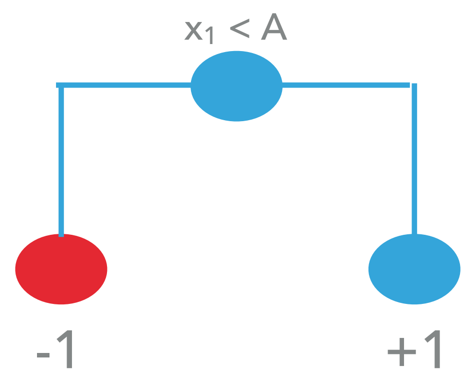


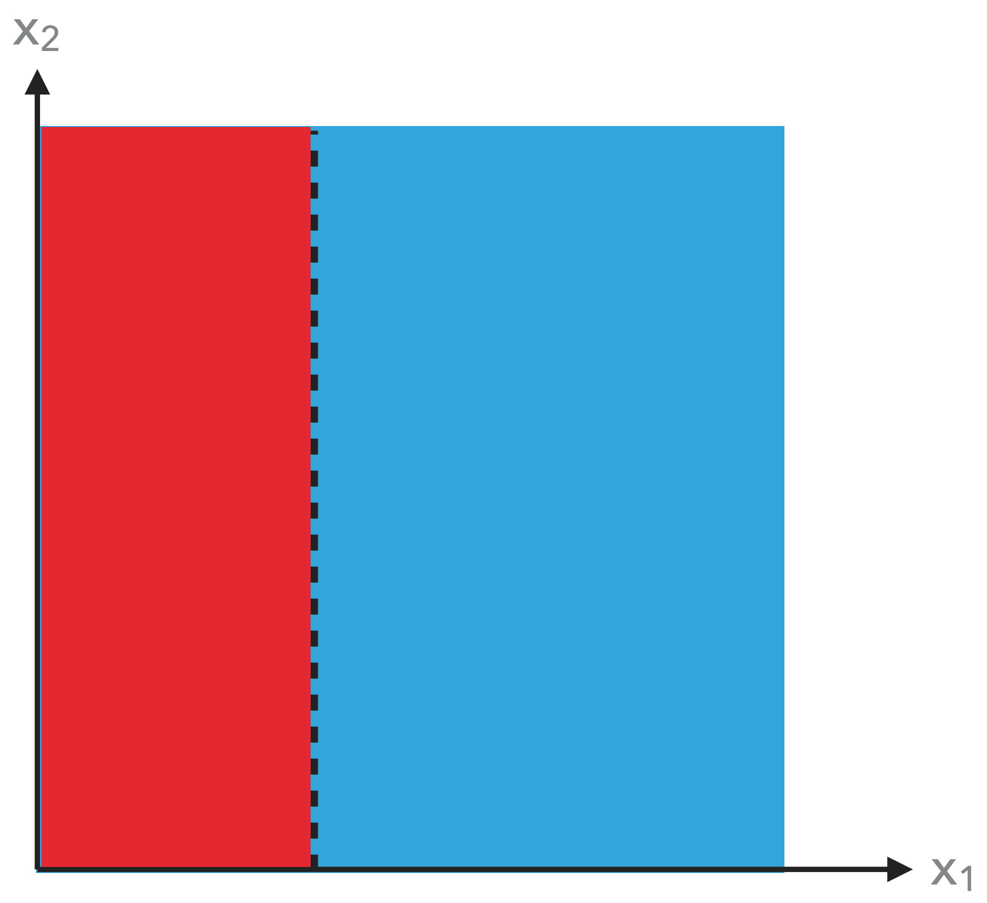
Example feature space described by $X=\{x_1, x_2\}$

DECISION TREES

- Decision trees divide the data feature space into a set of hypercubes that are classified as signal (+1) or background (-1) like.
  - Each region can be fitted with a constant to represent the data in that region.
  - We can recursively continue to sub-divide the data until some minimum number of examples are left in each sub-division.

The data get divided into two partitions.



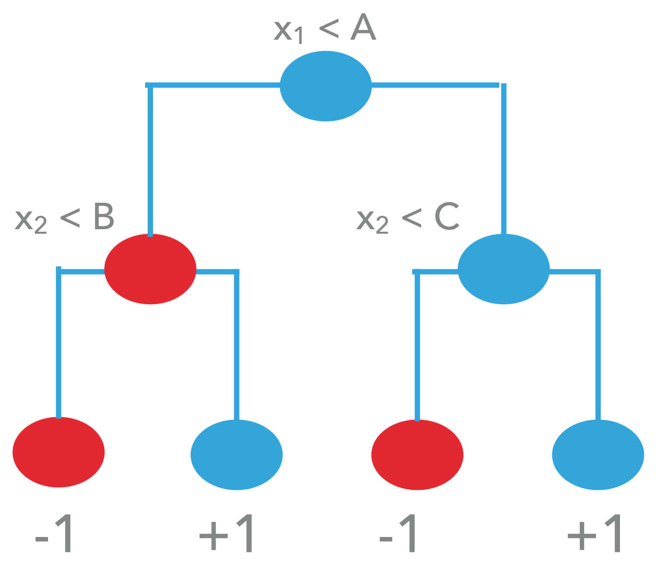


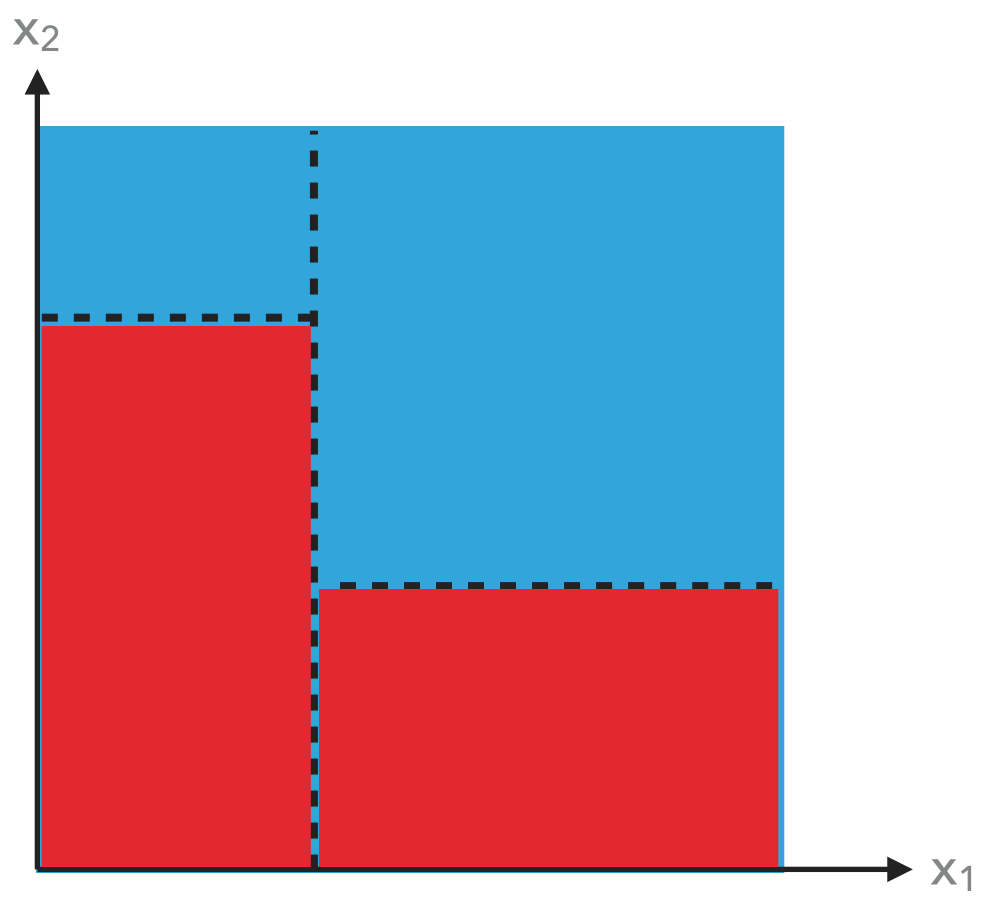
Cut on the feature space to separate the data into two different regions.

DECISION TREES

- Decision trees divide the data feature space into a set of hypercubes that are classified as signal (+1) or background (-1) like.
  - Each region can be fitted with a constant to represent the data in that region.
  - We can recursively continue to sub-divide the data until some minimum number of examples are left in each sub-division.

Divide the data again



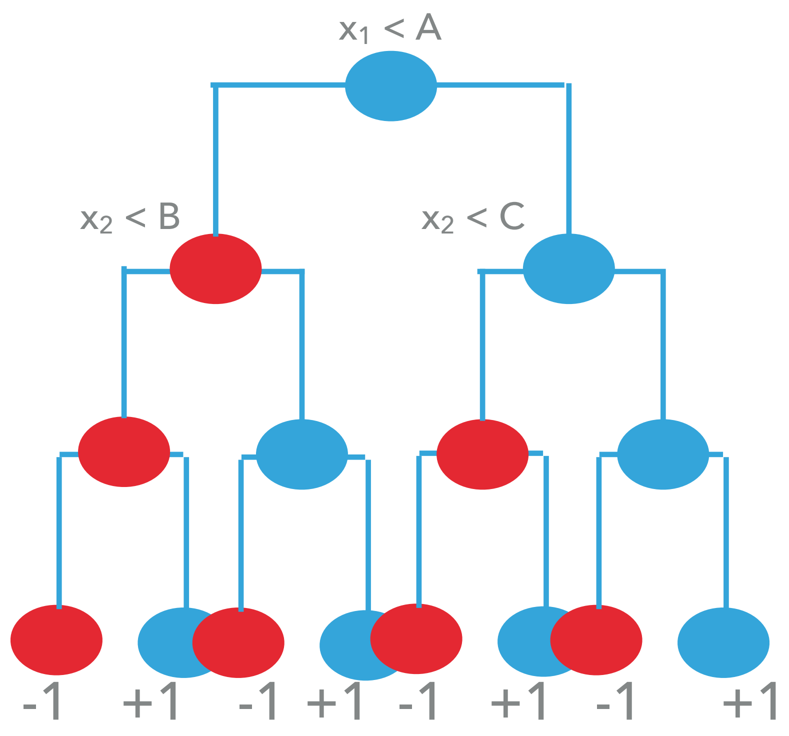


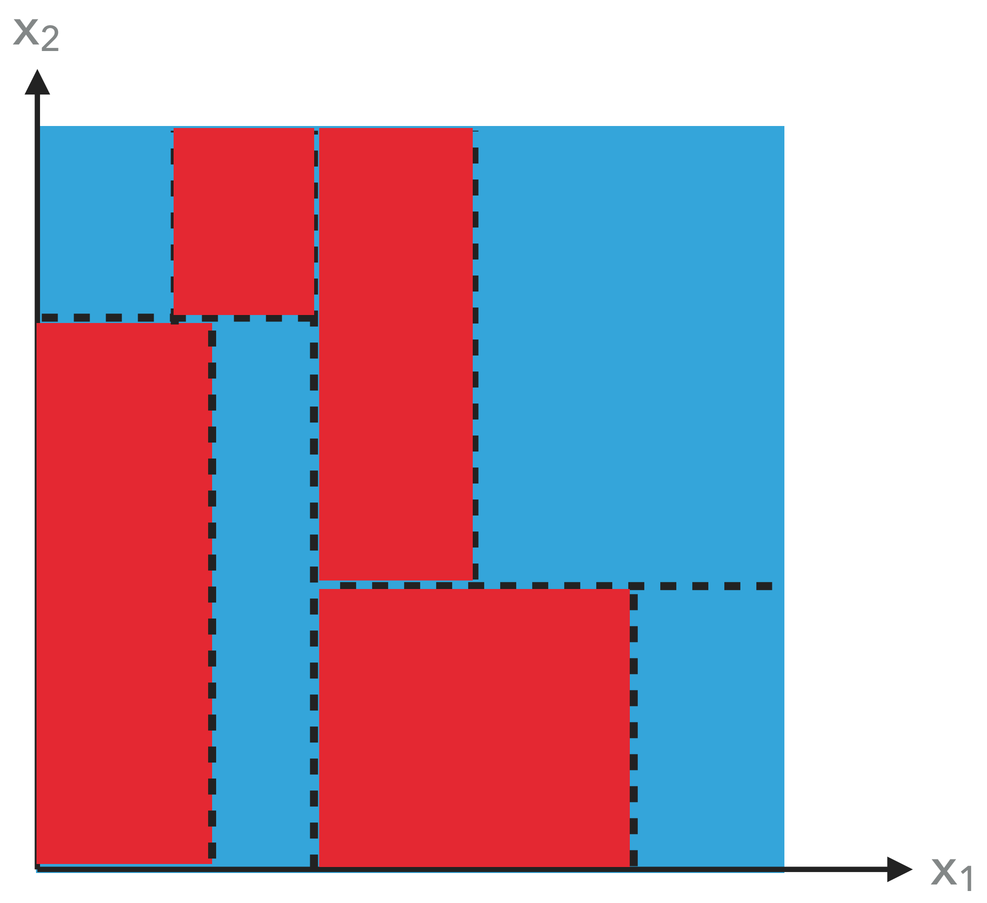
The feature space gets further sub-divided.

DECISION TREES

- Decision trees divide the data feature space into a set of hypercubes that are classified as signal (+1) or background (-1) like.
  - Each region can be fitted with a constant to represent the data in that region.
  - We can recursively continue to sub-divide the data until some minimum number of examples are left in each sub-division.

Divide the data again

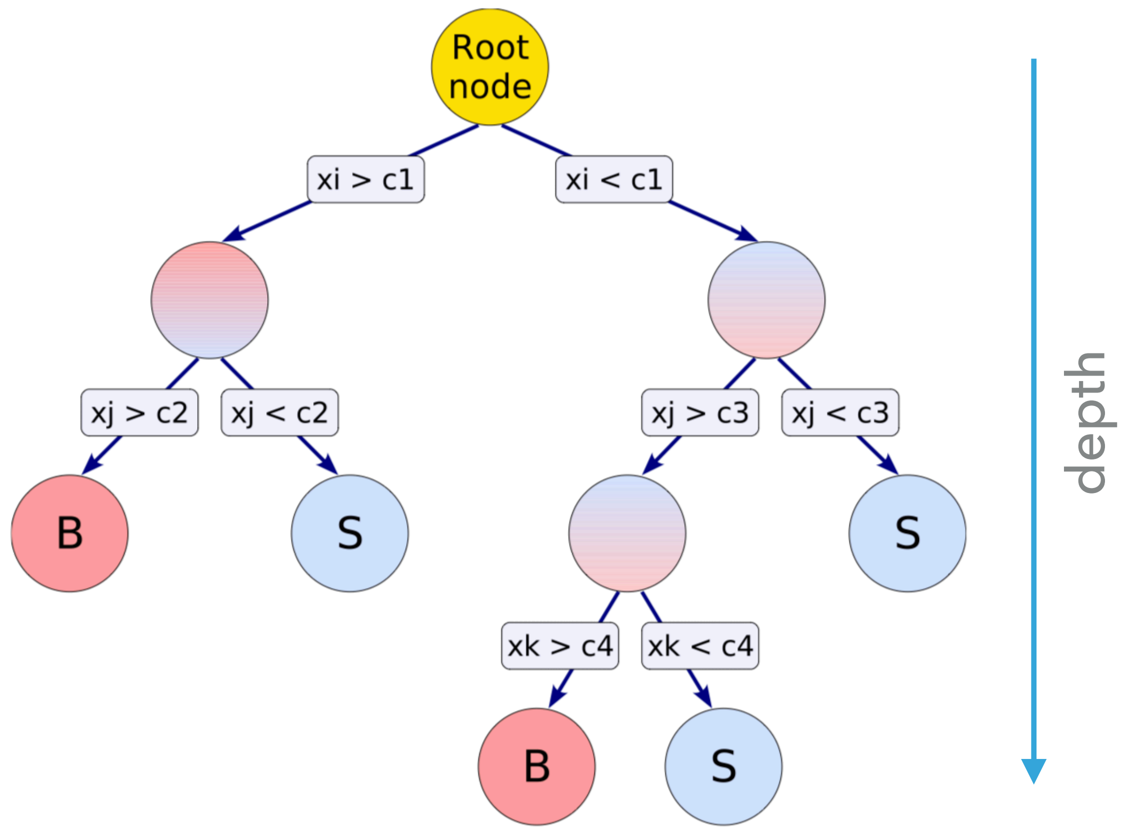




The feature space gets further sub-divided (again).

DECISION TREES

- The set of rectangular cuts applied to the data allow us to build a tree from the root note.
- We can impose limits on:
  - Tree depth (how many divisions are performed).
  - Node size (how many examples per partition).
- Trees can be extended to more than 2 categories.
- They lend themselves to classifying examples or adapted to make a quantitative prediction (regression)



The decision tree output for a classification problem is

$$G(x) = +1 \text{ or } -1$$

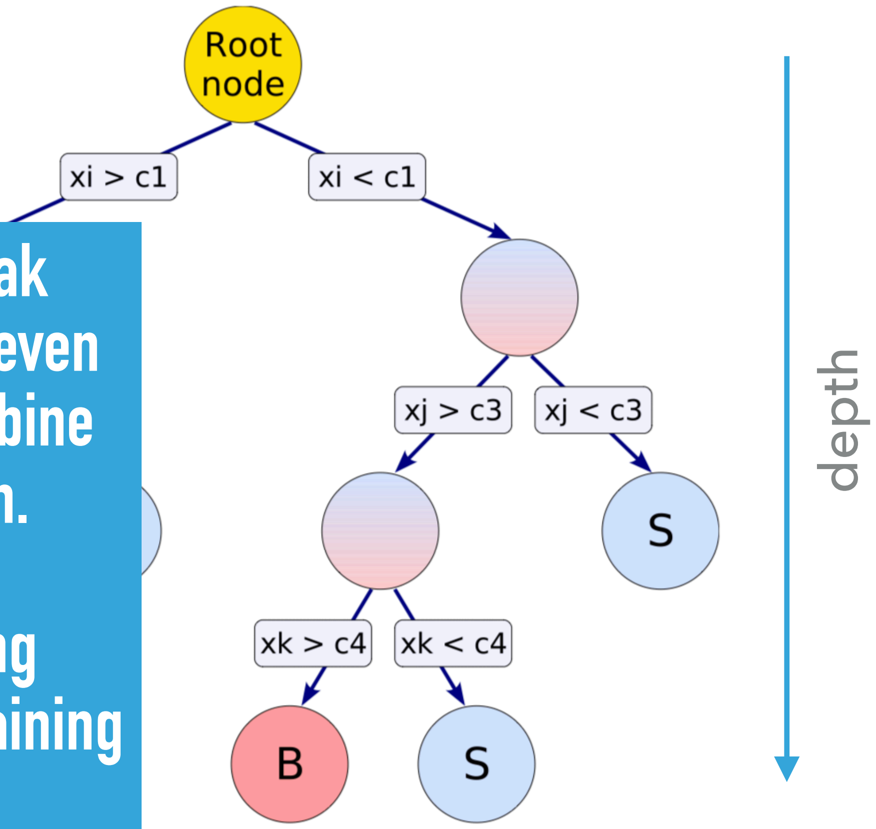
DECISION TREES

- The set of rectangular cuts applied to the data allow us to build a tree from the root note.
- We can impose limit
  - Tree depth (how r performed).
  - Node size (how m partition).
- Trees can be extende categories.
- They lend themselves to classifying examples or adapted to make a quantitative prediction (regression)

A Decision tree is what is known as a weak learner. It can take the input features in X, even when they are weakly separating, and combine those features to increase the separation.

A single tree is susceptible to overtraining (learning the statistical fluctuations in the training data)

As we shall see weak learners can be combined



The decision tree output for a classification problem is

$$G(x) = +1 \text{ or } -1$$

BOOSTING

- If a training example has been mis-classified in a training epoch, then the weight of that event can be increased for the next training epoch; so that the cost of mis-classification increases.
- The underlying aim is to take a weak learner and try and boost this into becoming a strong learner.
  - This example re-weighting technique is called boosting.
  - There are several re-weighting methods commonly used; here we discuss:
    - AdaBoost.M1 (popular variant of the Adaptive boosting method)
- Boosted Decision Trees are known as BDTs

Freund and Schapire J. Jap. Soc. AI **14** (1999) 771-780

BOOSTING: AdaBoost.M1

- i is the ith example out of a data set with N examples.
- m is the mth training out of an ensemble of M learners to be trained.

- Step 1:
  - Assign event weights of $w_i = 1/N$ to all of the N examples.

BOOSTING: AdaBoost.M1

- i is the ith example out of a data set with N examples.
- m is the mth training out of an ensemble of M learners to be trained.

- Step 1:
  - Assign event weights of w_i = 1/N to all of the N examples.

- Step 2: for m=1 through M
  - Train the weak learner (in our case this is a BDT): $G_m(x)$.
  - Compute the error rate ε_m.
  - Calculate the boost factor $\beta_m = \dfrac{\varepsilon_m}{1 - \varepsilon_m}$.
  - Update weights for misclassified examples $w_i \mapsto w_i e^{\ln(1/\beta_m)}$.

BOOSTING: AdaBoost.M1

- i is the ith example out of a data set with N examples.
- m is the mth training out of an ensemble of M learners to be trained.

- Step 1:
  - Assign event weights of w_i = 1/N to all of the N examples.

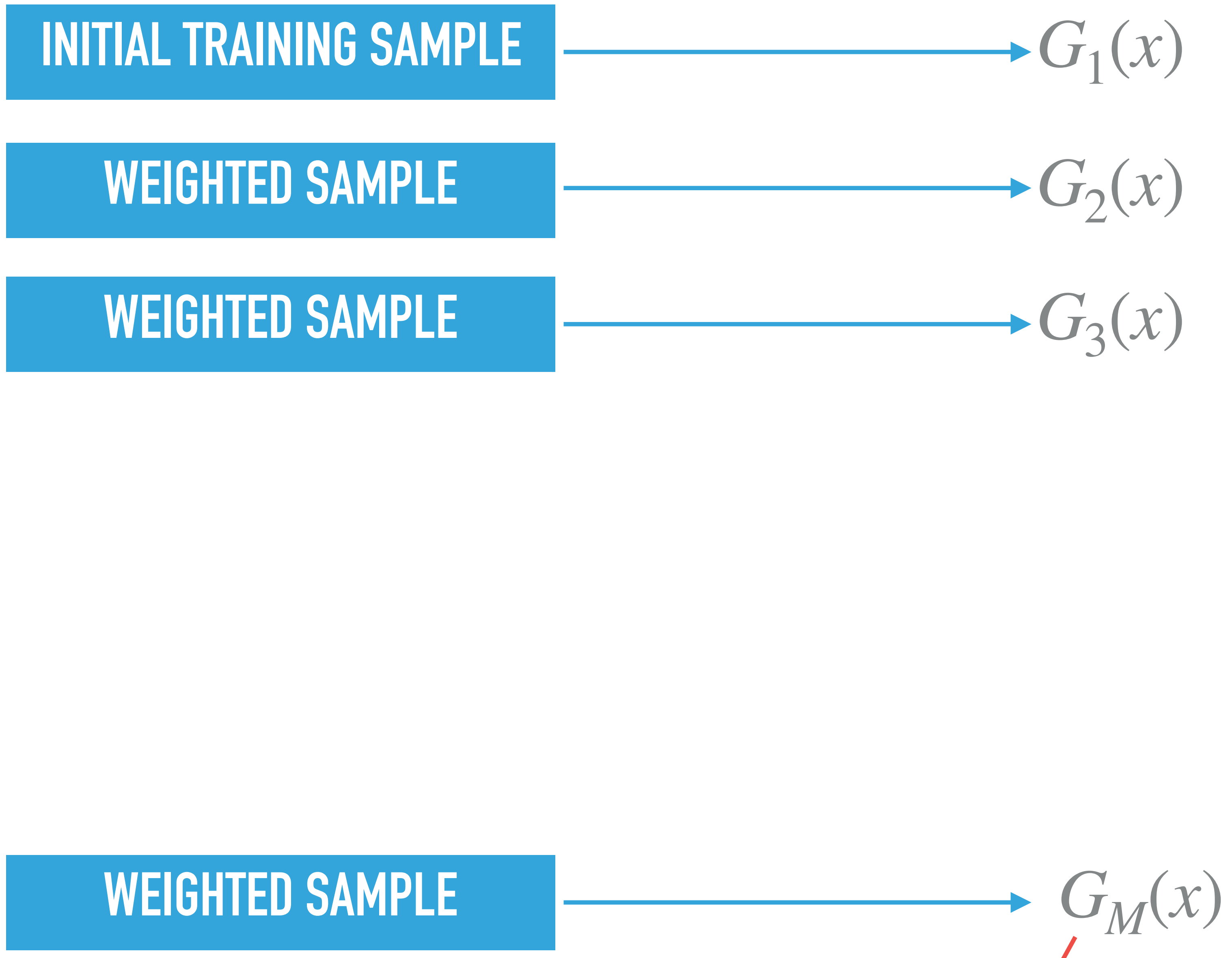
- Step 2: for m=1 through M
  - Train the weak learner (in our case this is a BDT): $G_m(x)$.
  - Compute the error rate ε_m.
  - Calculate the boost factor $\beta_m = \dfrac{\varepsilon_m}{1-\varepsilon_m}$.
  - Update weights for misclassified examples $w_i \mapsto w_i e^{\ln(1/\beta_m)}$.

- Step 3:
  - Return the weighted committee: a combination of the M trees that have been learned from the data:

$$G(x) = sign\left(\sum_{m=1}^{M} ln\left[\frac{1-\varepsilon_m}{\varepsilon_m}\right] G_m(x) \right)$$

BOOSTING: AdaBoost.M1

- m=1
- m=2
- m=3
- m=M



The $G_m(x)$ are individual weak learners; each is derived from a training using the data.

The m=1 training uses the original data; all subsequent trainings use reweighted data.

The final classifier output is formed from a committee that is a weighted majority vote algorithm.

$$G(x) = sign\left(\sum_{m=1}^{M} ln\left[\frac{1-\varepsilon_m}{\varepsilon_m}\right] G_m(x)\right)$$

BOOSTING

- Another popular boosting method is GradBoost.
- This uses some loss function (we will discuss these later when talking about optimisation) and determines the gradient with respect to the model prediction.
- See for example Ch. 10 of Hastie for more details.

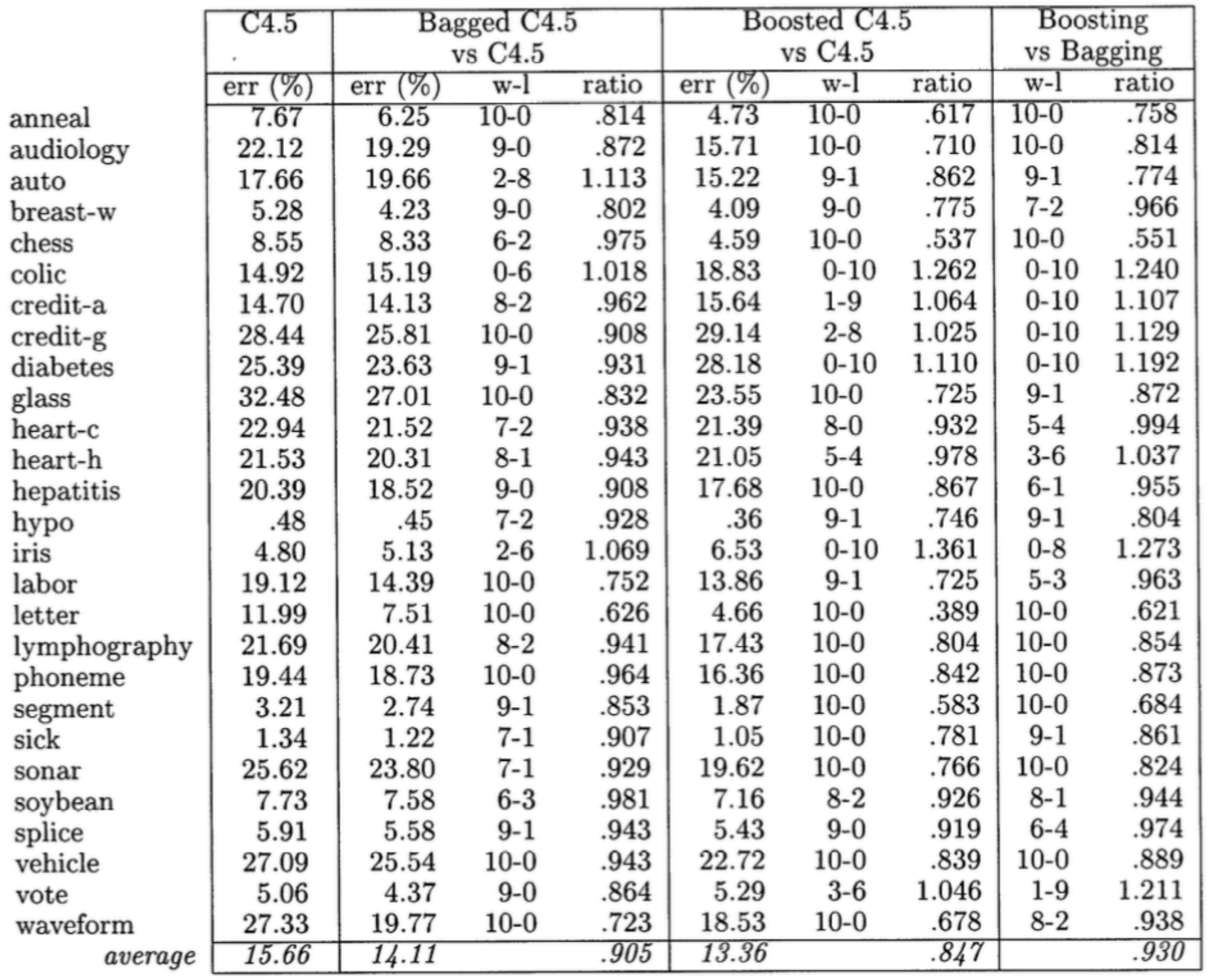
BAGGING: BOOTSTRAP AGGREGATING

- To overcome the issue of learning the statistical fluctuations of the training set, we can prepare bootstrap samples of data.
  - The sample of training data are resampled, and for each of the resampled sets a decision tree is trained.
  - The decisions of all of the models learned are combined by majority voting.
- i.e. if an example is found to be signal more times than background, then it is classified as signal.
- We can also ascribe a number between 0 and 1 for an example; this would just be the fraction of times that the resampled trees assigned the label "signal" to an example.

B. Efron, 1979, Ann. Stat **7** (1) 1-28
L. Breiman, Machine Learning **24** (1996) 123-140; J. R. Quinlan, Proc. AAAI 1996 725-730

BAGGING: BOOTSTRAP AGGREGATING

- The purpose of this method is to reduce the error on the prediction of the decision tree algorithm.



	C4.5	Bagged C4.5 vs C4.5			Boosted C4.5 vs C4.5			Boosting vs Bagging	
	err (%)	err (%)	w-l	ratio	err (%)	w-l	ratio	w-l	ratio
anneal	7.67	6.25	10-0	.814	4.73	10-0	.617	10-0	.758
audiology	22.12	19.29	9-0	.872	15.71	10-0	.710	10-0	.814
auto	17.66	19.66	2-8	1.113	15.22	9-1	.862	9-1	.774
breast-w	5.28	4.23	9-0	.802	4.09	9-0	.775	7-2	.966
chess	8.55	8.33	6-2	.975	4.59	10-0	.537	10-0	.551
colic	14.92	15.19	0-6	1.018	18.83	0-10	1.262	0-10	1.240
credit-a	14.70	14.13	8-2	.962	15.64	1-9	1.064	0-10	1.107
credit-g	28.44	25.81	10-0	.908	29.14	2-8	1.025	0-10	1.129
diabetes	25.39	23.63	9-1	.931	28.18	0-10	1.110	0-10	1.192
glass	32.48	27.01	10-0	.832	23.55	10-0	.725	9-1	.872
heart-c	22.94	21.52	7-2	.938	21.39	8-0	.932	5-4	.994
heart-h	21.53	20.31	8-1	.943	21.05	5-4	.978	3-6	1.037
hepatitis	20.39	18.52	9-0	.908	17.68	10-0	.867	6-1	.955
hypo	.48	.45	7-2	.928	.36	9-1	.746	9-1	.804
iris	4.80	5.13	2-6	1.069	6.53	0-10	1.361	0-8	1.273
labor	19.12	14.39	10-0	.752	13.86	9-1	.725	5-3	.963
letter	11.99	7.51	10-0	.626	4.66	10-0	.389	10-0	.621
lymphography	21.69	20.41	8-2	.941	17.43	10-0	.804	10-0	.854
phoneme	19.44	18.73	10-0	.964	16.36	10-0	.842	10-0	.873
segment	3.21	2.74	9-1	.853	1.87	10-0	.583	10-0	.684
sick	1.34	1.22	7-1	.907	1.05	10-0	.781	9-1	.861
sonar	25.62	23.80	7-1	.929	19.62	10-0	.766	10-0	.824
soybean	7.73	7.58	6-3	.981	7.16	8-2	.926	8-1	.944
splice	5.91	5.58	9-1	.943	5.43	9-0	.919	6-4	.974
vehicle	27.09	25.54	10-0	.943	22.72	10-0	.839	10-0	.889
vote	5.06	4.37	9-0	.864	5.29	3-6	1.046	1-9	1.211
waveform	27.33	19.77	10-0	.723	18.53	10-0	.678	8-2	.938
average	*15.66*	*14.11*		*.905*	*13.36*		*.847*		*.930*

Table 1: Comparison of C4.5 and its bagged and boosted versions.

From: J. R. Quinlan, Proc. AAAI 1996 725-730

BAGGING: BOOTSTRAP AGGREGATING

- The purpose of this method is to reduce the error on the prediction of the decision tree algorithm.

From: J. R. Quinlan, Proc. AAAI 1996 725-730

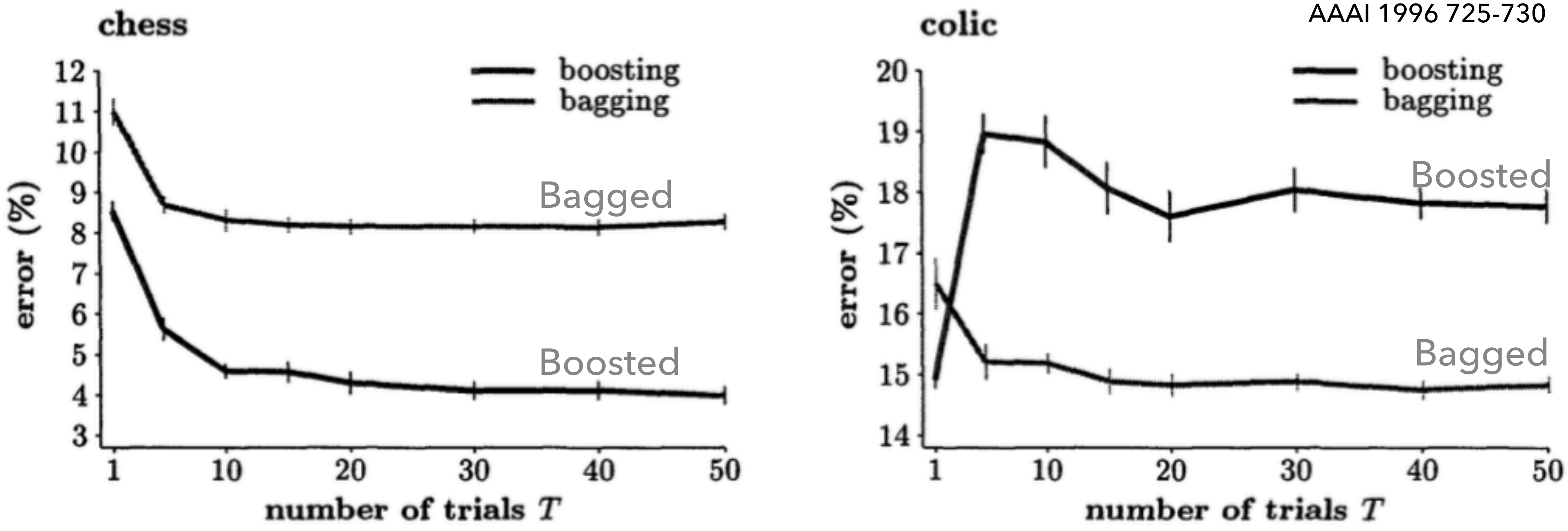


Figure 1: Comparison of bagging and boosting on two datasets

Bagging and boosting lead to better classifiers (smaller errors) for al 27 data sets studied by Quinlan.

Boosting is superior to Bagging on 20 of the 27 datasets.

RANDOM FORESTS

- Bagging involves oversampling the data for a given tree configuration in order to make a majority vote decision on the classification of an example, or to compute a regression output.
- Random Forests (RFs) are an extension of bagged decision trees that use bootstrap samples of data, and grow an ensemble of randomly different trees T_b.
- The output of the ensemble can be used to perform regression and classification tasks.

RANDOM FORESTS

- Step 1: b = 1 to B bootstrap samples
  - Create a bootstrap sample of size N from the training data.
  - Grow a random forest of trees, T_b, by repeatedly following the steps below for each terminal node, until some minimum node size is reached
    - Randomly selecting m variables to create the tree
    - Pick the best split point among the m
    - Split the node into two daughter nodes

RANDOM FORESTS

- Step 1: b = 1 to B bootstrap samples
  - Create a bootstrap sample of size N from the training data.
  - Grow a random forest of trees, T_b, by repeatedly following the steps below for each terminal node, until some minimum node size is reached
    - Randomly selecting m variables to create the tree
    - Pick the best split point among the m
    - Split the node into two daughter nodes

- Step 2:
  - Output the ensemble of trees given by the set $\{T_b\}_1^B$.

RANDOM FORESTS

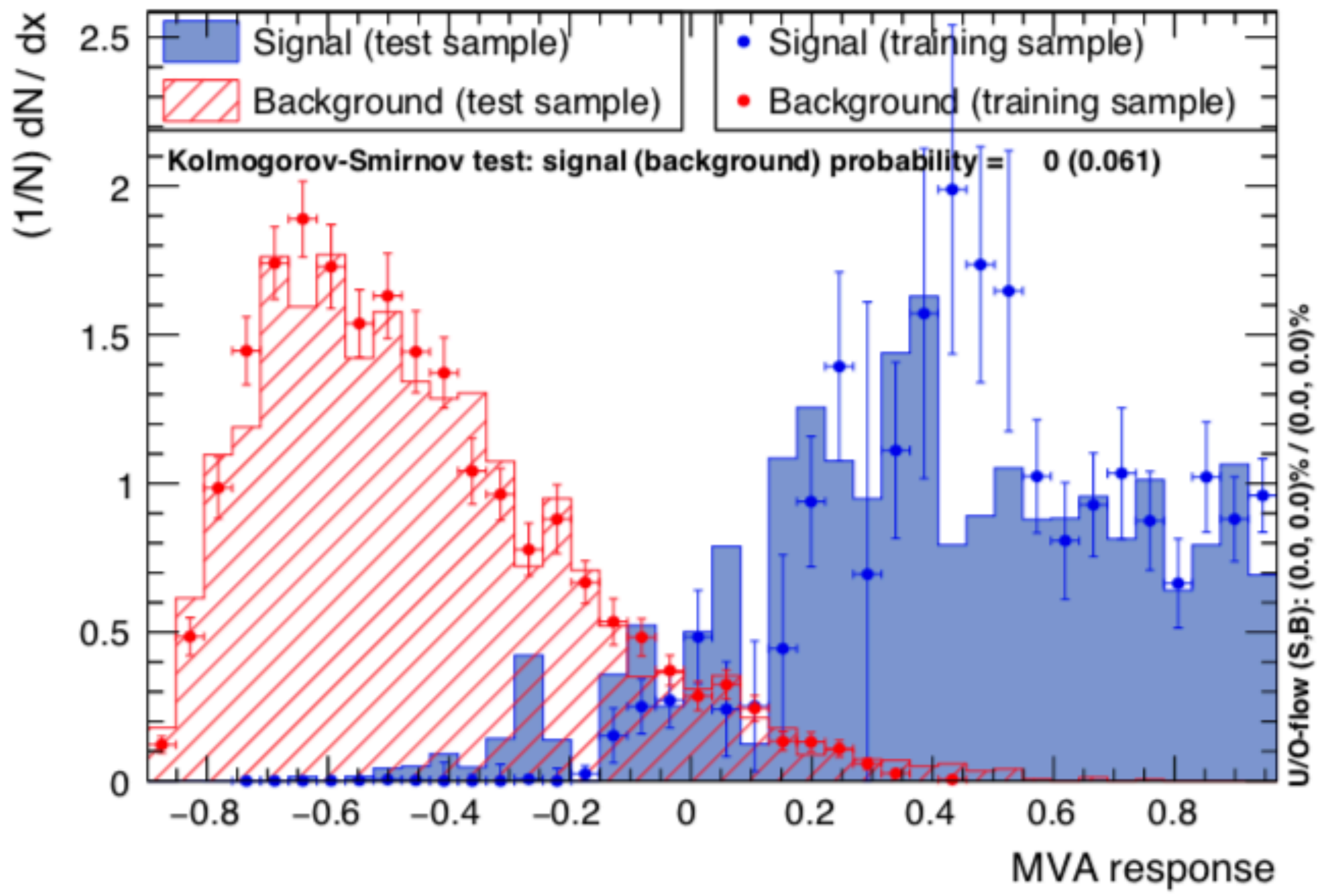
- Step 1: b = 1 to B bootstrap samples
  - Create a bootstrap sample of size N from the training data.
  - Grow a random forest of trees, T_b, by repeatedly following the steps below for each terminal node, until some minimum node size is reached
    - Randomly selecting m variables to create the tree
    - Pick the best split point among the m
    - Split the node into two daughter nodes

- Step 2:
  - Output the ensemble of trees given by the set $\{T_b\}_1^B$.

- Step 3:
  - For some new example x, make a prediction:
    - Regression: $$\hat{f}(x) = \frac{1}{B}\sum_1^B T_b(x)$$
    - Classification: majority vote of the $\{T_b\}_1^B$ predictions for the example x.

OVERTRAINING CHECK IN TMVA

- How do you tell if an MVA is overtrained?
  - It turns out there is no general agreement on how to do this. Many similarity metrics that can be found in the literature.
- TMVA has an "overtraining check" implemented: a binned KS test (known to be biased).



If a BDT is overtrained then the output distribution will be different for test and training samples.

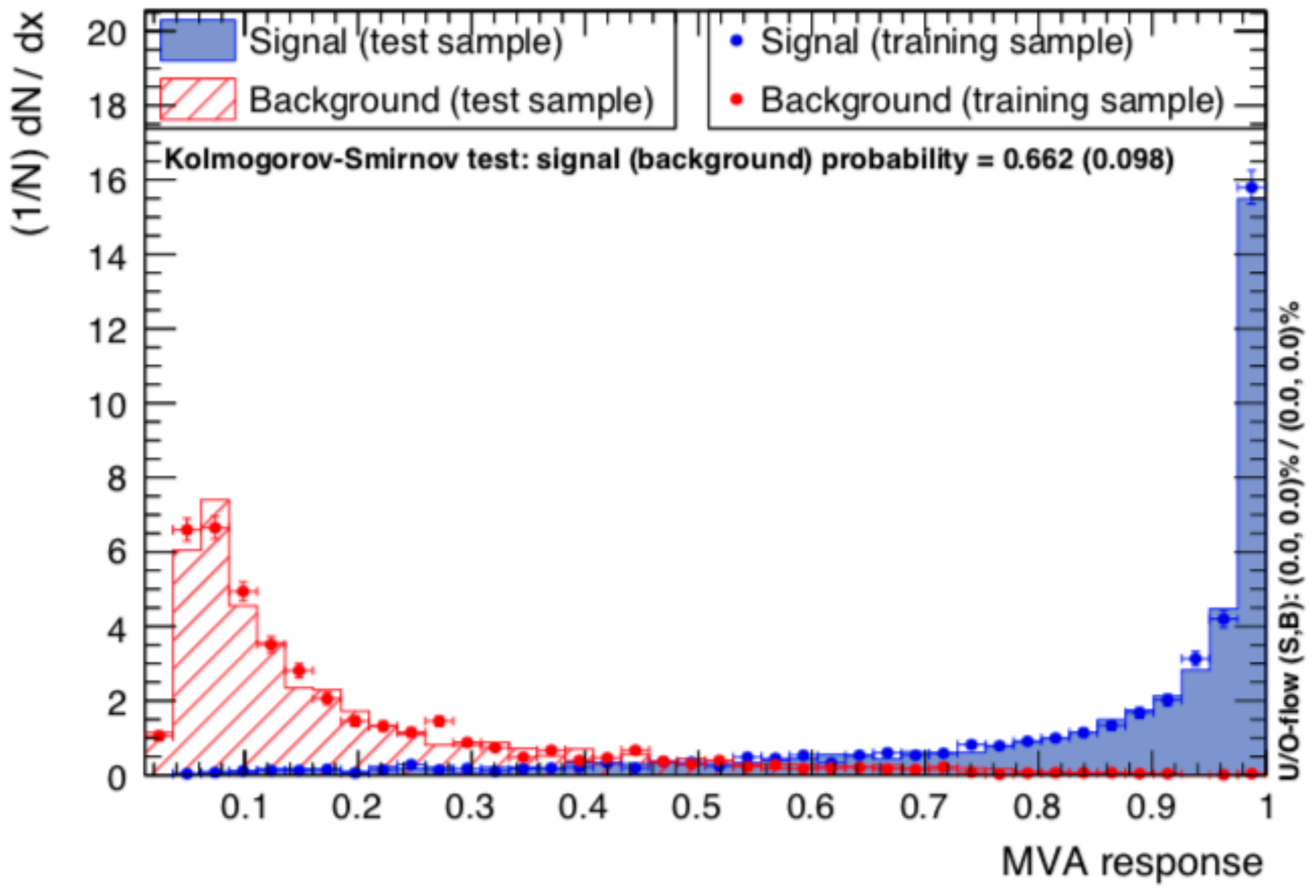
The KS probability recorded in this plot is not correct. It is a binned, rather than un-binned computation that is biased toward small probabilities.

A large probability means the test and training data agree, a small probability don't necessarily mean they disagree. However in this case the signal is clearly overtrained.

Example overtrained BDT from Tom Stevenson; PhD thesis CERN-THESIS-2018-119.

OVERTRAINING CHECK IN TMVA

- How do you tell if an MVA is overtrained?
  - It turns out there is no general agreement on how to do this. Many similarity metrics that can be found in the literature.
- TMVA has an "overtraining check" implemented: a binned KS test (known to be biased).



If an MVA (this is an SVM) is not overtrained then the output distribution will be similar for test and training samples.

In this case both signal and background have large KS probabilities, so even though the binned KS is biased, we conclude that this model is not overtrained (in contrast to the BDT shown on the previous page that was trained for the same problem with the same data).

Example SVM (not overtrained) from Tom Stevenson; PhD thesis CERN-THESIS-2018-119.

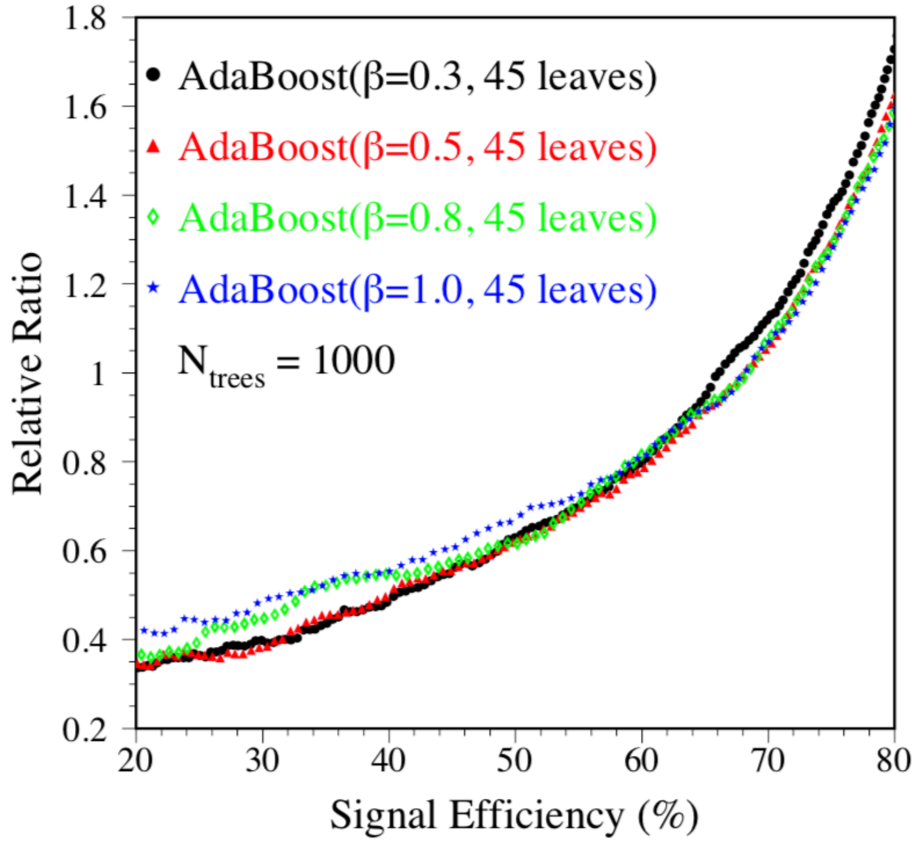
EXAMPLES: MINIBOONE

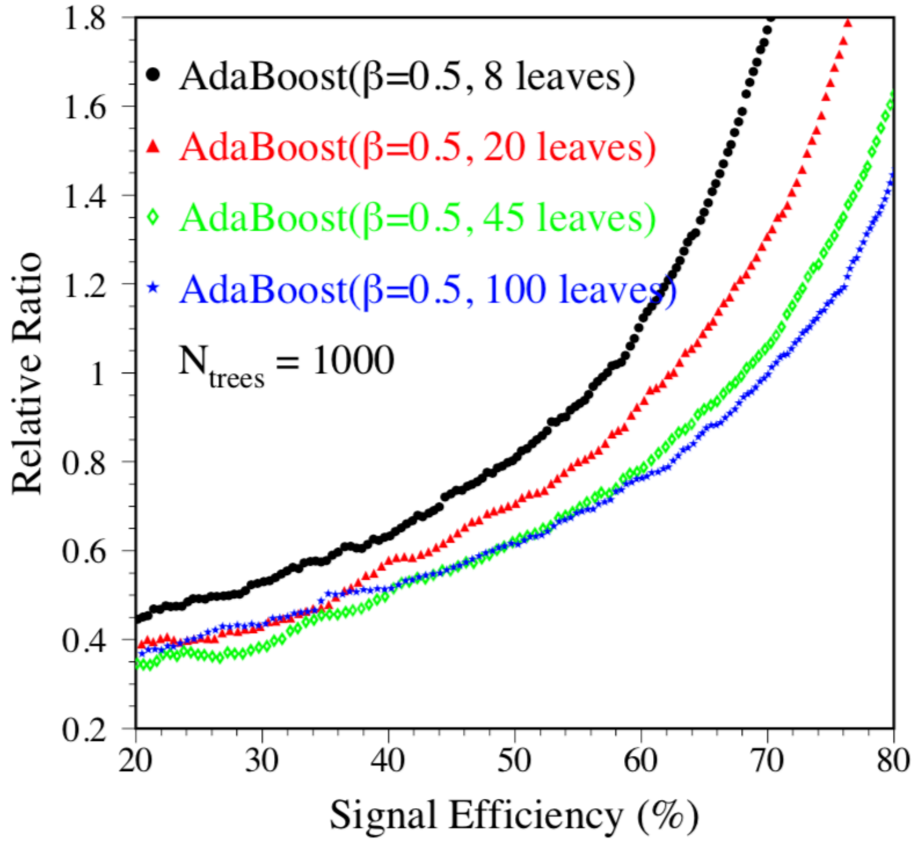
- The first experiment to start using BDTs was MiniBooNE.
- Following on from the introduction of these methods into particle physics the techniques became popular.
  - In part this will have been because of the comparative ease of understanding.
  - The availability of algorithms to apply to data will have facilitated the use of BDTs in HEP.
  - Nowadays BDTs are more commonly used than the previously well established approaches (e.g. neural networks).

H.-J. Yang et al., Nucl.Instrum.Meth. A**555** (2005) 370-385

EXAMPLES: MINIBOONE

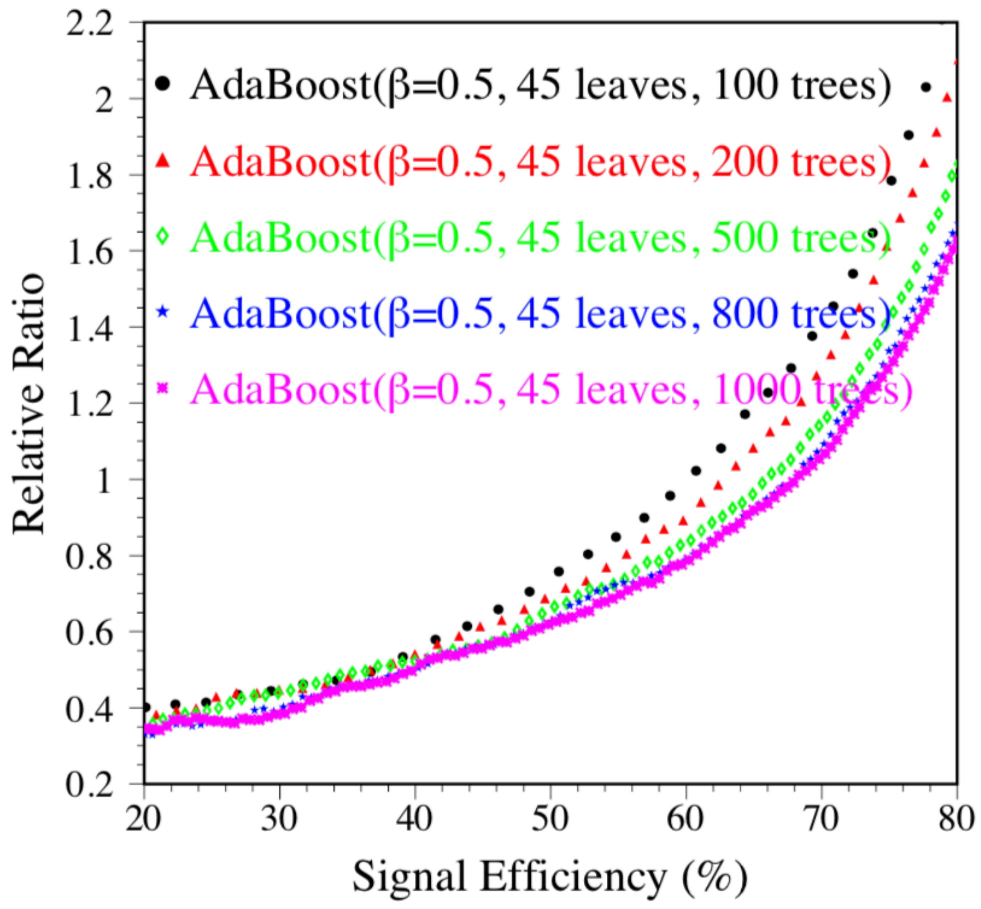
- The MiniBooNE physics programme was to confirm or refute the LSND anomaly of $\nu_\mu \rightarrow \nu_e$ with $\Delta m^2 \sim 1\ (\mathrm{eV}/c^2)^2$
- BDTs were used for background suppression.

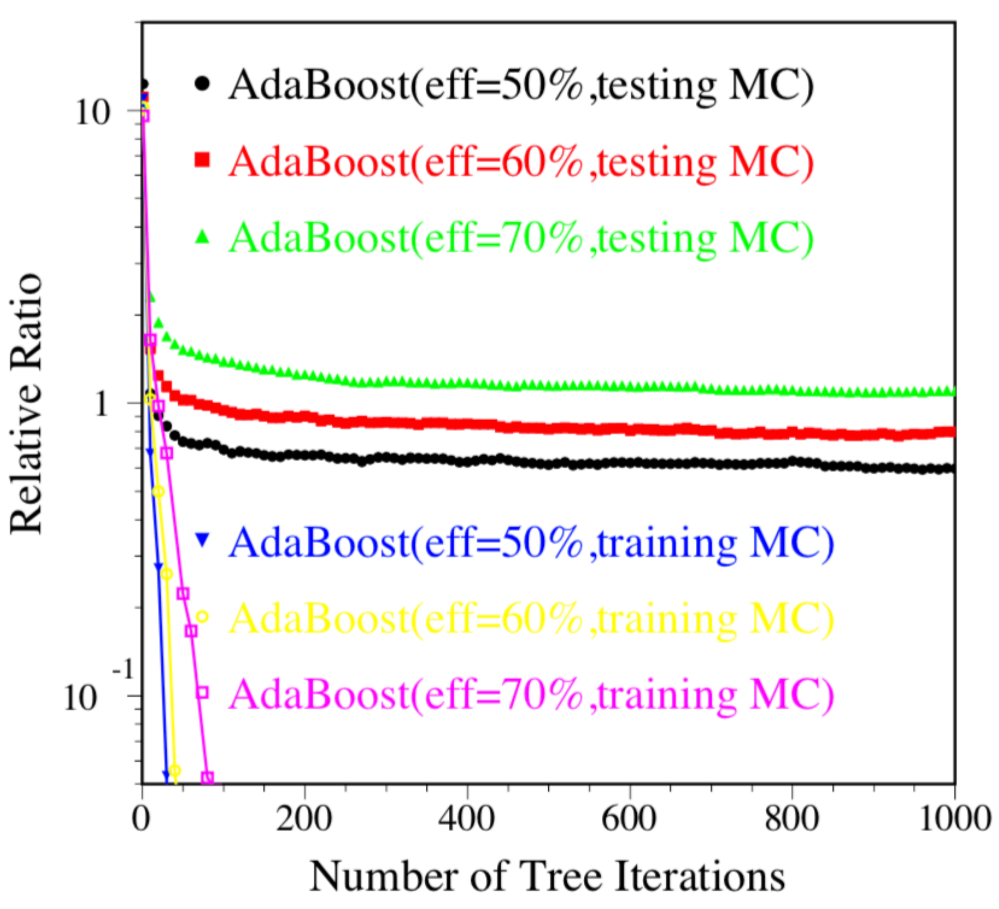




EXAMPLES: MINIBOONE

- The MiniBooNE physics programme was to confirm or refute the LSND anomaly of $\nu_\mu \rightarrow \nu_e$ with $\Delta m^2 \sim 1\ (\mathrm{eV}/c^2)^2$
- BDTs were used for background suppression.





EXAMPLES: BABAR

- The SLAC-based particle physics experiment was built to discover CP violation in the B meson system.
- Charged particle identification was a key requirement of the programme.
- BDTs were one MVA tool used to perform particle identification.

Several sub-systems provided measurement information that was fed into different MVA methods to identify charged particles, and to quantify mis-id rates.

This table shows typical performance achieved for μ and K identification.

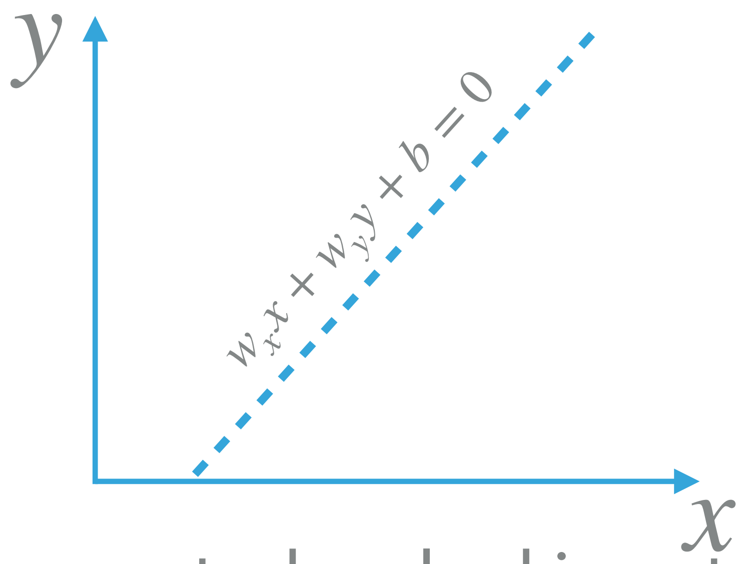
Muon selector	efficiency (%)	π mis-id rate (%)
Cut based	65.0 ± 0.5	1.43 ± 0.05
NN	60.5 ± 0.5	0.97 ± 0.05
BDT	59.4 ± 0.5	0.76 ± 0.05
Kaon selector	**efficiency (%)**	**π mis-id rate (%)**
Cut based	80.2 ± 0.2	1.39 ± 0.07
Likelihood based	83.0 ± 0.2	1.47 ± 0.07
ECOC	84.2 ± 0.2	1.10 ± 0.07

A. Bevan et al., Eur. Phys. J. C**74** (2014) 3026

DECISION TREES (REFLECTION)

- We can use a set of rectangular cuts - each equivalent to applying a Heavyside function as a decision boundary on the data.
  - Decision trees (including boosted, bagged, RFs) are based on this approach.

- We have not explored using a hyperplane given by $w^T \cdot x + b$ to separate signal from background.
  - Note: This hyperplane has the same functional form as the Fisher linear discriminant.
  - In 2D this becomes the constraint $w_x x + w_y y + b = 0.$



  - We will explore this aspect when looking at neural networks and support vector machines.

SUMMARY

- Decision trees are extensions of a cut based approach that allow weak learners to be trained.
- Various methods have been developed (including boosting and bagging) to promote strong learners to be developed from the underlying weak learners.
- The concepts of decision trees can be extended to random forests.
- A significant advantage of these algorithms is that they are (relatively) straightforward to understand and interpret, in comparison with some of the other methods we will discuss in the remainder of these lectures.
- BDTs generally perform well when applied to HEP problems.

SUGGESTED TOOLS

- Many decision tree tools exist that you may wish to explore. A selection of these are linked below:
  - TMVA
  - SciKitLearn
  - XGBoost
- In addition to Python and C++ based tools, you will find decision trees implemented in R, Matlab, Mathmatica etc.

SUGGESTED READING (NON-HEP)

- In addition to the references given in the slides you may be interested in:
  - Hastie, Tibshirani and Friedman, The Elements of Statistical Learning, Springer (2011).

APPENDIX

- AdaBoost (original form); this was superseded by the AdaBoost.M1 variant after a period of theoretical and empirical study of performance.
    - Normally when people talk about using an AdaBoost in HEP they are referring to the AdaBoost.M1 algorithm.

BOOSTING: ADABOOST (THE ORIGINALLY PROPOSED FORM)

- Compute the error rate ε_m for a given weak learner (e.g. the decision tree for a given training epoch) for some example x_i resulting in some distribution $D_m(x_i)$.
- Each hypothesis h_t (=0, 1) that differers from the example label y_t contributes to the error. $\varepsilon_m = \sum_{i=h_m(x_i)\neq y_m} D_m(x_i)$
- Compute an event weight based on this; given by

$$\beta_m = \frac{1}{2}\ln\left(\frac{\varepsilon_m}{1+\varepsilon_m}\right)$$

- re-weight the misclassified example such that:

$$w_i^{m+1} = w_i^m \beta_m^{(1-|h_m(x_i)-y_i|)}$$

- where $h_f(x) = \begin{cases} 1 & \text{if } \sum_{m=1}^{M} (\ln 1/\beta_m) h_m(x) \geq \frac{1}{2}\sum_{m=1}^{M}(\ln 1/\beta_m) \\ 0 & \text{otherwise} \end{cases}$

Freund and Schapire J. Jap. Soc. AI **14** (1999) 771-780; J. Comp. and Syst Sciences **55** (1997) 119-139