Decision Trees and Random Forests

Tom Rainforth
3YP Saving Oneself

23/01/17

Overview

- Motivation
- Supervised Learning
- Prediction using a Decision Tree
- Learning a Decision Tree
- Tree Ensembles and Random Forests
- Practicalities
- Extensions

Motivation

Motivation

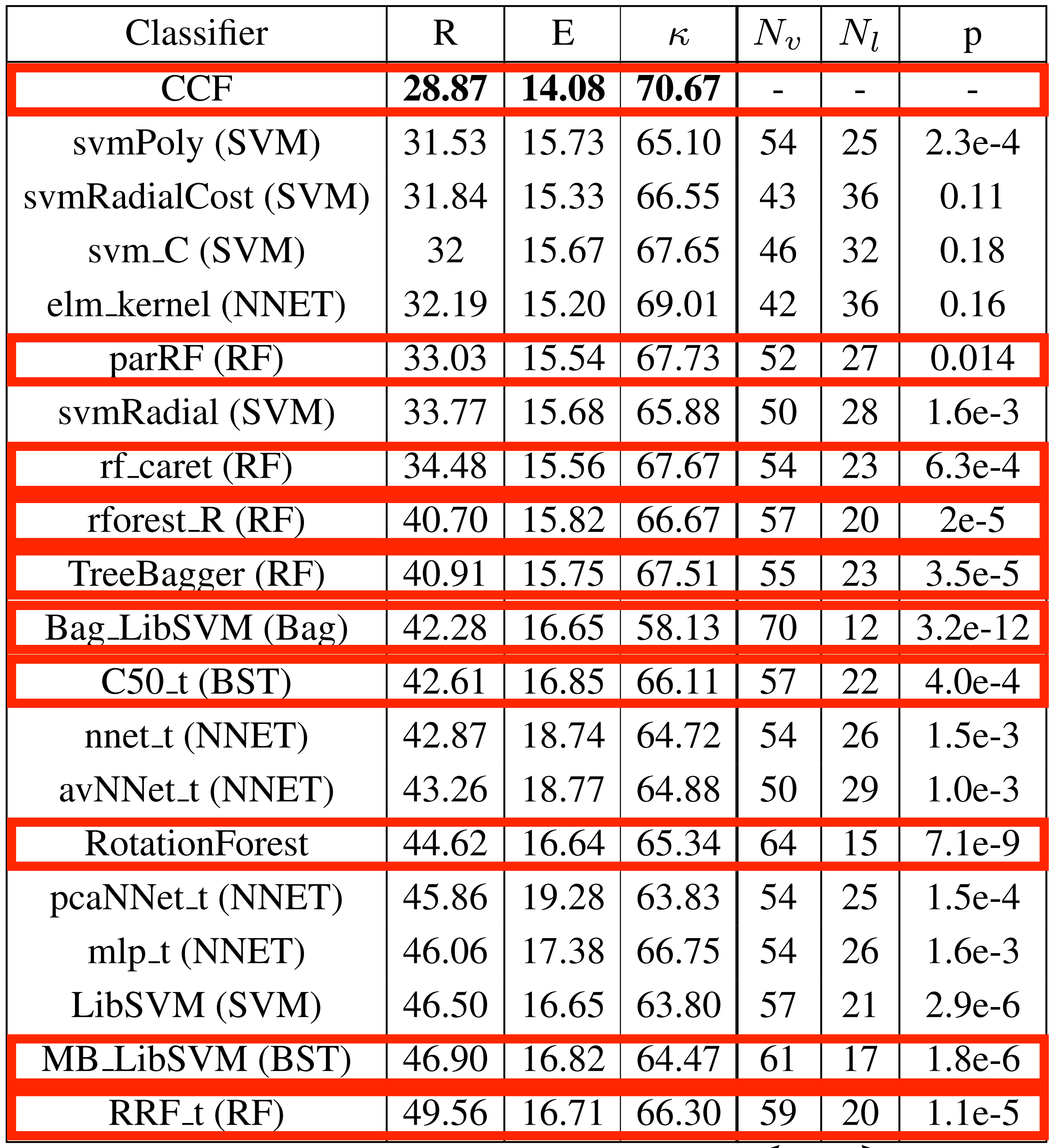
- Extremely fast
  - Training in $O(NL\lambda(\log(N))^2)$
  - Prediction in $O(L\log(N))$
- Trivial to use - most packages require only the data itself
- Ambivalent to data type - continuous, categorical etc
- Exceptional “out-of-the-box” performance

Average Rank

Average Error (%)

Cohen's κ (%)

CCF Better Significance Level



Classifier	R	E	κ	N_v	N_l	p
CCF	**28.87**	**14.08**	**70.67**	-	-	-
svmPoly (SVM)	31.53	15.73	65.10	54	25	2.3e-4
svmRadialCost (SVM)	31.84	15.33	66.55	43	36	0.11
svm_C (SVM)	32	15.67	67.65	46	32	0.18
elm_kernel (NNET)	32.19	15.20	69.01	42	36	0.16
parRF (RF)	33.03	15.54	67.73	52	27	0.014
svmRadial (SVM)	33.77	15.68	65.88	50	28	1.6e-3
rf_caret (RF)	34.48	15.56	67.67	54	23	6.3e-4
rforest_R (RF)	40.70	15.82	66.67	57	20	2e-5
TreeBagger (RF)	40.91	15.75	67.51	55	23	3.5e-5
Bag_LibSVM (Bag)	42.28	16.65	58.13	70	12	3.2e-12
C50_t (BST)	42.61	16.85	66.11	57	22	4.0e-4
nnet_t (NNET)	42.87	18.74	64.72	54	26	1.5e-3
avNNet_t (NNET)	43.26	18.77	64.88	50	29	1.0e-3
RotationForest	44.62	16.64	65.34	64	15	7.1e-9
pcaNNet_t (NNET)	45.86	19.28	63.83	54	25	1.5e-4
mlp_t (NNET)	46.06	17.38	66.75	54	26	1.6e-3
LibSVM (SVM)	46.50	16.65	63.80	57	21	2.9e-6
MB_LibSVM (BST)	46.90	16.82	64.47	61	17	1.8e-6
RRF_t (RF)	49.56	16.71	66.30	59	20	1.1e-5

N times CCF better

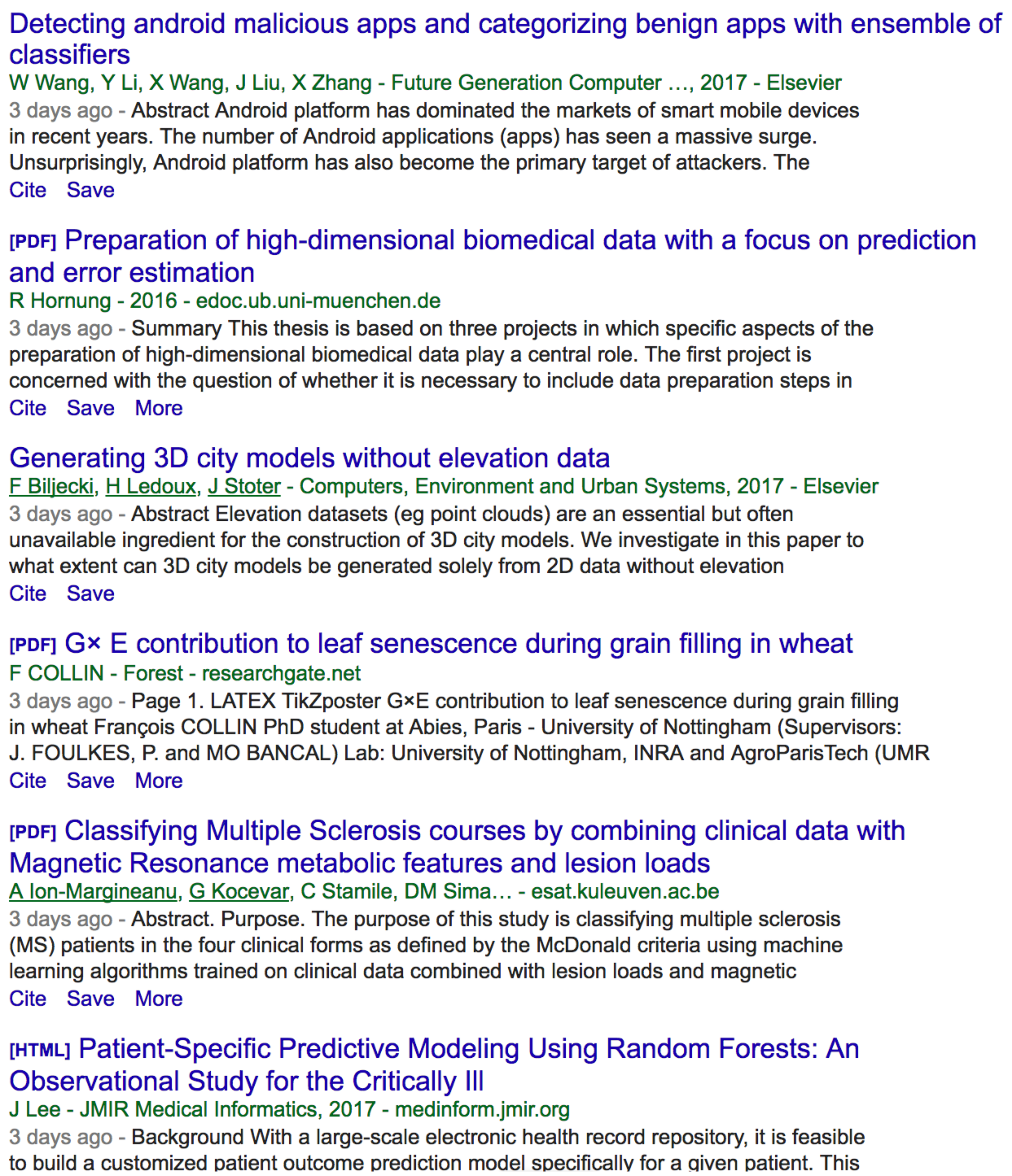
N times CCF worse

10 out of top 20 classifiers from recent survey of 180 classifiers on 82 datasets are based on ensembles of decision trees.

[Rainforth & Wood 2015]

[Fernandez-Delgado et al 2014]

Used for literally everything



Detecting android malicious apps and categorizing benign apps with ensemble of classifiers
W Wang, Y Li, X Wang, J Liu, X Zhang - Future Generation Computer …, 2017 - Elsevier
3 days ago - Abstract Android platform has dominated the markets of smart mobile devices in recent years. The number of Android applications (apps) has seen a massive surge. Unsurprisingly, Android platform has also become the primary target of attackers. The
Cite Save

[PDF] Preparation of high-dimensional biomedical data with a focus on prediction and error estimation
R Hornung - 2016 - edoc.ub.uni-muenchen.de
3 days ago - Summary This thesis is based on three projects in which specific aspects of the preparation of high-dimensional biomedical data play a central role. The first project is concerned with the question of whether it is necessary to include data preparation steps in
Cite Save More

Generating 3D city models without elevation data
F Biljecki, H Ledoux, J Stoter - Computers, Environment and Urban Systems, 2017 - Elsevier
3 days ago - Abstract Elevation datasets (eg point clouds) are an essential but often unavailable ingredient for the construction of 3D city models. We investigate in this paper to what extent can 3D city models be generated solely from 2D data without elevation
Cite Save

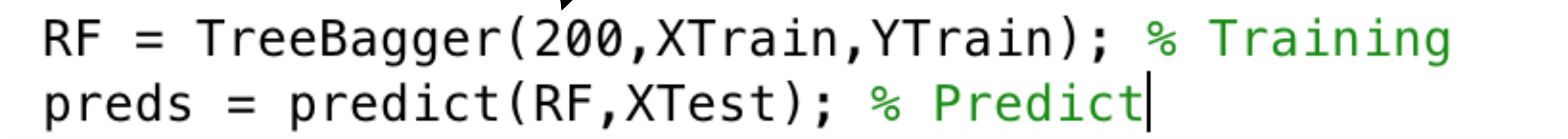
[PDF] G× E contribution to leaf senescence during grain filling in wheat
F COLLIN - Forest - researchgate.net
3 days ago - Page 1. LATEX TikZposter G×E contribution to leaf senescence during grain filling in wheat François COLLIN PhD student at Abies, Paris - University of Nottingham (Supervisors: J. FOULKES, P. and MO BANCAL) Lab: University of Nottingham, INRA and AgroParisTech (UMR
Cite Save More

[PDF] Classifying Multiple Sclerosis courses by combining clinical data with Magnetic Resonance metabolic features and lesion loads
A Ion-Margineanu, G Kocevar, C Stamile, DM Sima… - esat.kuleuven.ac.be
3 days ago - Abstract. Purpose. The purpose of this study is classifying multiple sclerosis (MS) patients in the four clinical forms as defined by the McDonald criteria using machine learning algorithms trained on clinical data combined with lesion loads and magnetic
Cite Save More

[HTML] Patient-Specific Predictive Modeling Using Random Forests: An Observational Study for the Critically Ill
J Lee - JMIR Medical Informatics, 2017 - medinform.jmir.org
3 days ago - Background With a large-scale electronic health record repository, it is feasible to build a customized patient outcome prediction model specifically for a given patient. This

Trivial to Use

Size of forest (bigger the better)



```
RF = TreeBagger(200,XTrain,YTrain); % Training
preds = predict(RF,XTest); % Predict
```

Supervised Learning

Supervised Learning

X **y**

Samples	feature 1	feature 2	...	feature *M*	Outcome
S_1	3.1	1.3		0.9	type 1
S_2	3.7	1.0		1.3	type 2
S_3	2.9	2.6		0.6	type 1
…					…
S_N	1.7	2.0		0.7	type 5

N (samples)

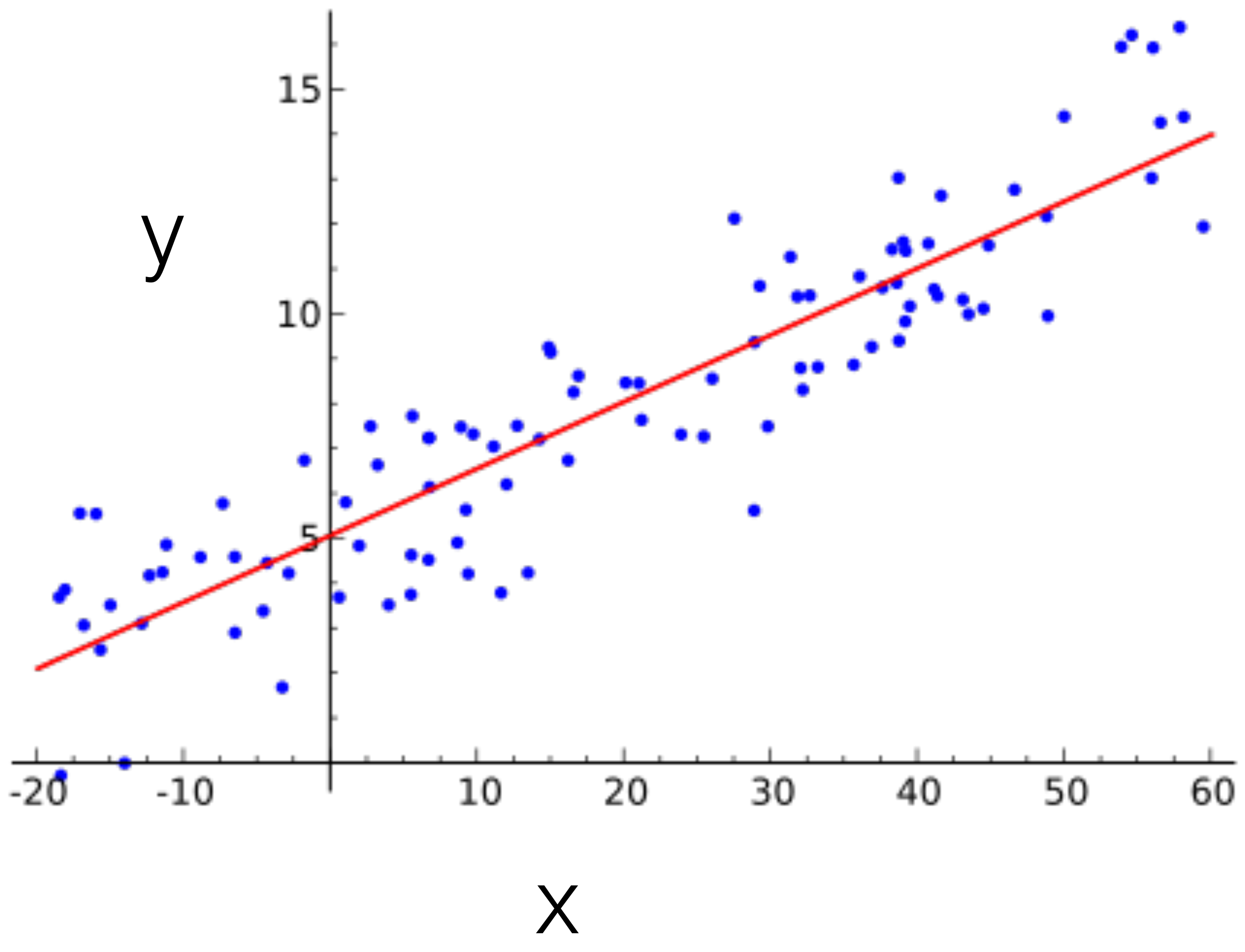
M (features or characteristics)

$$\mathbf{y} = f(\mathbf{X})$$

f **: mapping**	**X: Design matrix**	**y: outcome**

[Tsanas 2015]

Linear Regression



$f(x)=mx+c$

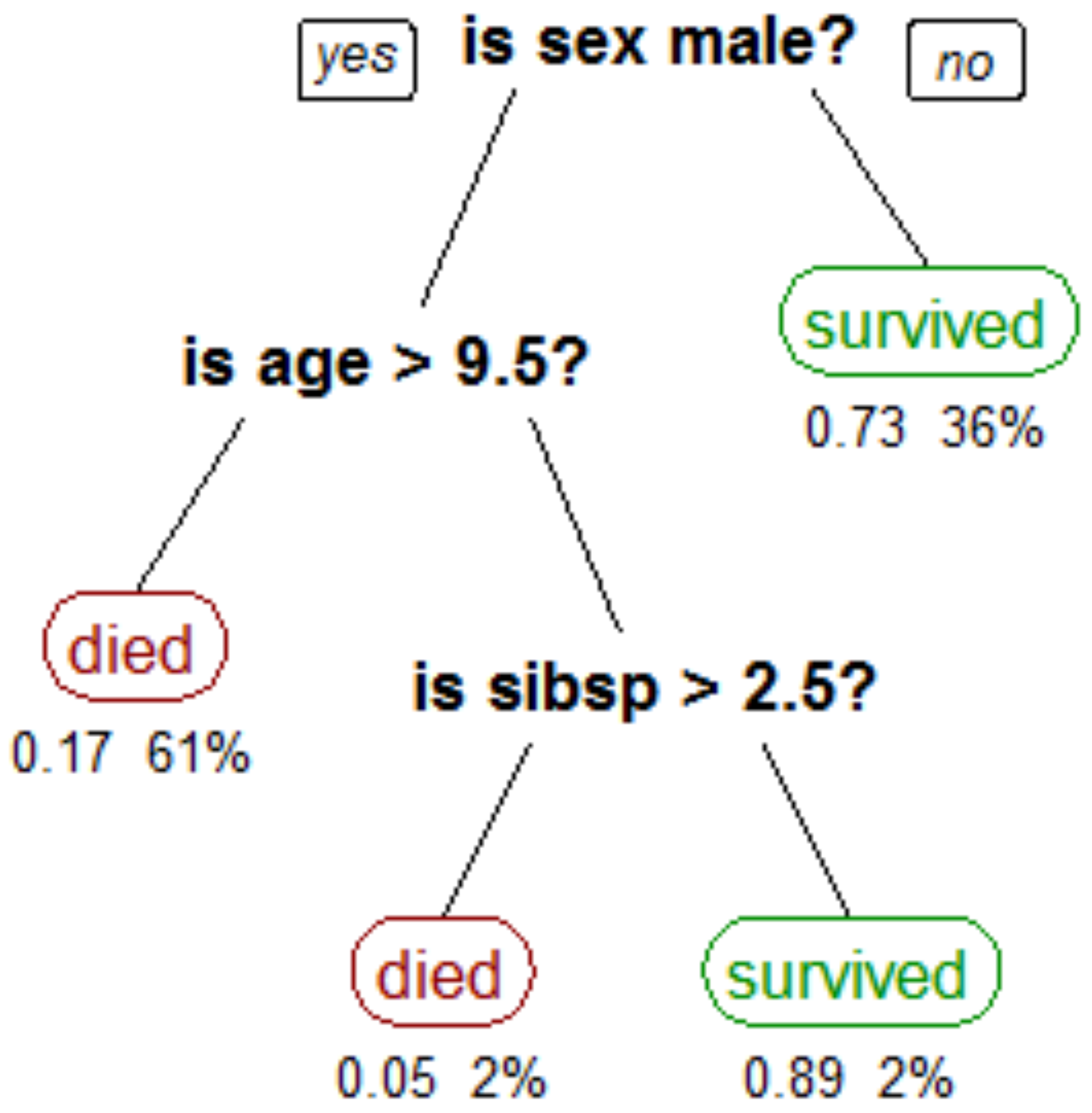
[Wikipedia]

Prediction using Decision Trees

Decision Trees

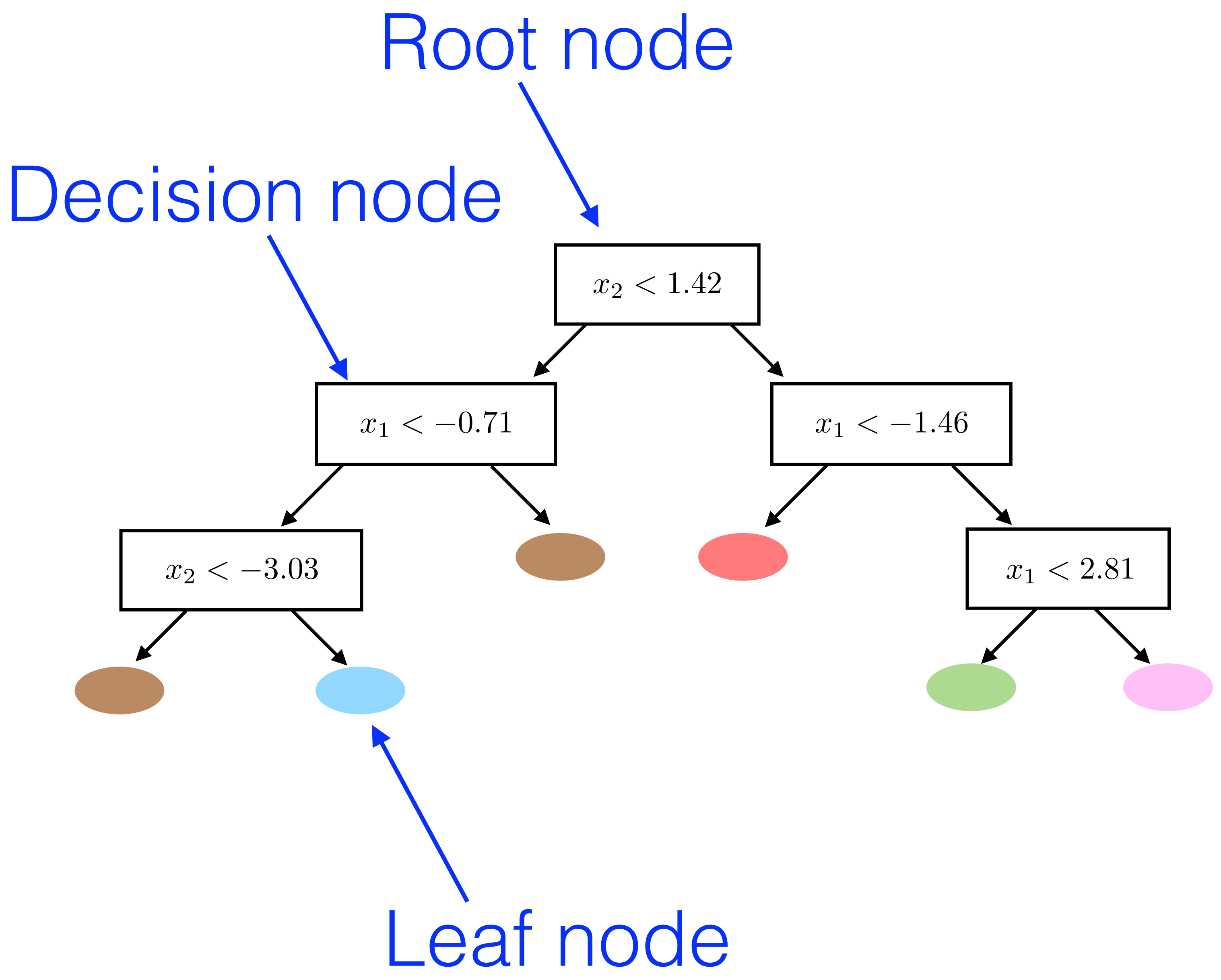
- Predictive models that impose sequential divisions of an input space
- This assigns points to “leafs”
- Prediction based on local leaf model
- Only need to consider binary trees (i.e. nodes split into two children) as can rearrange to multiple splits

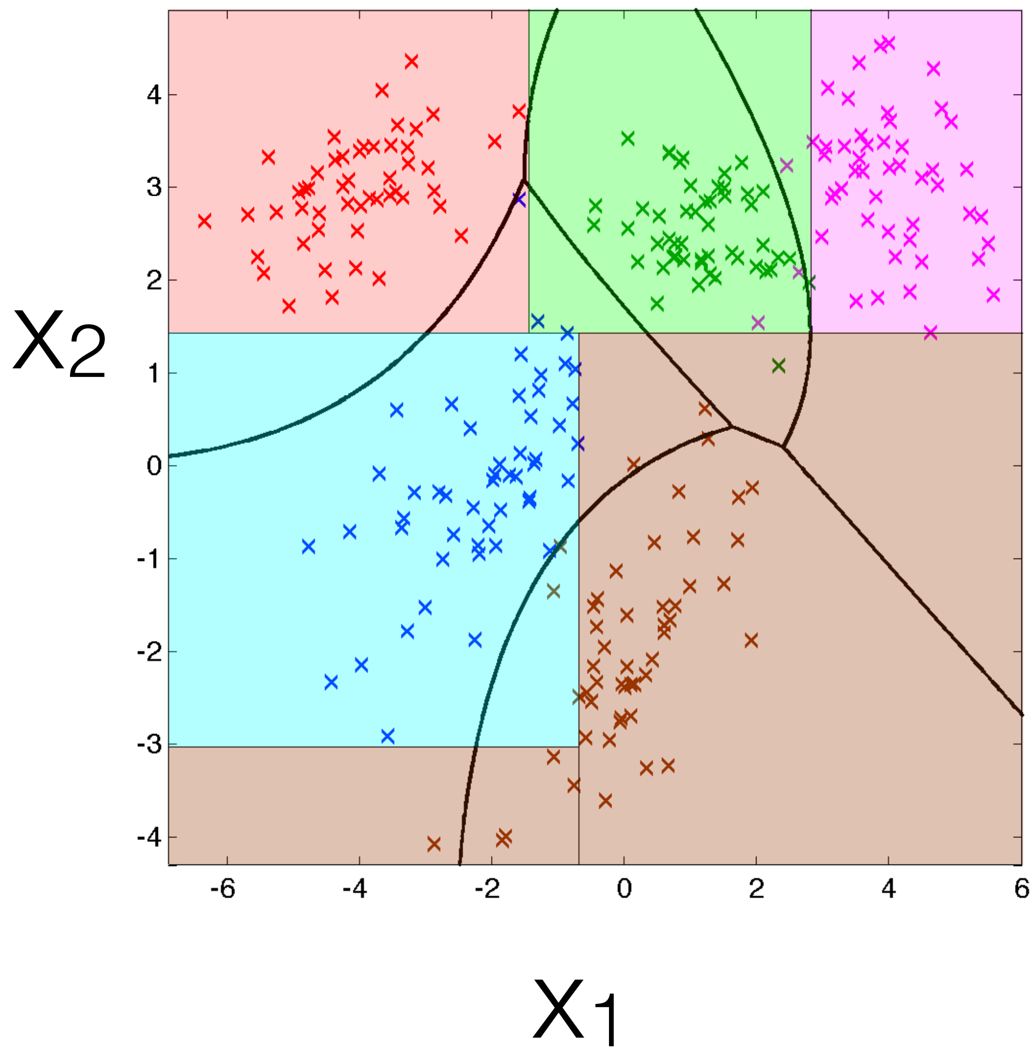
Decision Trees - Prediction



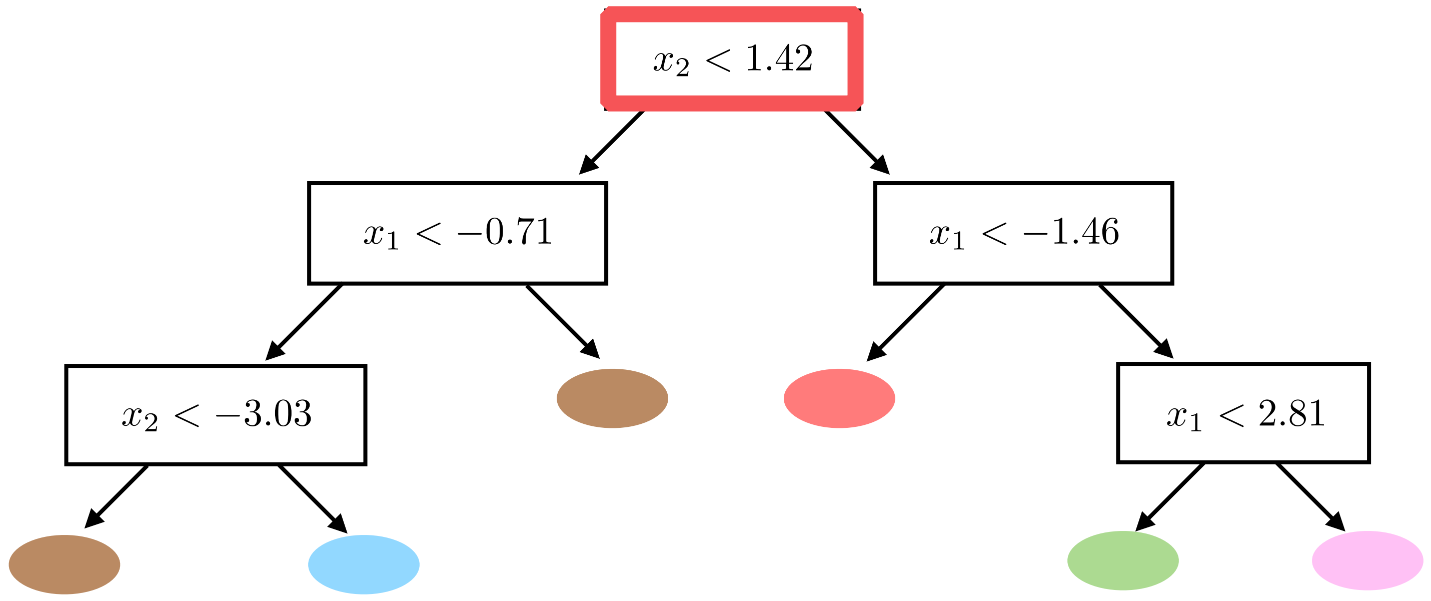
[Wikipedia]

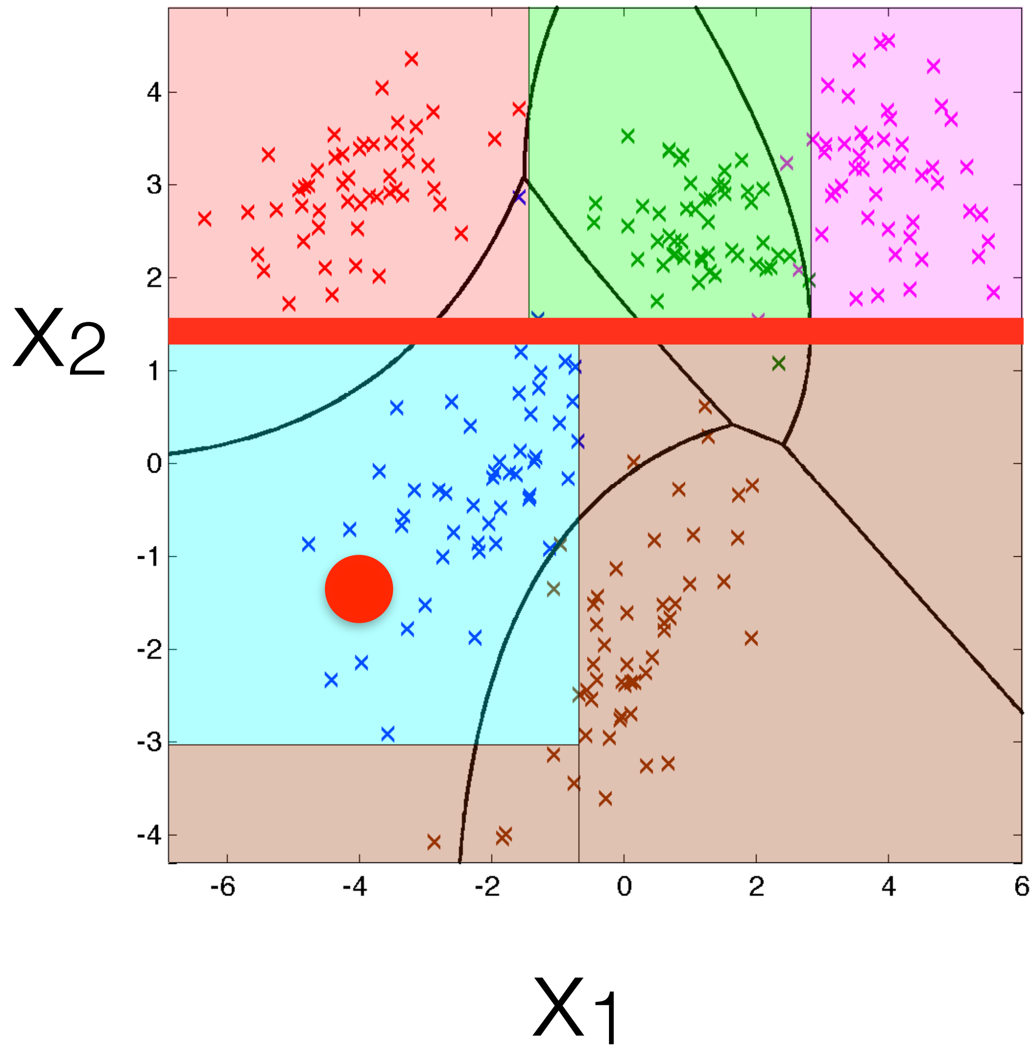
Decision Trees - Prediction



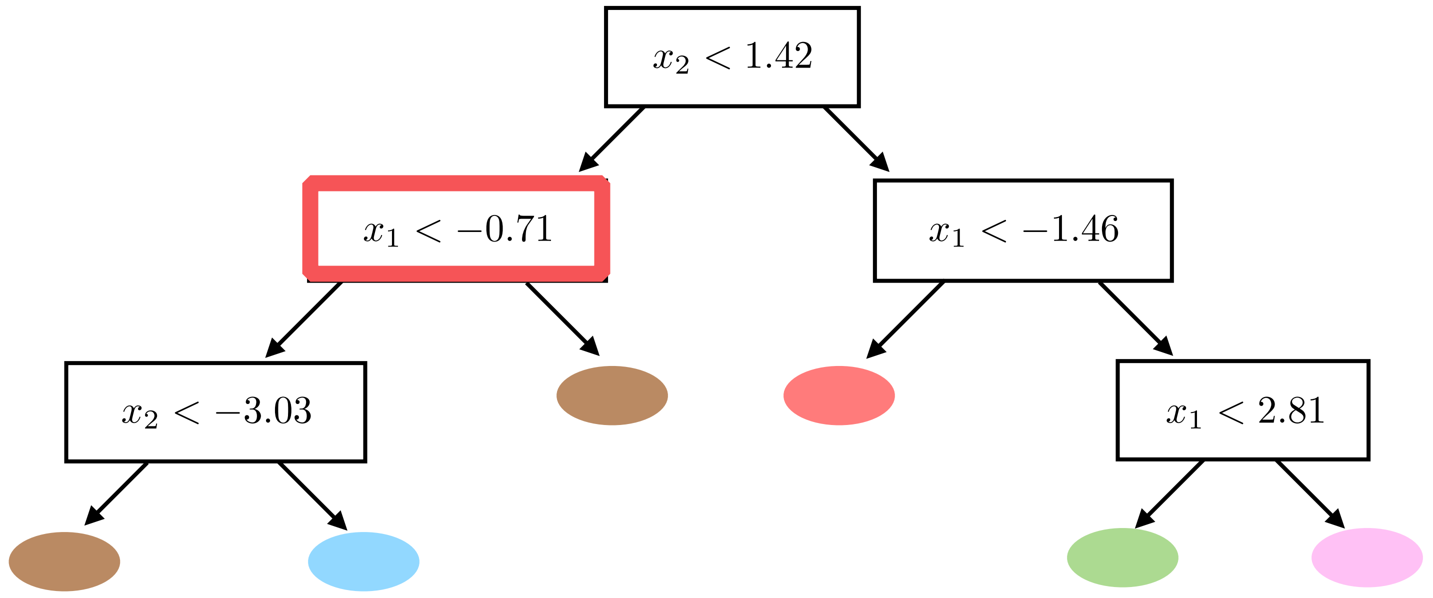


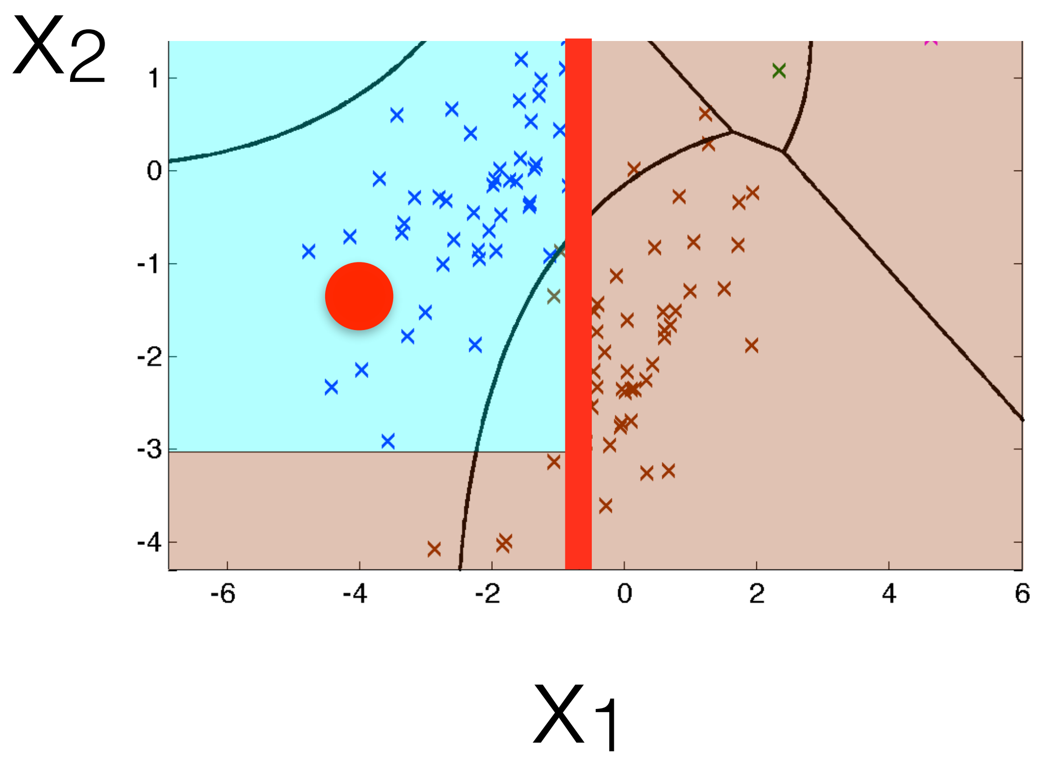
Decision Trees - Prediction



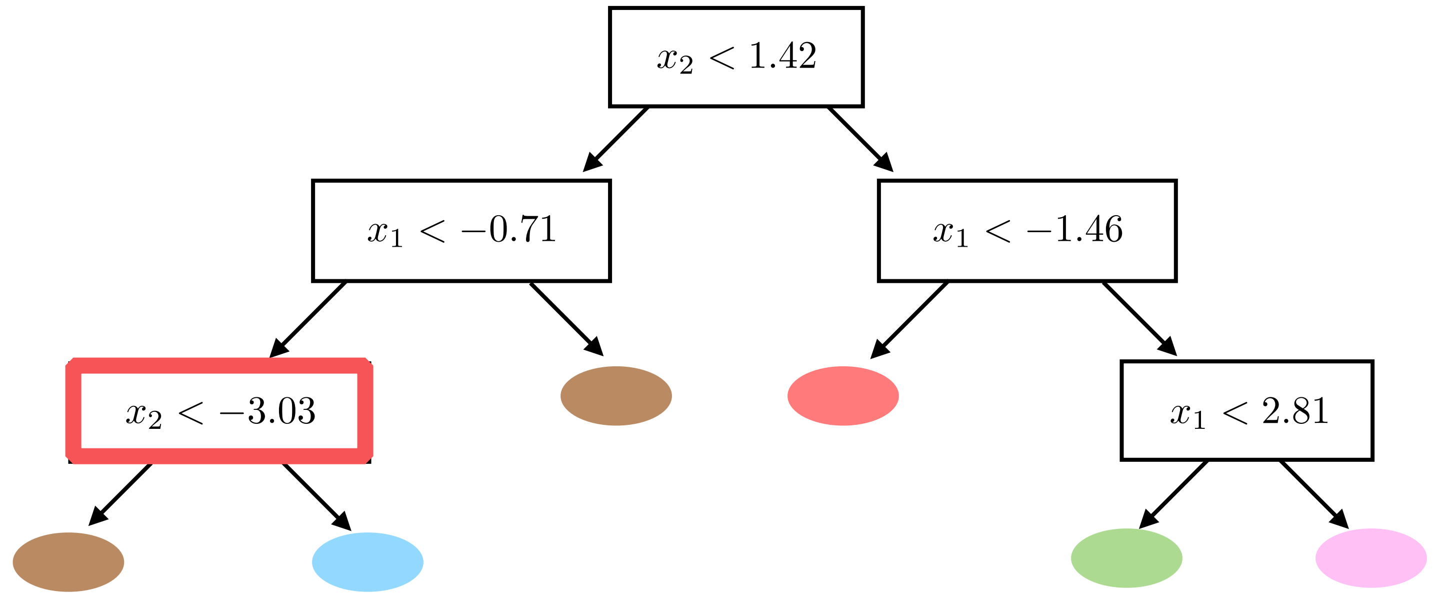


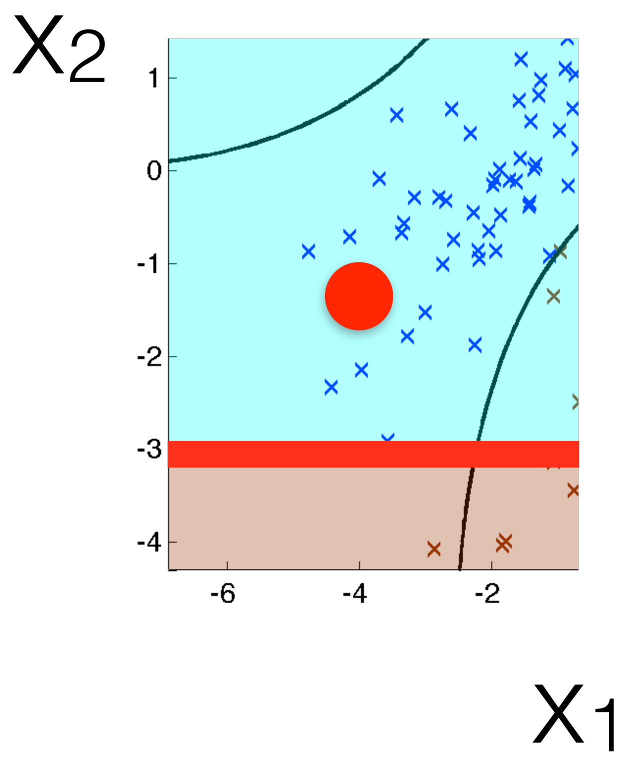
Decision Trees - Prediction



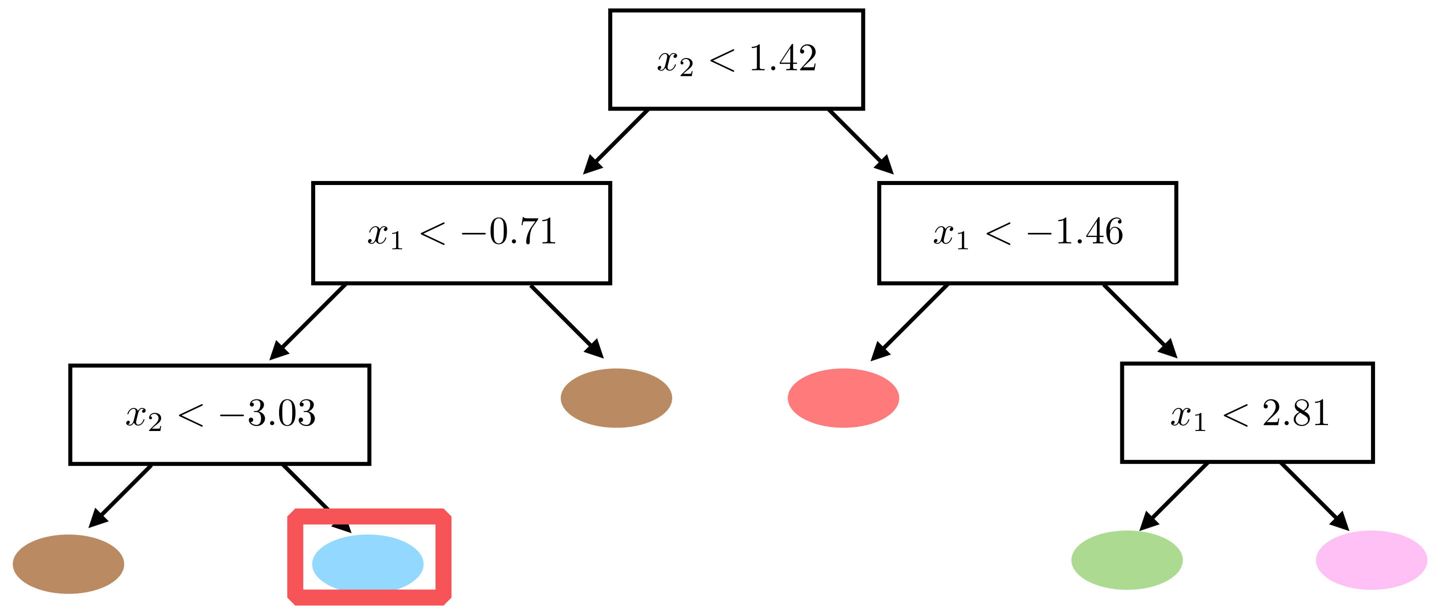


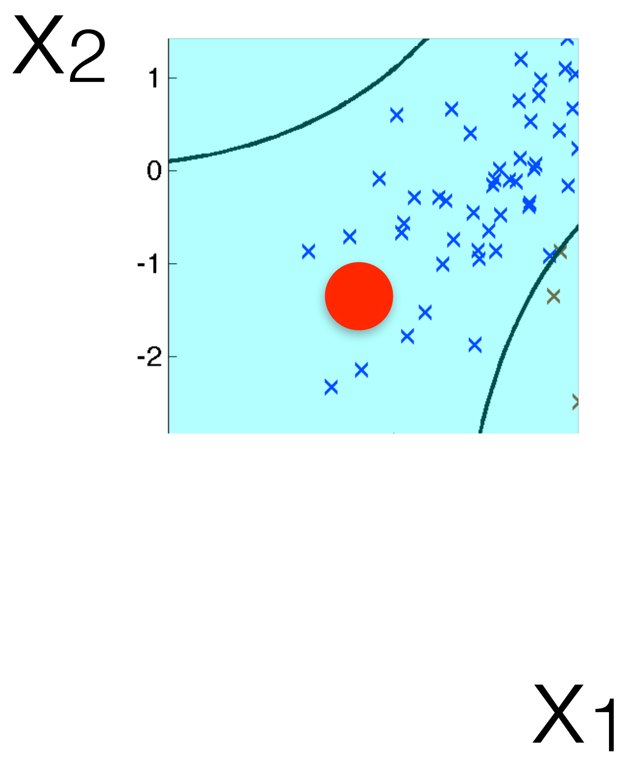
Decision Trees - Prediction





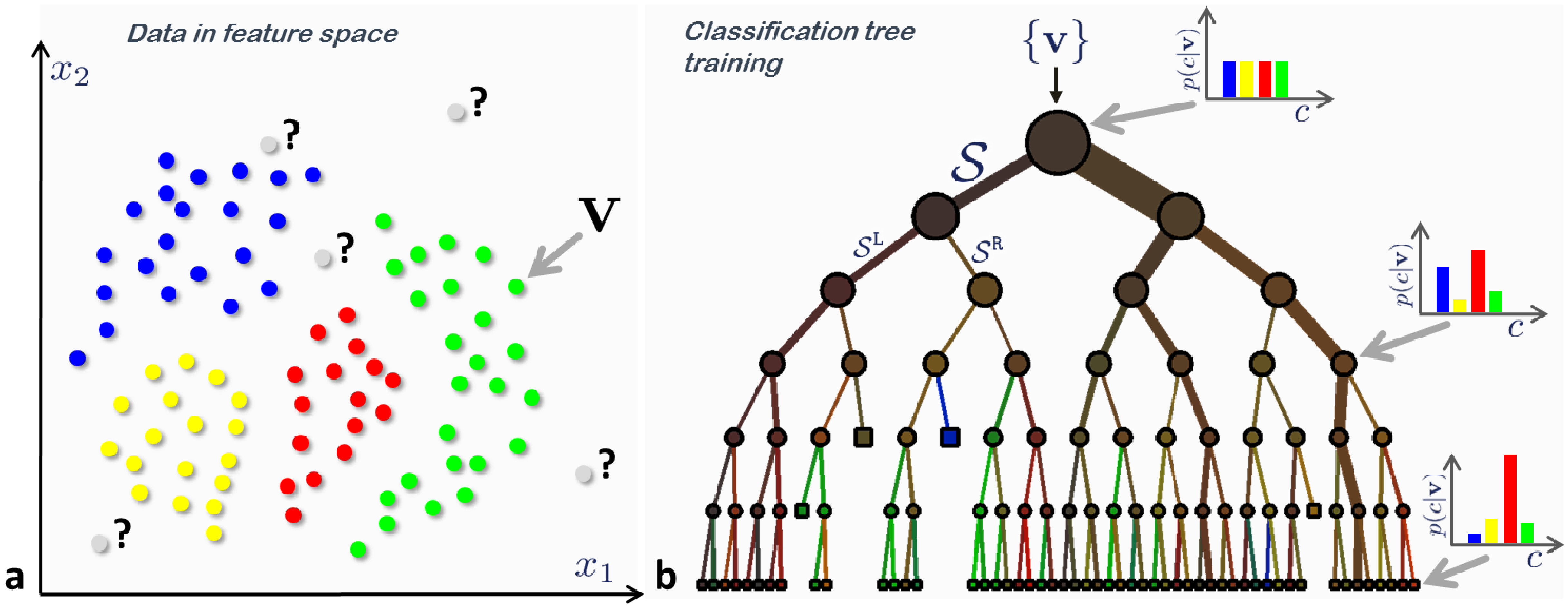
Decision Trees - Prediction





Learning a Decision Tree

What makes a good split?



[Criminsini et al 2011]

Shannon Entropy

- Measure of uncertainty
- Expected amount of information conveyed in a message or observation

$$H[Y] = -\sum_{k=1}^{K} p(y=k) \log_b p(y=k) \qquad Y \sim p(y)$$

- When b=2, measured in bits
- Higher entropy means that less is known about the outcome, thus corresponds to more evenly distributed probability
- Lower Entropy = More certainty in outcome

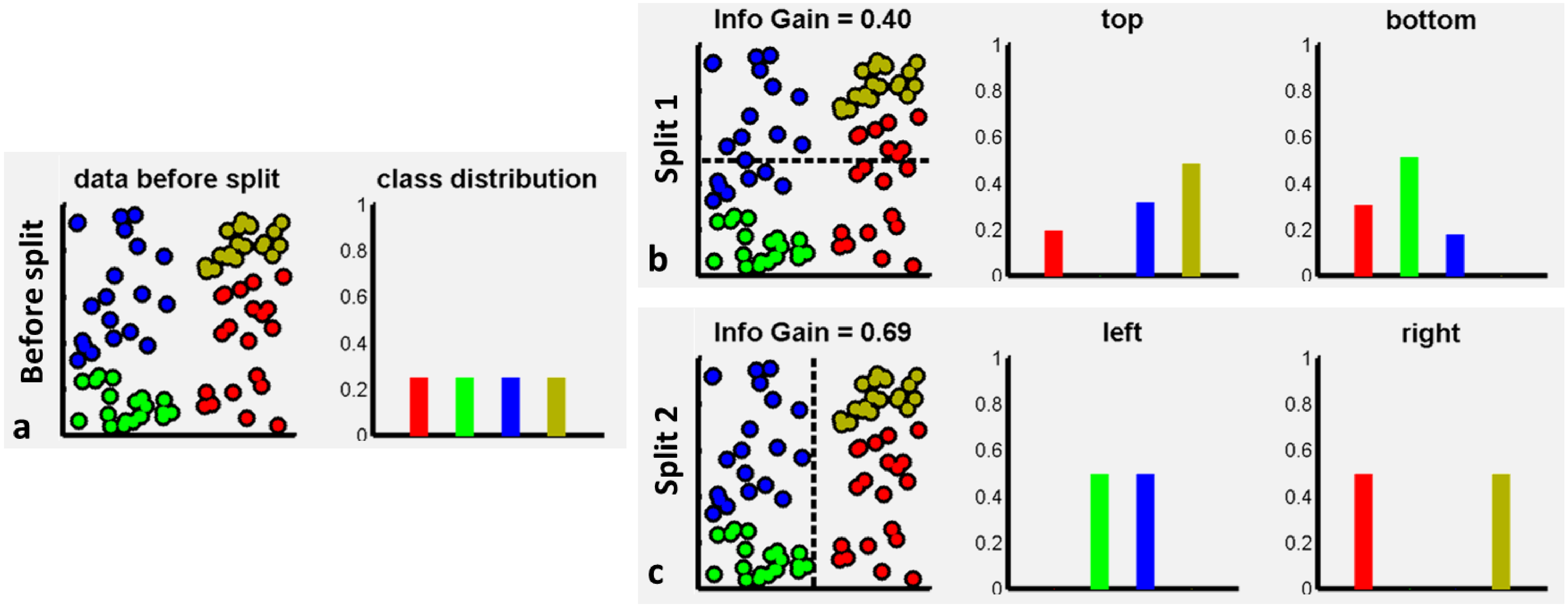
Information Gain

- The expected information gain is the reduction in entropy from one state to another
- Thus the information gain from a split is the entropy before the split minus the entropy after the split
- Need to weight the two post split entropies by the size of the respective nodes

$$IG = H[\mathcal{Y}] - \frac{N_{\text{left}}}{N} H[\mathcal{Y}_{\text{left}}] - \frac{N_{\text{right}}}{N} H[\mathcal{Y}_{\text{right}}]$$

- Calculate the entropies using the empirical distributions, i.e. p(y=k) is the proportion of class k at the node.
- Note that $0 \cdot \log(0) = 0$ so empty classes have no effect. This also means that the minimum entropy is 0 and occurs when only a single class

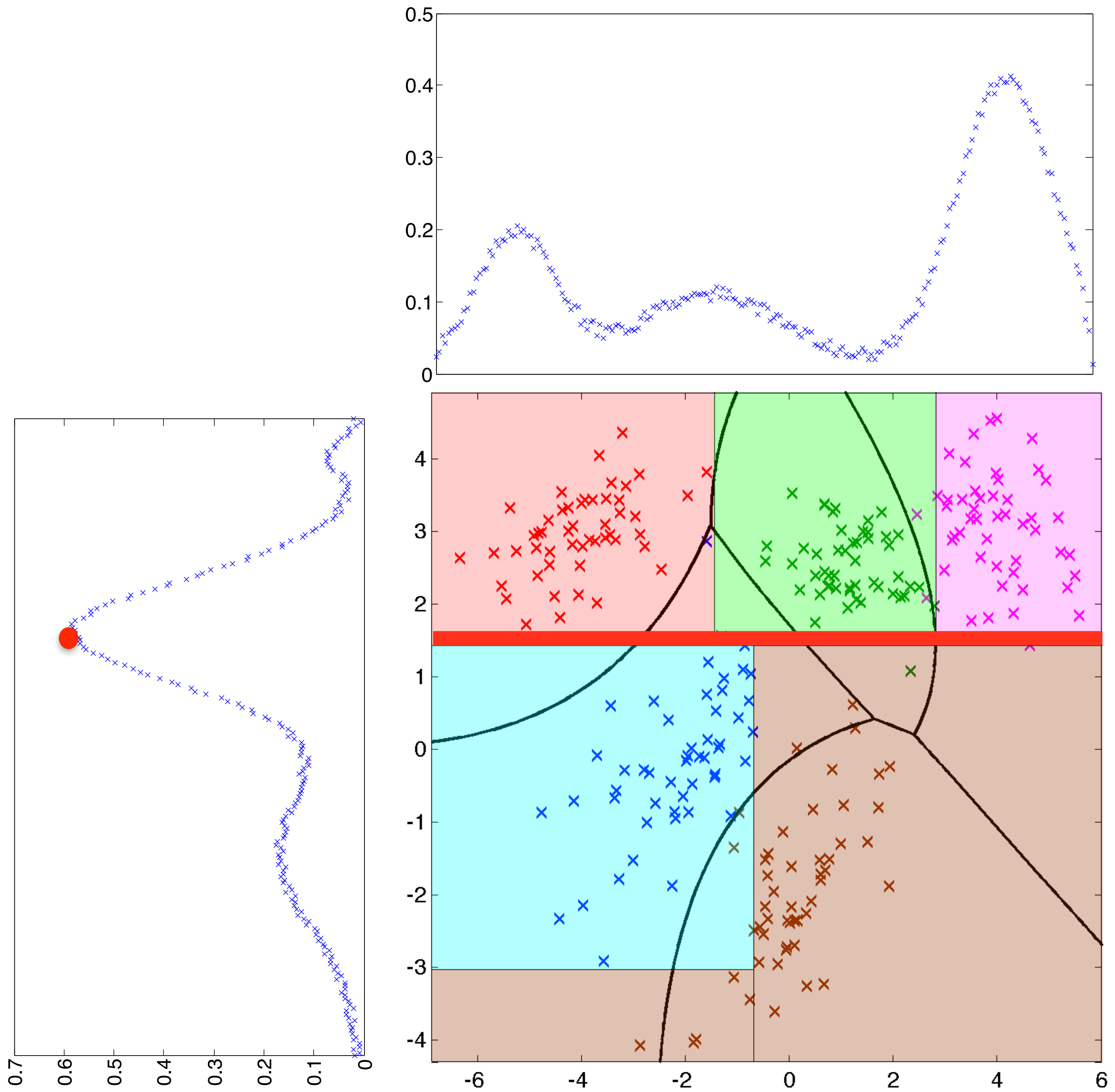
Splitting



[Criminsini et al 2011]

Splitting a Node - Exhaustive Search

- For each feature
  - Sort data by that feature
  - Calculate information gain resulting in putting the split between consecutive data points
- Choose the split (i.e feature + position) that gives the highest gain
- Place the split halfway between the two data points

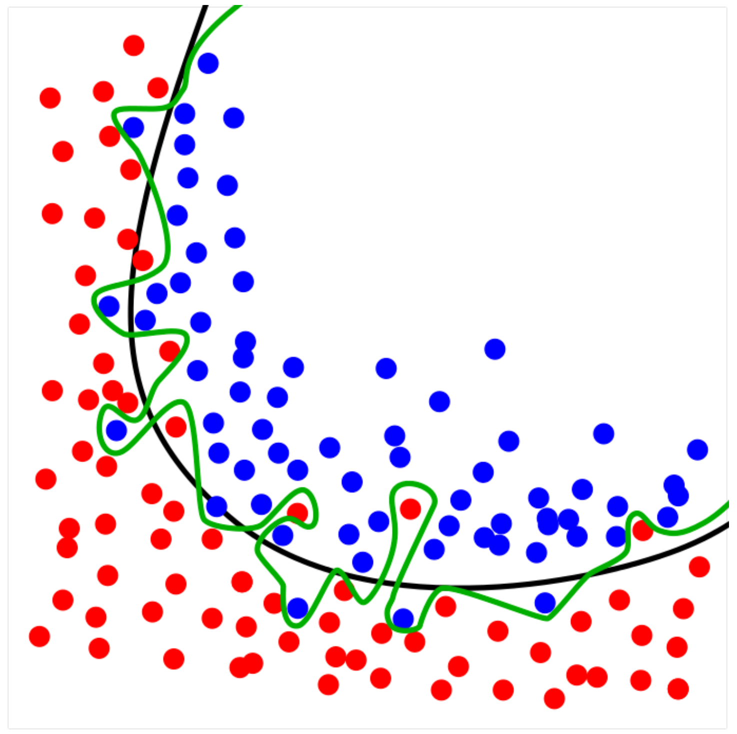


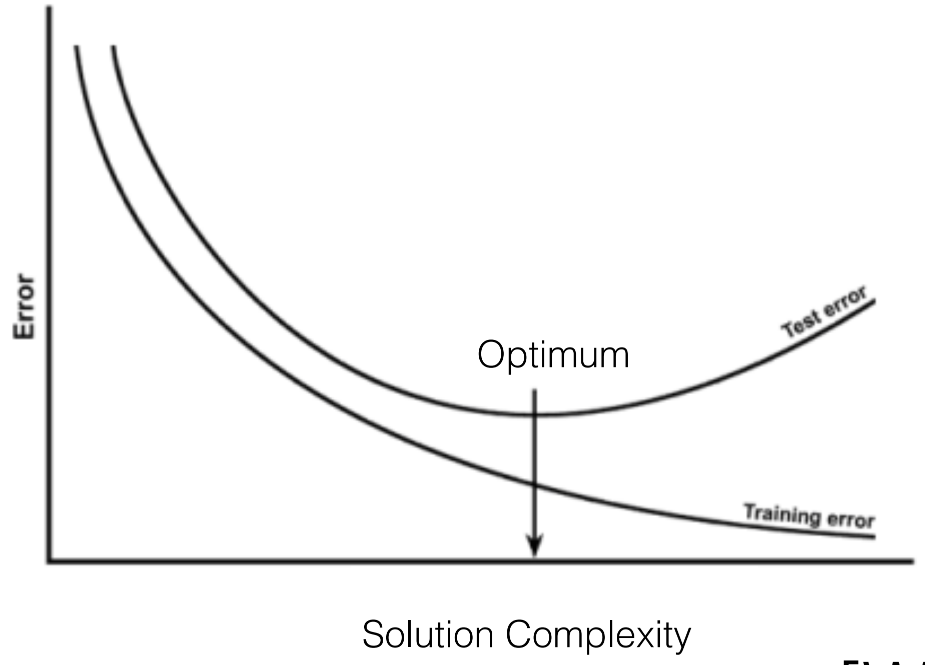
Training - Putting it Together

1. Start with a root node with all the data points
2. Exhaustively search possible splits and choose the best to create two new child nodes
3. Split the data into the new nodes. For each of these go back to 2 and grow in a self similar fashion
4. Continue until all points in a node have the same label or some other criterion is met (e.g. min number of points in a node).

Overfitting

- Best scoring solution perfectly separates the data
- This can clearly assign more structure than really exists
- Always possible to construct a function that fits the training data perfectly but does terribly on the test data
- Need to regularise to reduce the variance it estimates

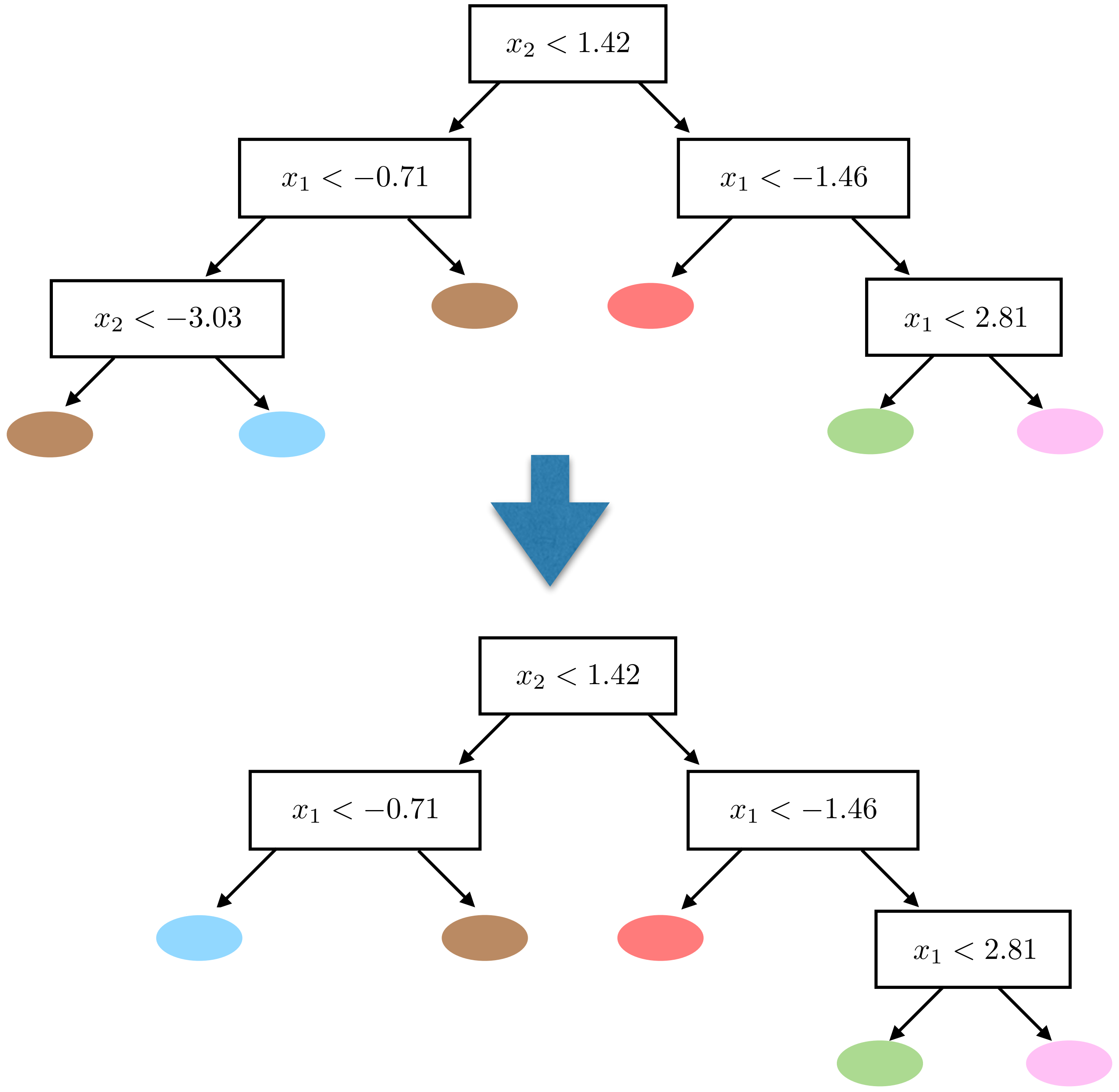




[Wikipedia]

Pruning

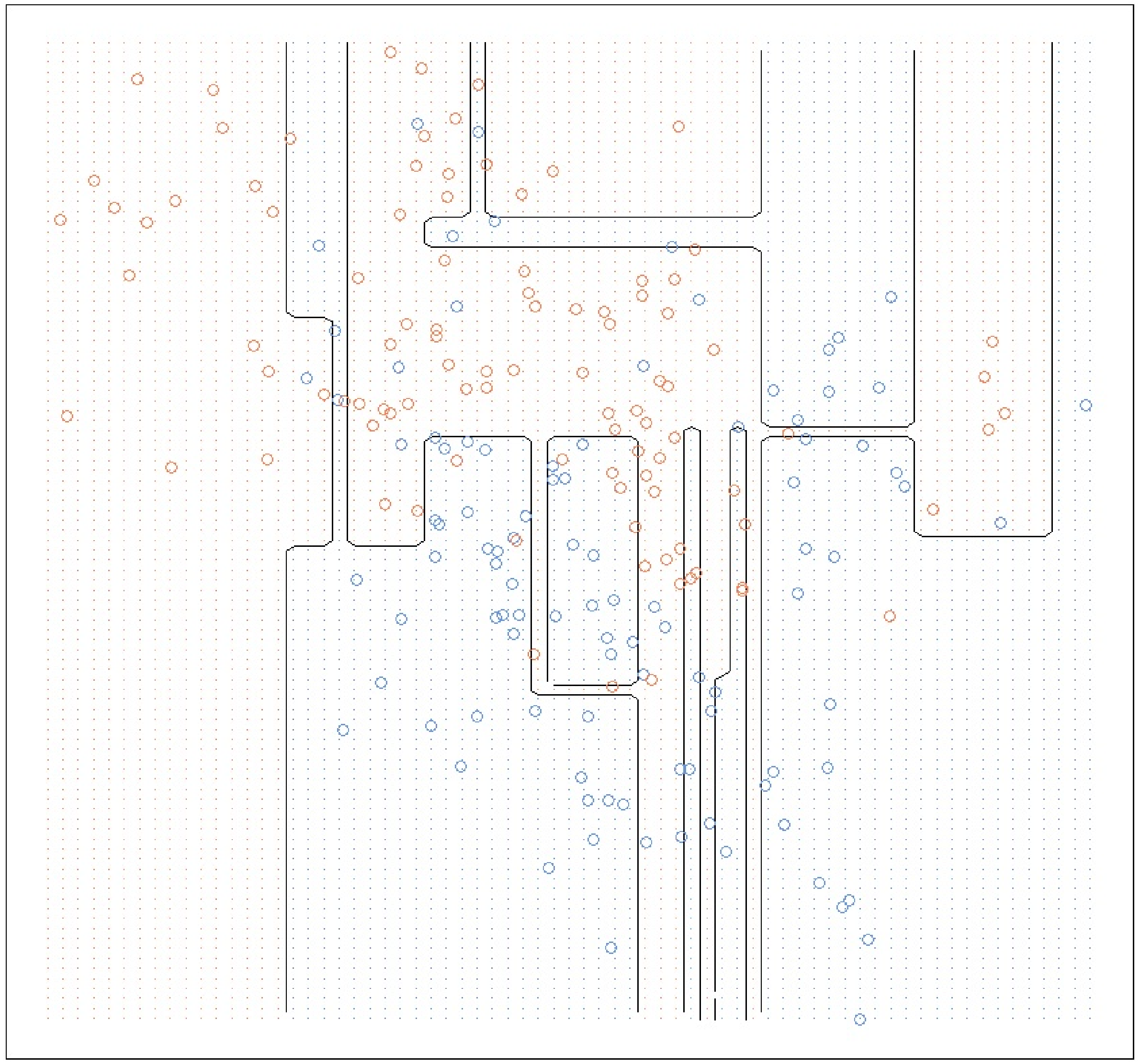
- Collapsing down some nodes to a single leaf node
- Start at the leaves and step upwards deciding whether to collapse based on some metric
- Smaller tree that is less prone to overfitting
- Computational expensive and unreliable

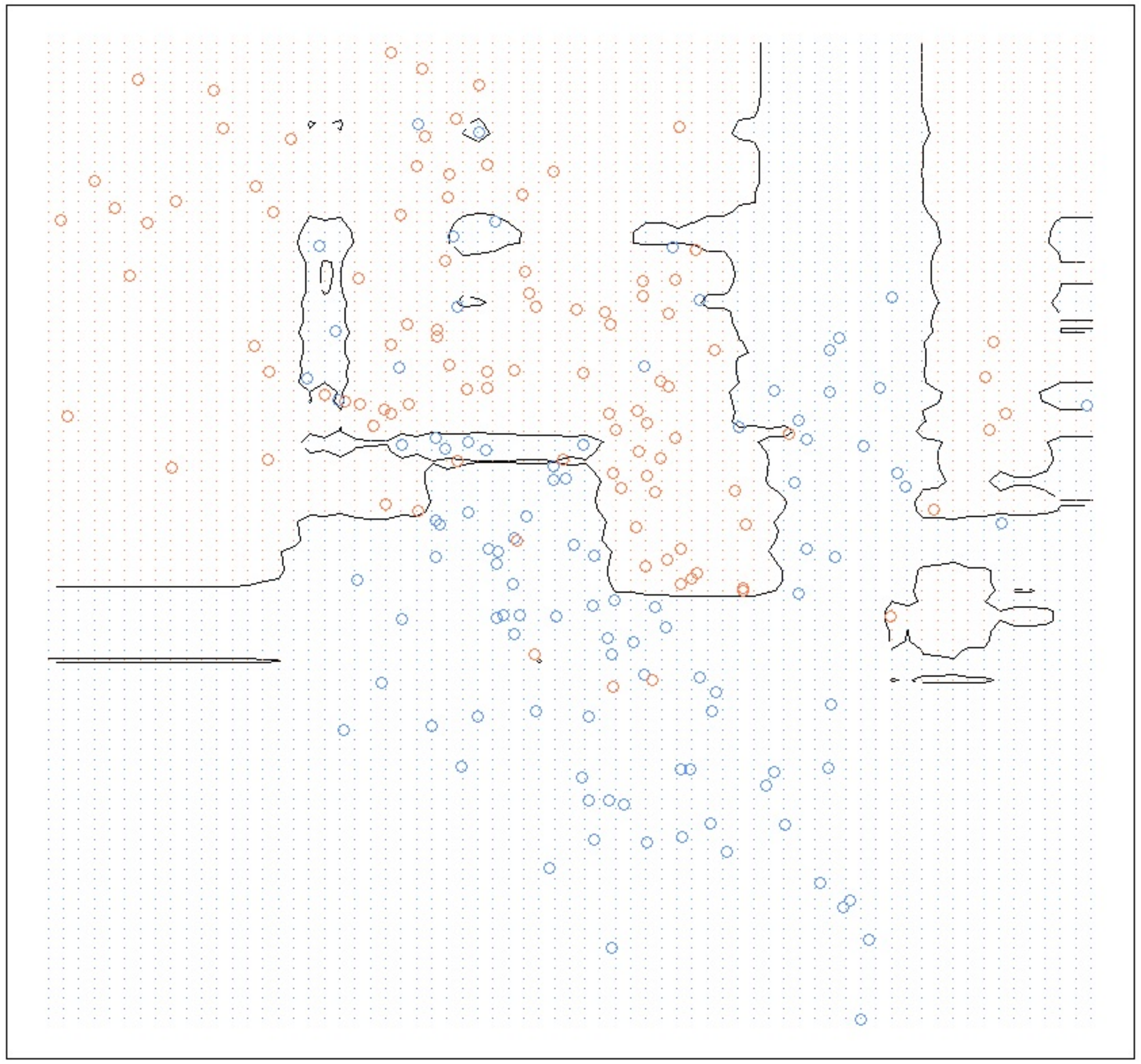


Tree Ensembles

A Better Approach: Ensembles

- Train lots of trees and average the predictions
- Reduces the variance on prediction / less overfitting- not all trees will make the same “mistakes”
- Fundamentally more powerful model - can have more complex decision boundaries
- Massive performance improvements
- Estimation of uncertainty - not all trees will make the same prediction, particularly near the boundaries
- No need to prune - can actually be faster!


(a) Single Tree


(c) **RF** (ntree=10000, mtry=1)

[Blaser & Fryzlewicz 2015]

Randomising Trees

- Training process previously described is completely deterministic and thus always produces the same tree

- Need to impart some randomness to the training process

- Trade-off between diversity of the ensemble and predictive power of individual trees

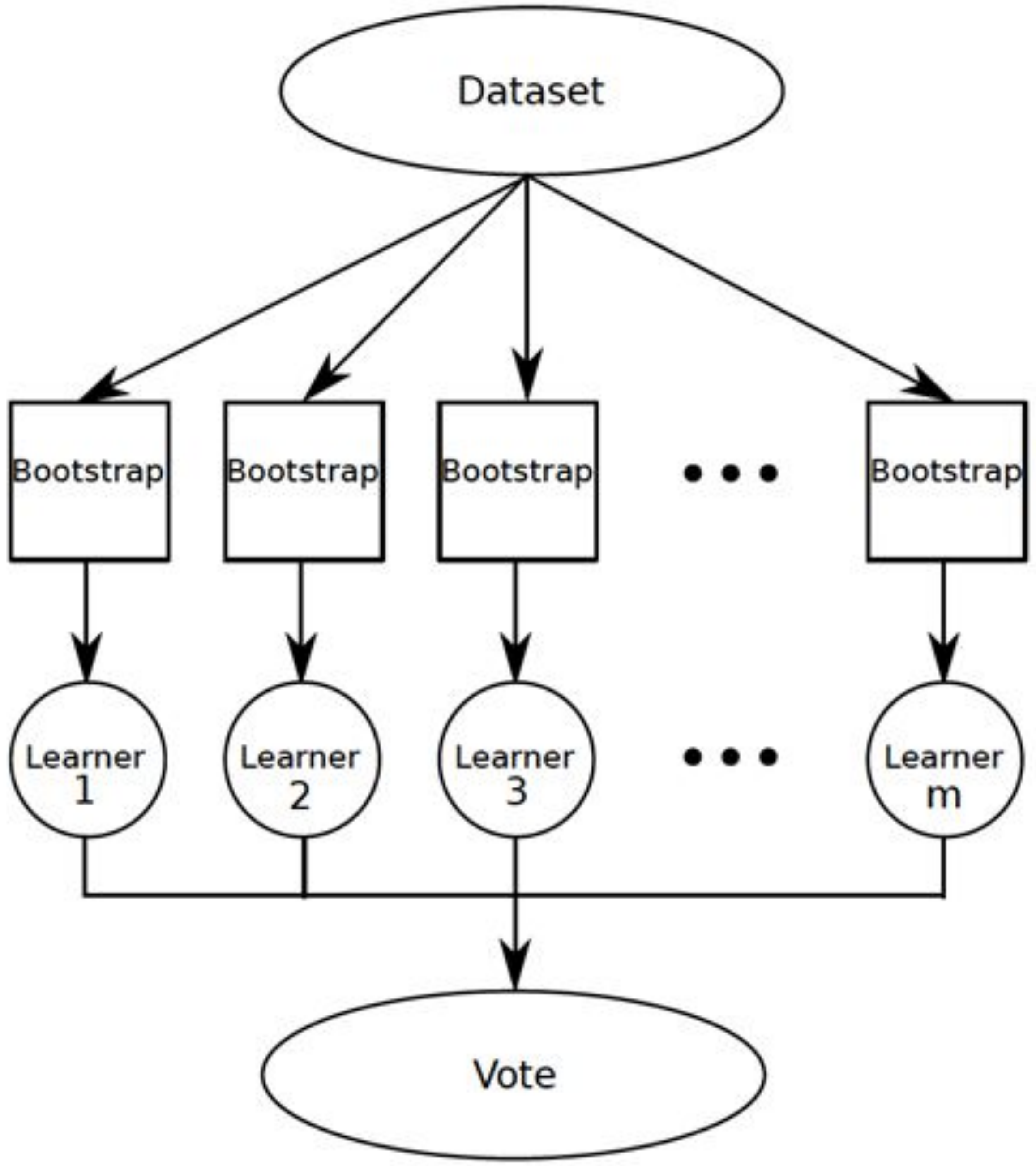
Bootstrap Sampling

- Sample with replacement a dataset of the same size
- Each sample is drawn independently from the full dataset

```
for n=1:N
    i ~ UniformDiscrete(1:N)
    $x'_n = x_i$
end
```

Bagging

- Train each tree using a slightly different dataset
- Generate datasets by taking bootstrap sampling - i.e. sample with replacement a dataset of the same size
- Predict by aggregating the predictions of the different trees



[Beatrix Mathews 2016]

Random Subspacing

- At **each** node randomly select some subspace of features to search over

```
1: Subsample features ids $\delta$ by sampling from $\{1, \ldots, D\}$ $\lambda$ times without replacement.
2: Set $\mathcal{X} \leftarrow X_{(:,\delta)}$
```

- This forces different trees to split along different dimensions at each node

- As this changes the partitioning at the first few nodes, this encourages further diversity at the latter nodes

- Typically $\lambda << D$

Random Forests

- Use bagging and feature subspacing to de-correlate trees
- Aggregate the predictions by voting
- One of the most successful algorithms of the 21st century with ~26000 citations
- Most people still use this “vanilla” version (better variants to come later)

Random Forest Algorithm in Full (1)

Algorithm 1: TRAIN RANDOM FOREST

Inputs: Features $X \in \mathbb{R}^{N \times D}$, classes $\mathcal{Y} \in \mathbb{I}^{N \times K}$, number of features to sample $\lambda \in \{1, \ldots, D\}$, number of trees to construct L

Outputs: Forest of L trees $\mathcal{T}_{1:L}$

```
1: for $\ell \in 1 : L$ do
2:     Construct bootstrap sample of data $\{X'_\ell, \mathcal{Y}'_\ell\}$ by sampling $N$ times with replacement from $\{X, \mathcal{Y}\}$
3:     $\mathcal{T}_\ell \leftarrow$ GROWTREE($X'_\ell, \mathcal{Y}'_\ell, \lambda$)
4: end for
```

See next slide

Random Forest Algorithm in Full (2)

Algorithm 2: GROWTREE

Inputs: Features $X \in \mathbb{R}^{N \times D}$, classes $\mathcal{Y} \in \mathbb{I}^{N \times K}$, number of features to sample $\lambda \in \{1, \dots, D\}$

Outputs: Subtree: a tuple {split dimension, split point, left branch, right branch} if root is a discriminant node, otherwise a class label $u \in 1^{1 \times K}$ representing a leaf.

```
 1: Subsample features ids $\delta$ by sampling from $\{1, \dots, D\}$ $\lambda$ times without replacement.
 2: Set $\mathcal{X} \leftarrow X_{(:,\delta)}$
 3: $E_{\text{base}} = \text{ENTROPY}(\mathcal{Y})$
 4: for $\nu = 1 : \lambda$ do
 5:     $u \leftarrow \text{SORT}(\mathcal{X}_{(:,\nu)})$
 6:     for $i = 2 : N$ do                          ▷ Exhaustive search on unique splits
 7:         $S_{i,\nu} \leftarrow (u_i + u_{i-1})/2$          ▷ Split halfway between consecutive points.
 8:         $j_\ell \leftarrow \{j \in \{1, \dots, N\} : \mathcal{X}_{j,\nu} \le S_{i,\nu}\}$   ▷ Index of data points that would take left branch
 9:         $j_r \leftarrow \{1, \dots, N\} \setminus j_\ell$
10:         $G_{i,\nu} = E_{\text{base}} - \frac{\|j_\ell\|_0}{N}\text{ENTROPY}(\mathcal{Y}_{(j_\ell,:)}) - \frac{\|j_r\|_0}{N}\text{ENTROPY}(\mathcal{Y}_{(j_r,:)})$   ▷ Information gain
11:     end for   ▷ In practise gain is calculated by iterating from previous value to avoid $N^2$ complexity.
12: end for
13: $\{i^*, \nu^*\} = \text{argmax}_{i,\nu}\, G_{i,\nu}$
14: if $G_{i^*,\nu^*} \le 0$ then
15:     return $\text{MODE}(\mathcal{Y})$   ▷ Node becomes a leaf with class set to the most common class
16: end if
17: $j_\ell = \{j \in \{1, \dots, N\} : U_{j,\nu^*} \le S_{i^*,\nu^*}\}$   ▷ Index of data points taking left branch
18: $j_r = \{1, \dots, N\} \setminus j_\ell$
19: return $\{\nu^*,\ i^*,\ \text{GROWTREE}(X_{(j_\ell,:)}, \mathcal{Y}_{(j_\ell,:)}, \lambda),\ \text{GROWTREE}(X_{(j_r,:)}, \mathcal{Y}_{(j_r,:)}, \lambda)\}$
```

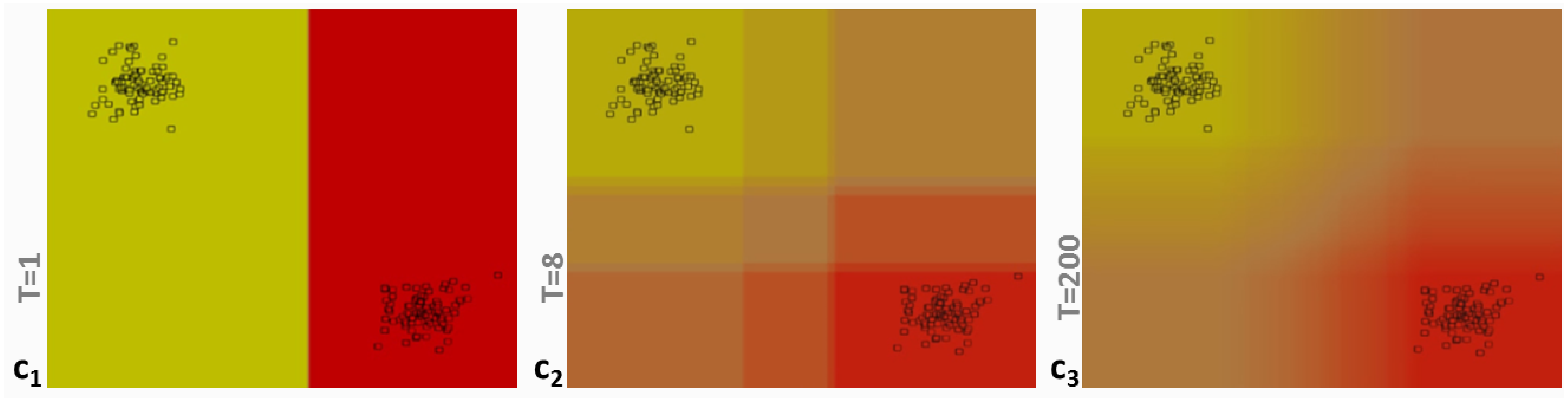
Practicalities

Uncertainty

- Each tree has its own prediction, often these predictions will disagree
- Can estimate probabilities rather by taking the proportion of trees that estimated that class

Number of Trees

- More the merrier - limited by computational budget
- 500 is relatively standard
- More trees gives a smoother surface



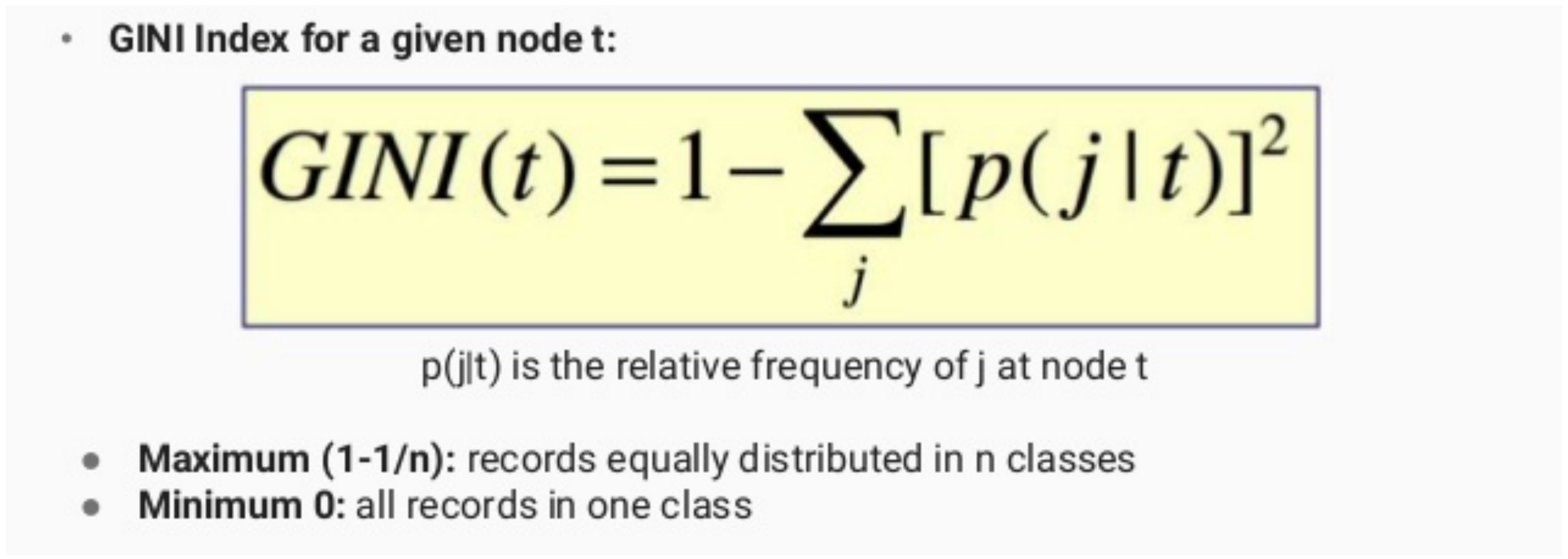
[Criminsini et al 2011]

Subspace Size

- Usually makes surprisingly little difference and often chosen for speed more than accuracy
- Avoid extreme values
- Common choices are $\lambda = D^{0.5}$ and $\lambda = \log_2 D$

Gini Split Criterion

- Sometimes people use the Gini split criterion instead of entropy



- GINI Index for a given node t:

$$GINI(t) = 1 - \sum_j [p(j|t)]^2$$

p(j|t) is the relative frequency of j at node t

- **Maximum (1-1/n):** records equally distributed in n classes
- **Minimum 0:** all records in one class

- Usually only makes minimal difference and neither is particularly considered to be better on average than the other

[Amr Barakat 2015]

Tree Ensembles vs Other Algorithms

Positives

- Very easy to use - 1 line of code
- Fast at train and test time
- State-of-the-art out-of-box performance for many problems
- Easily deal with different data types
- Very simple compared to most machine learning algorithms
- Decent robustness against overfitting
- Great as a baseline before trying some thing more involved

Negatives

- Little flexibility or ability to incorporate prior knowledge
- Typically worse performance then deep neural nets for huge datasets
- Vanilla RF poor on data with highly correlated features (CCFs and Rotations forests still good though)
- Can be quite dependent on the quality of the features used - typically helpful to preprocess data using some sort of feature extractor
- Provides uncertainty estimates but these often have poor accuracy
- Not good at extrapolation (though at the end of the day nothing really is)

Uneven Classes

- Need to be careful with any parameter tuning as predictive accuracy likely to be an unreliable metric - just going for the most common class will do well

- If you have more data than you can actually use, try to generate a more even sample as training data

- Alternative you can also create an artificial dataset with duplicate instances of the small classes to balance them out

Extensions

Regression

- Same principle but leaves have a local regression model (e.g. linear regression, Gaussian process) rather than a class
- Unlike classification, usually necessary to have a minimum number of points in each leaf node for this local model
- Less common than classification but still powerful

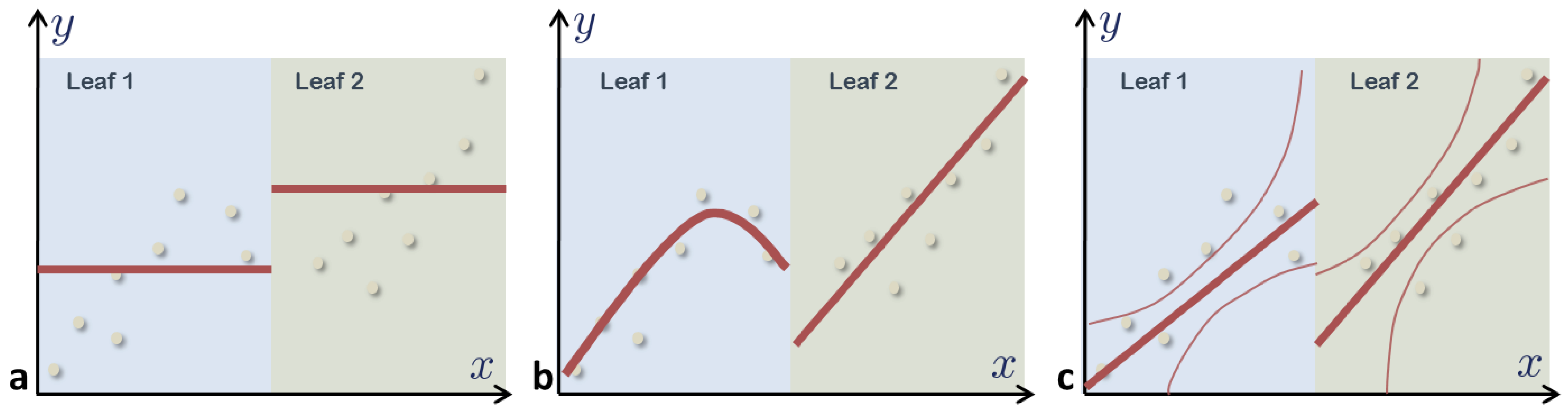


Fig. 4.2: **Example predictor models.** Different possible predictor models. **(a)** Constant. **(b)** Polynomial and linear. **(c)** Probabilistic-linear. The conditional distribution $p(y|x)$ is returned in the latter.

[Criminsini et al 2011]

Regression (2)

- Use different split criterion, e.g. total variance instead of entropy

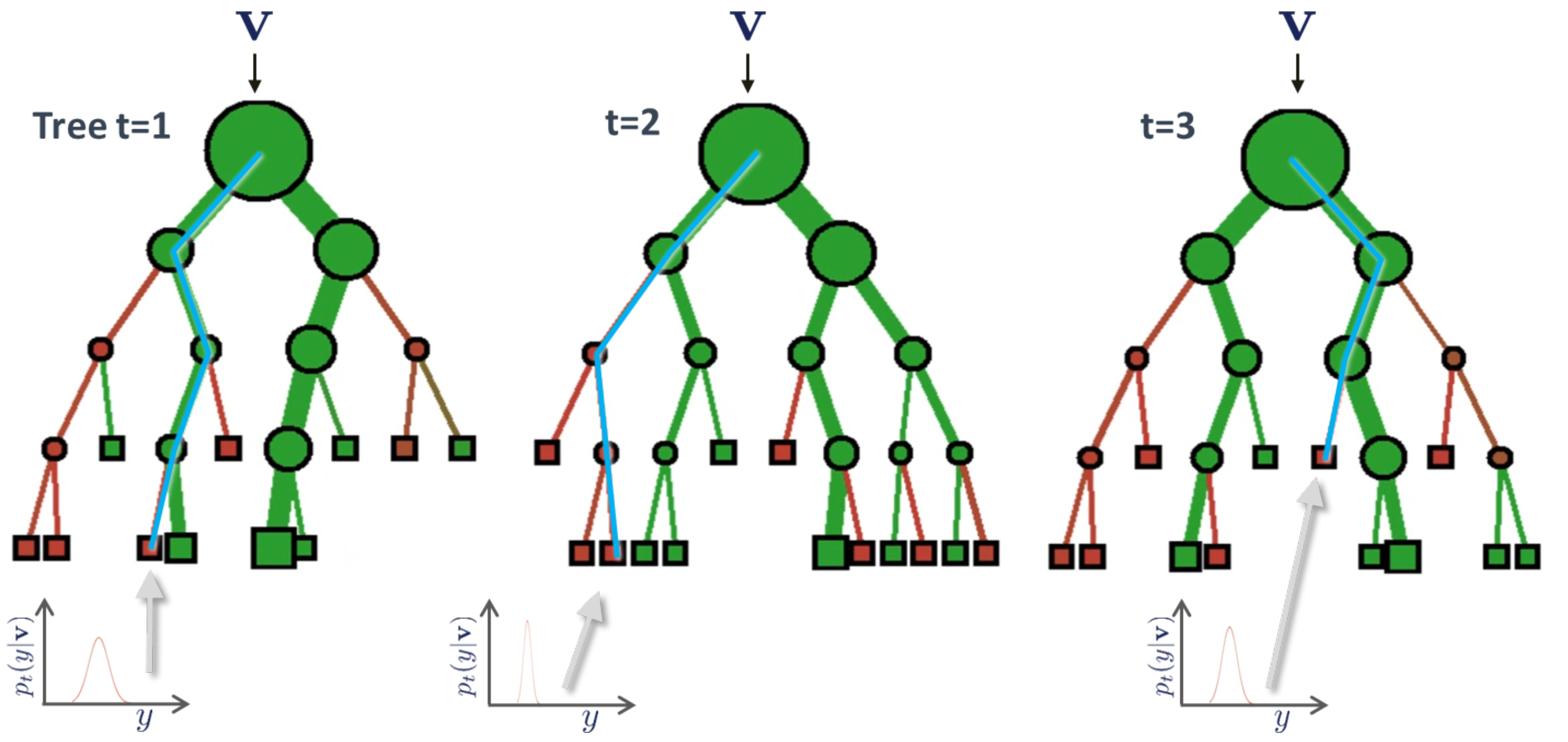
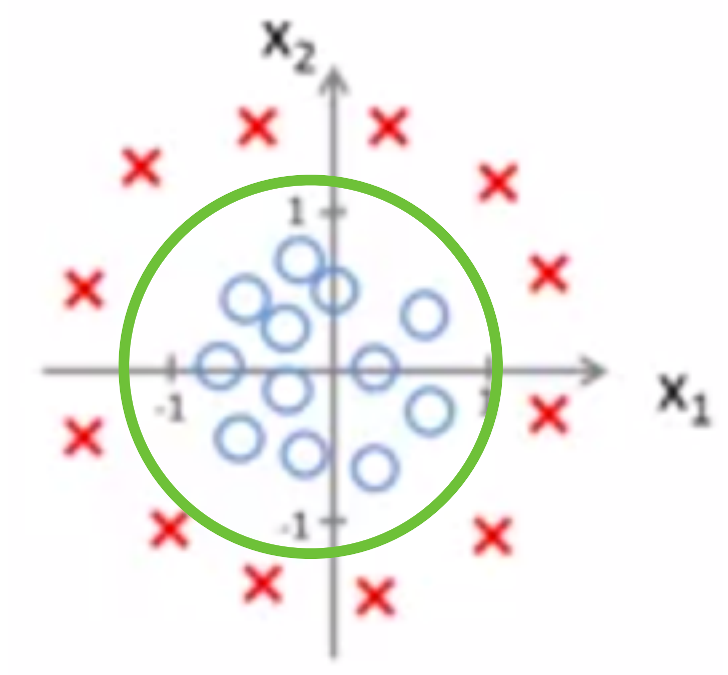


Fig. 4.3: **Regression forest: the ensemble model.** The regression forest posterior is simply the average of all individual tree posteriors $p(\mathbf{y}|\mathbf{v}) = \frac{1}{T}\sum_{t=1}^{T} p_t(\mathbf{y}|\mathbf{v})$.

[Criminsini et al 2011]

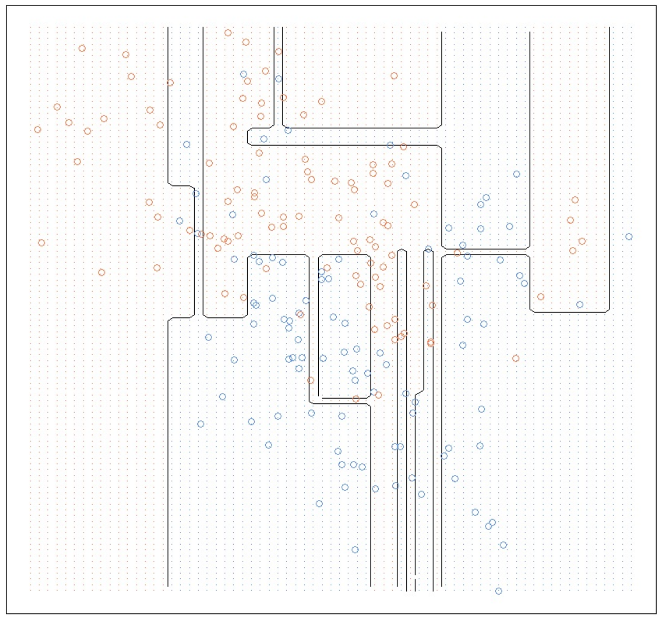
Custom Features

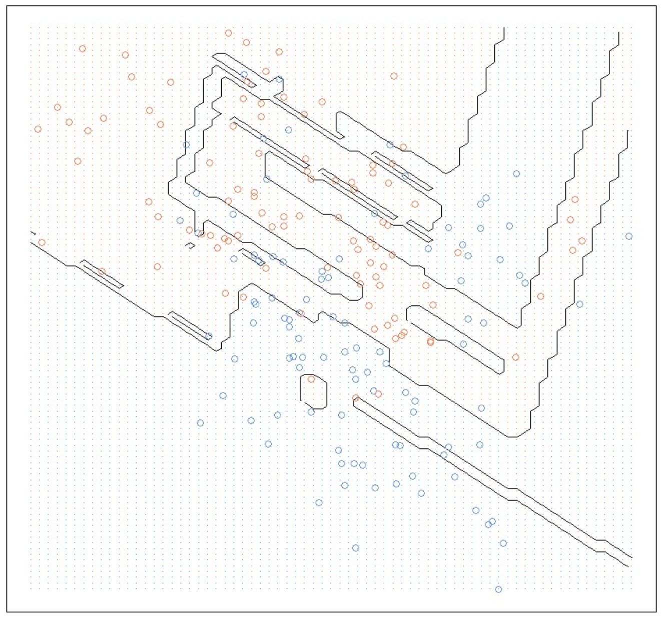
- Instead of operating on the input data X directly, construct additional features X<-[X,g(X)]
- This allows more complex splits in the original space
- For example, $g(X) = (x_1\text{-}a)^2+(x_2\text{-}b)^2$ allows splitting on the distance from a point (a,b)
- More generally, use the output from some nonlinear feature generator such as a neural net

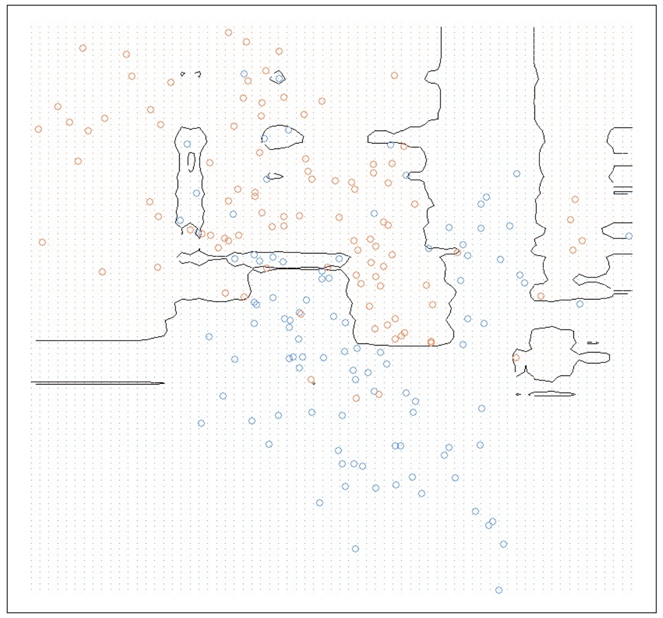


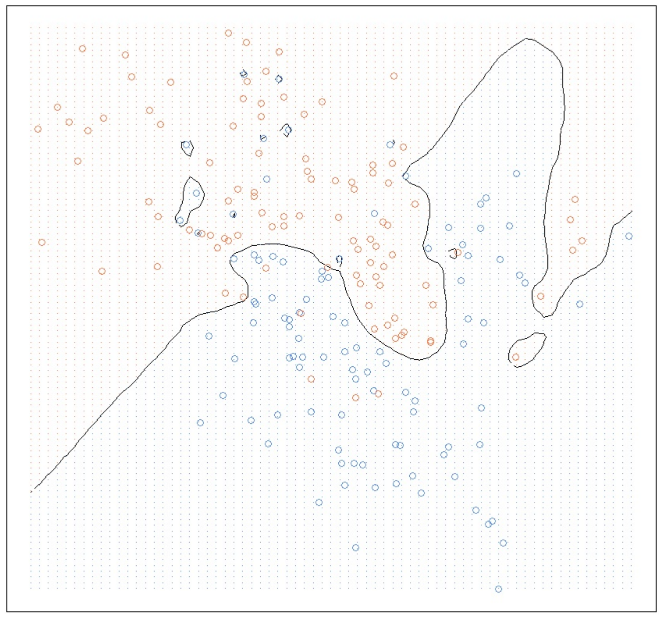
Random Rotations

- Randomly rotate the dataset separately for each tree
- Reduces correlation between trees which can improve performance
- No longer restricted to piecewise linear decision surfaces => smoother
- Can damage performance when little correlation between features


(a) **RF** (ntree=1, mtry=1)


(b) **RR-RF** (ntree=1, mtry=1)


(c) **RF** (ntree=10000, mtry=1)


(d) **RR-RF** (ntree=10000, mtry=1)

[Blaser & Fryzlewicz 2015]

Random Projections

- Similar to random rotations, but applied at **each** node separately, rather than globally for each tree
- Again can give performance improvements or worsen performance depending on dataset
- On average does slightly better than random forests

Rotation Forests

- Instead of projecting randomly, use principle component analysis (PCA) on small groups of randomly sampled features
- Significantly reduces sensitivity to correlation between features, without damaging performance when there is little correlation
- Typically does not use random subspacing => significantly slower
- Can give large performance improvements, particularly when strong correlation

[Rodrigeuz et al 2006]

Extremely Randomized Trees

- Choose splits randomly rather than in a principled manner

- Amazingly can actually improve performance compared with random forests in some situations

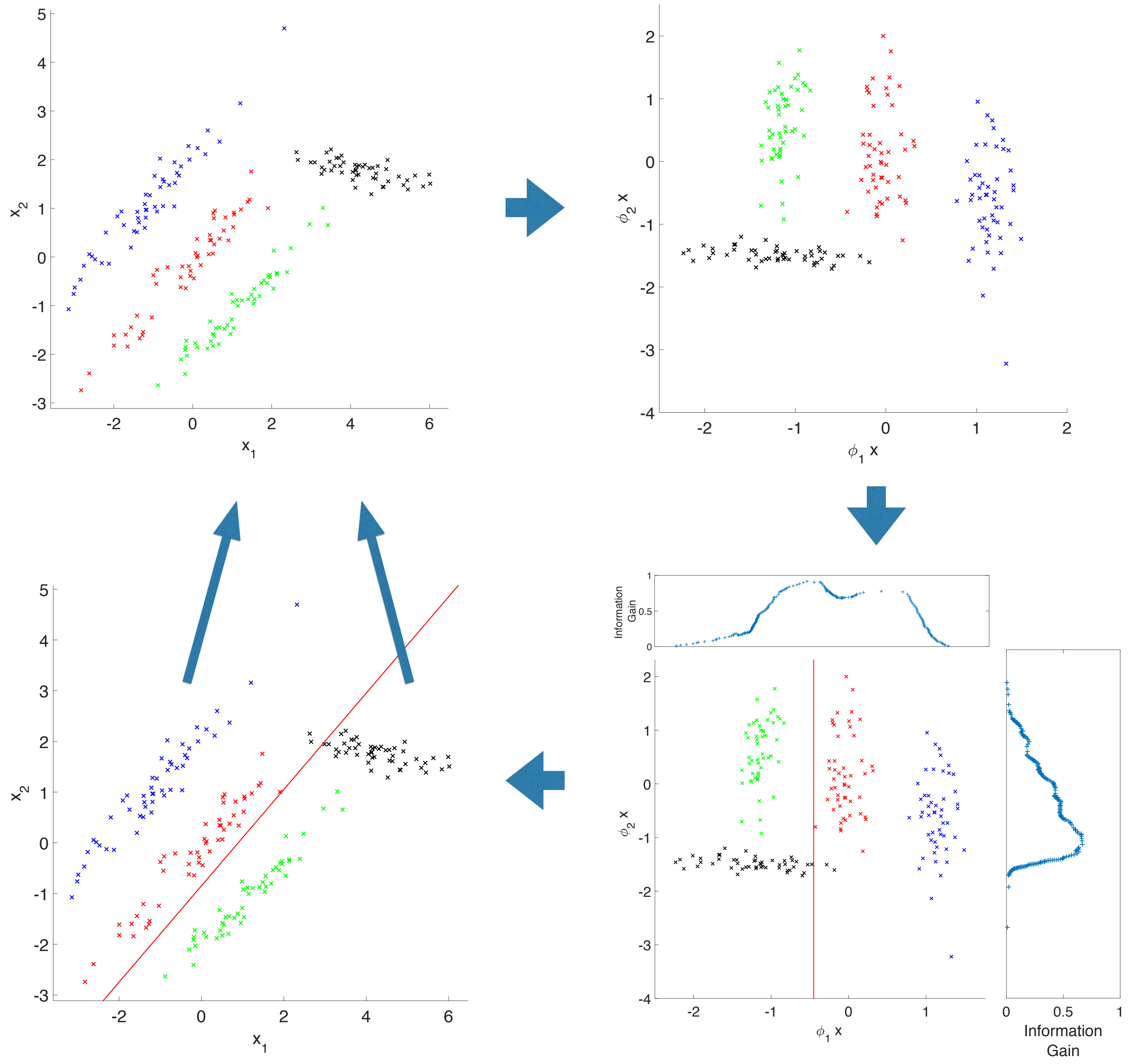
- Comparable performance on average

[Geurts et al 2006]

Canonical Correlation Forests

- Project to maximally de-correlated space at **each** node using canonical correlation analysis (CCA)

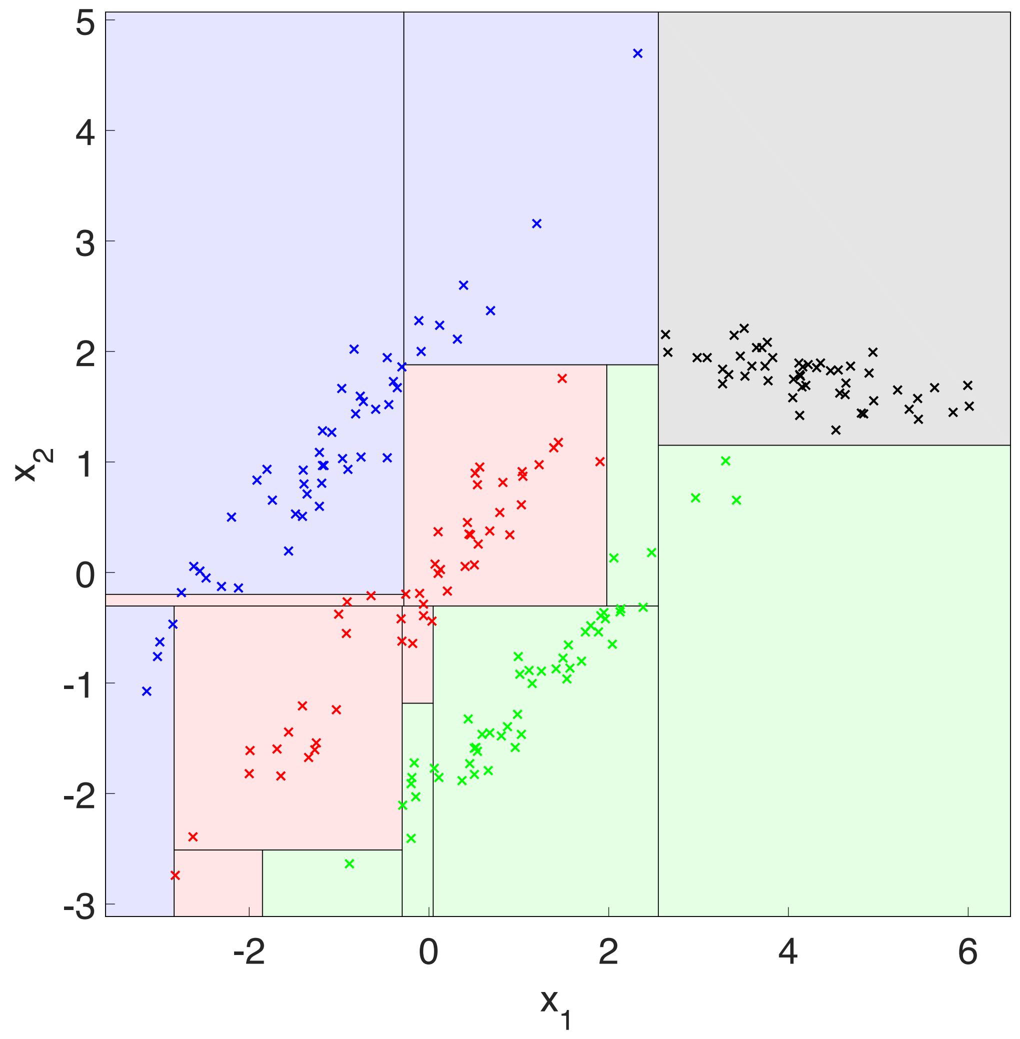
- Speed of random forests but with big improvements in accuracy, particularly with correlated data

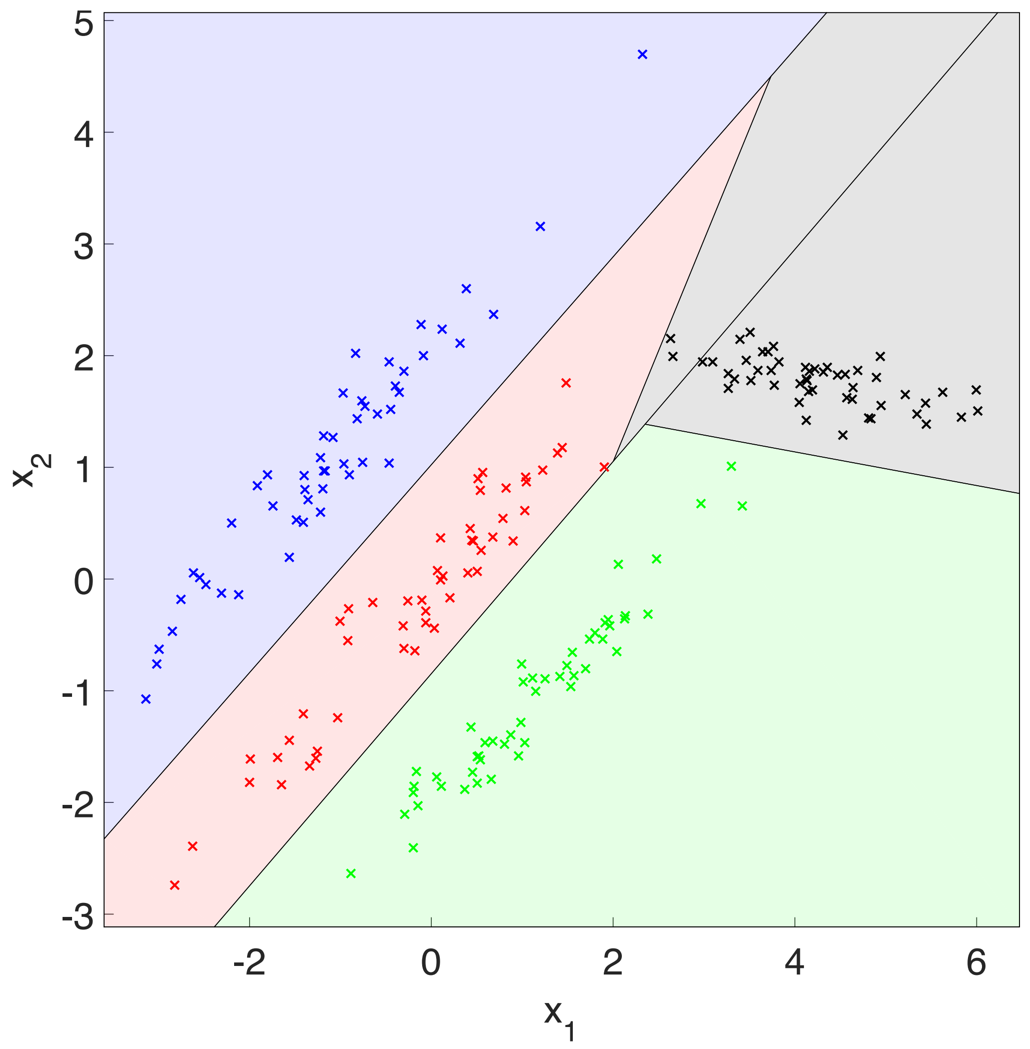


[Rainforth and Wood 2015]

Canonical Correlation Forests (2)

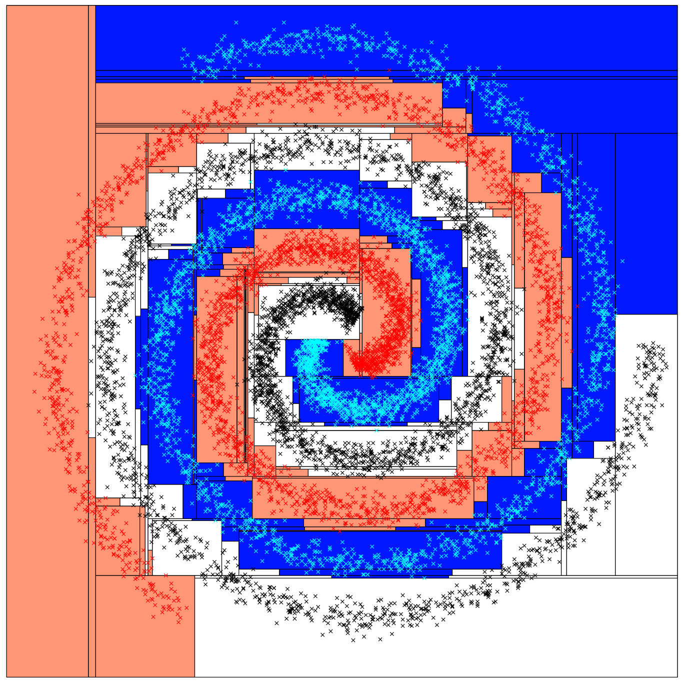
- Not being axis aligned means each tree can fit the data better
- Also reduces correlation between trees predictions which improves accuracy further



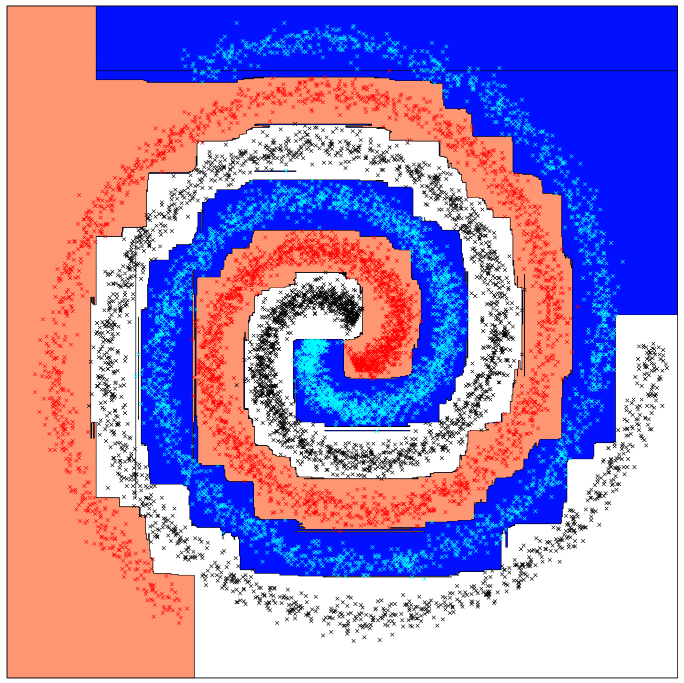


Canonical Correlation Forests (3)

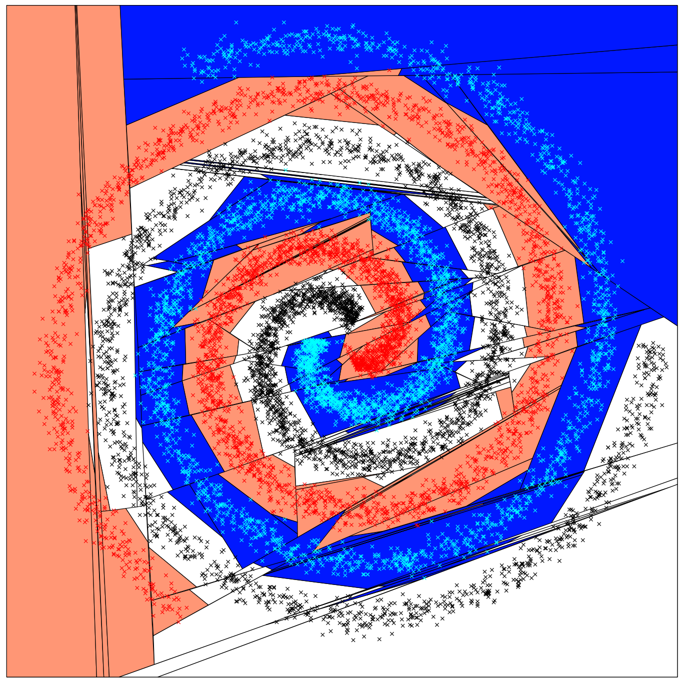
(a) Single CART



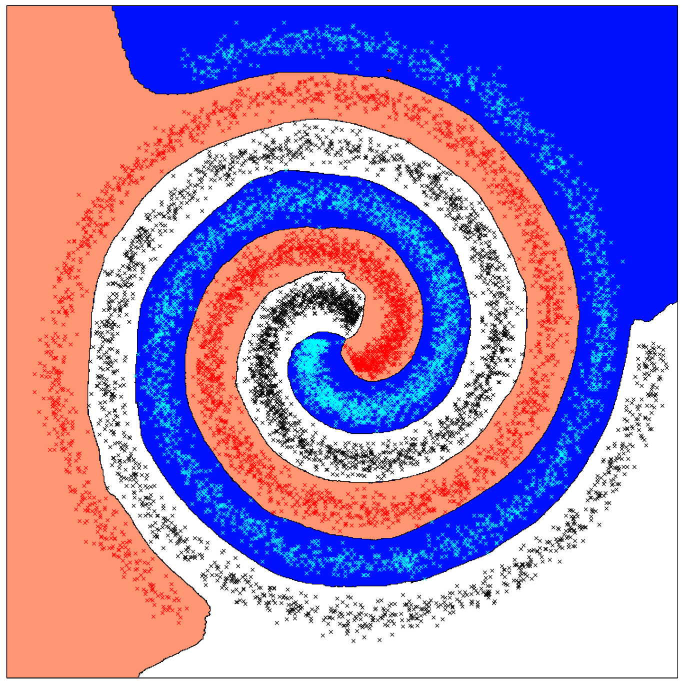
(b) RF with 200 Trees



(c) Single CCT



(d) CCF with 200 Trees



Uneven Voting

- Some trees may better than others
- Can get small performance improvements by not weighting trees evenly
- For example weighting simpler trees higher or using cross-validation schemes to calculate weights

[Robnik-Sikonja 2004]

Useful Packages

- Weka (stand alone) - http://www.cs.waikato.ac.nz/ml/weka/ - Gui with java back end. Operates on csv files and allows lots of algorithms to be used at the same package

- Scikit learn (python) - http://scikit-learn.org/stable/ - open source package with lots of machine learning algorithms built in.

- TreeBagger (Matlab) - https://uk.mathworks.com/help/stats/treebagger.html - in built random forest package

- randomforest-matlab - https://code.google.com/archive/p/randomforest-matlab/ - Faster open source matlab version

Useful Packages (2)

- Canonical correlation forests (matlab, python) - https://bitbucket.org/twgr/ccf (matlab), https://github.com/asross/oo_trees (python, not our implementation so no guarantees) - our algorithm, state-of-the-art predictive accuracy (classification only)

- C++ - https://github.com/bjoern-andres/random-forest

- R - https://www.tutorialspoint.com/r/r_random_forest.htm

Further Reading

- Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning. Criminisi et al 2014. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/decisionForests_MSR_TR_2011_114.pdf

- Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.

- Nando de Freitas' lectures - https://youtu.be/-dCtJjlEEgM and https://youtu.be/3kYujfDgmNk

- A good high level introduction - https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/

- Self plug - Canonical correlation forests. Rainforth and Wood 2015. arXiv preprint arXiv:1507.05444 (2015). https://arxiv.org/pdf/1507.05444.pdf

Thanks for listening, any questions?

Feel free to email me at twgr@robots.ox.ac.uk