Introduction to Machine Learning
HPC IA — 2019-20

8. Tree-based approaches

Chloé-Agathe Azencott
Centre for Computational Biology, Mines ParisTech
chloe-agathe.azencott@mines-paristech.fr

Inserm
Institut national
de la santé et de la recherche médicale

institutCurie

PSL
RESEARCH UNIVERSITY PARIS

MINES
ParisTech

Learning objectives

- Build **decision trees:**
  - Decide how to grow a tree
  - Decide when to stop growing a tree
- Explain why they are examples of **non-metric learning** and **hierarchical learning.**
- **Combine decision trees** (or other **weak learners**) to make more powerful classifiers.

Decision trees

Hierarchical learning

- **Single-stage** classifiers:
  - Assign a class to object **x** using a single operation.
  - Use a single set of features for all classes.
  - Difficulties:

    - when classes have **multi-modal** distributions

    - when features are **nominal.**
- **Hierarchical** classifiers:

  Multiple successive tests.

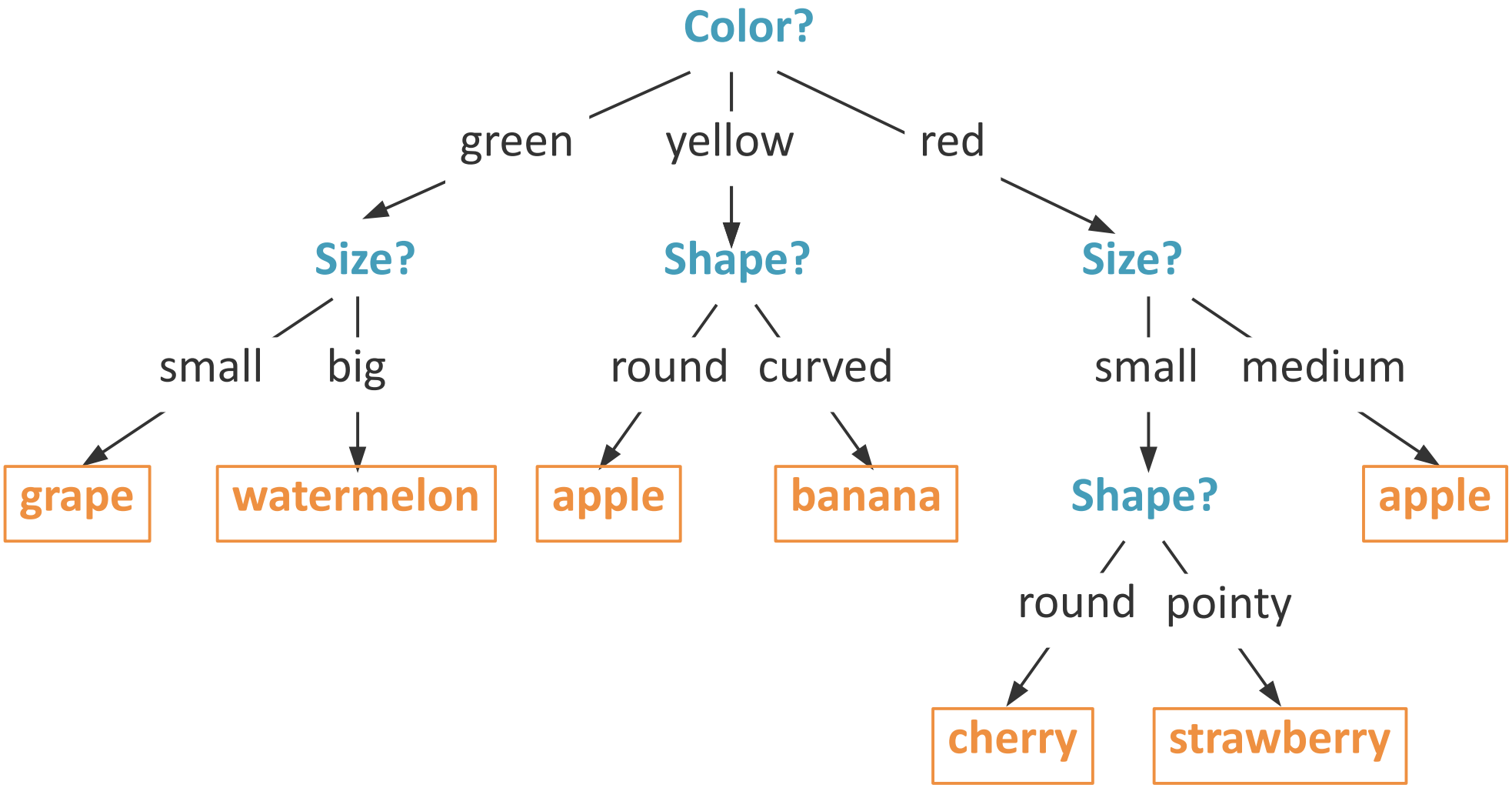
Nominal data

- Attributes that are
  - **Discrete**
  - Without any natural notion of **similarity/ordering**

    → **Non-metric learning.**
- Example:

  Classify fruit from {color, shape, texture, size}.

Decision trees: The 20Q game



Multiclass classification

- **One-versus-all**

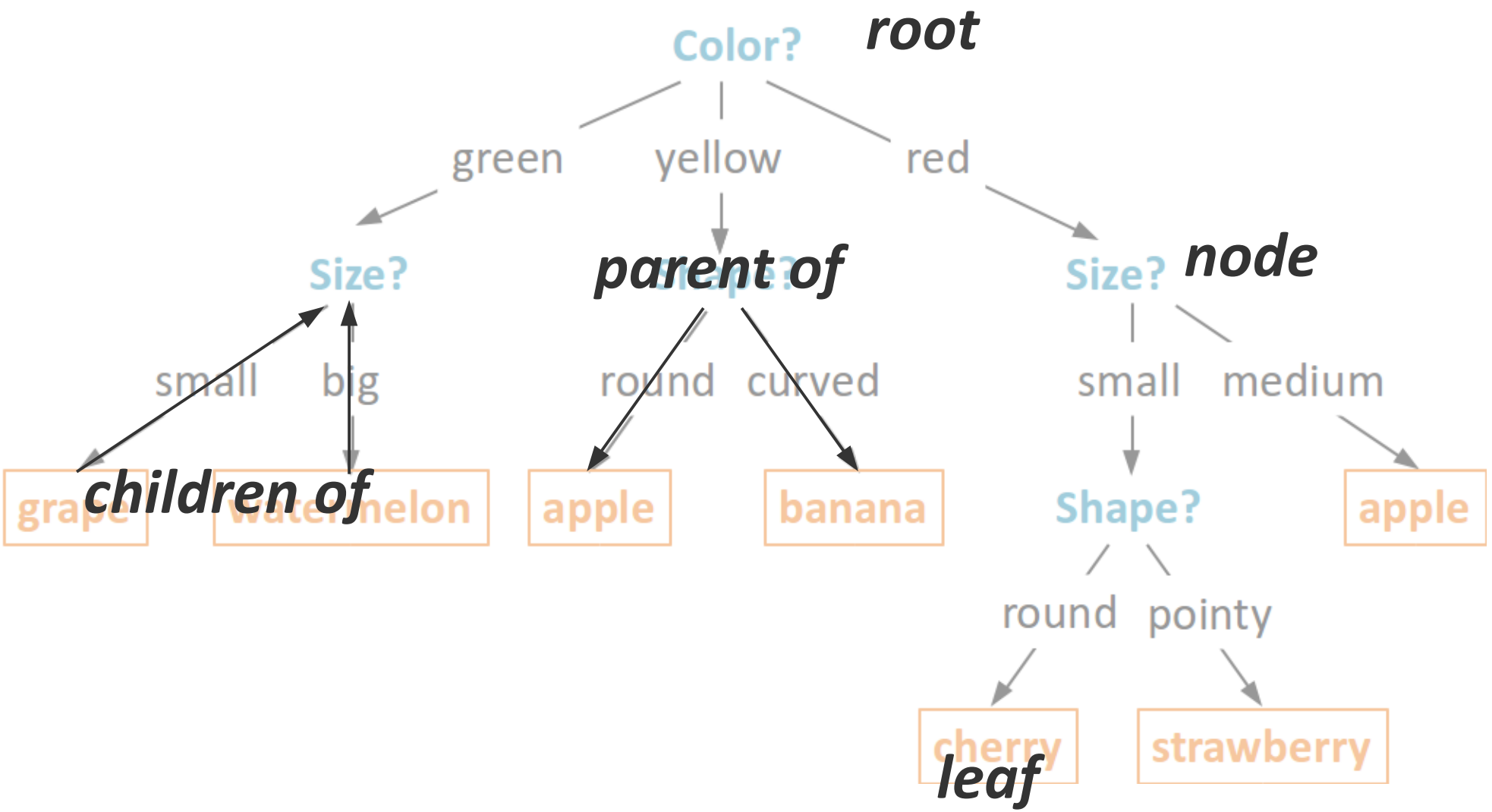
  Build K classifiers, make them vote.

- **One-versus-one**

  Build K (K-1) / 2 classifiers, make them vote.

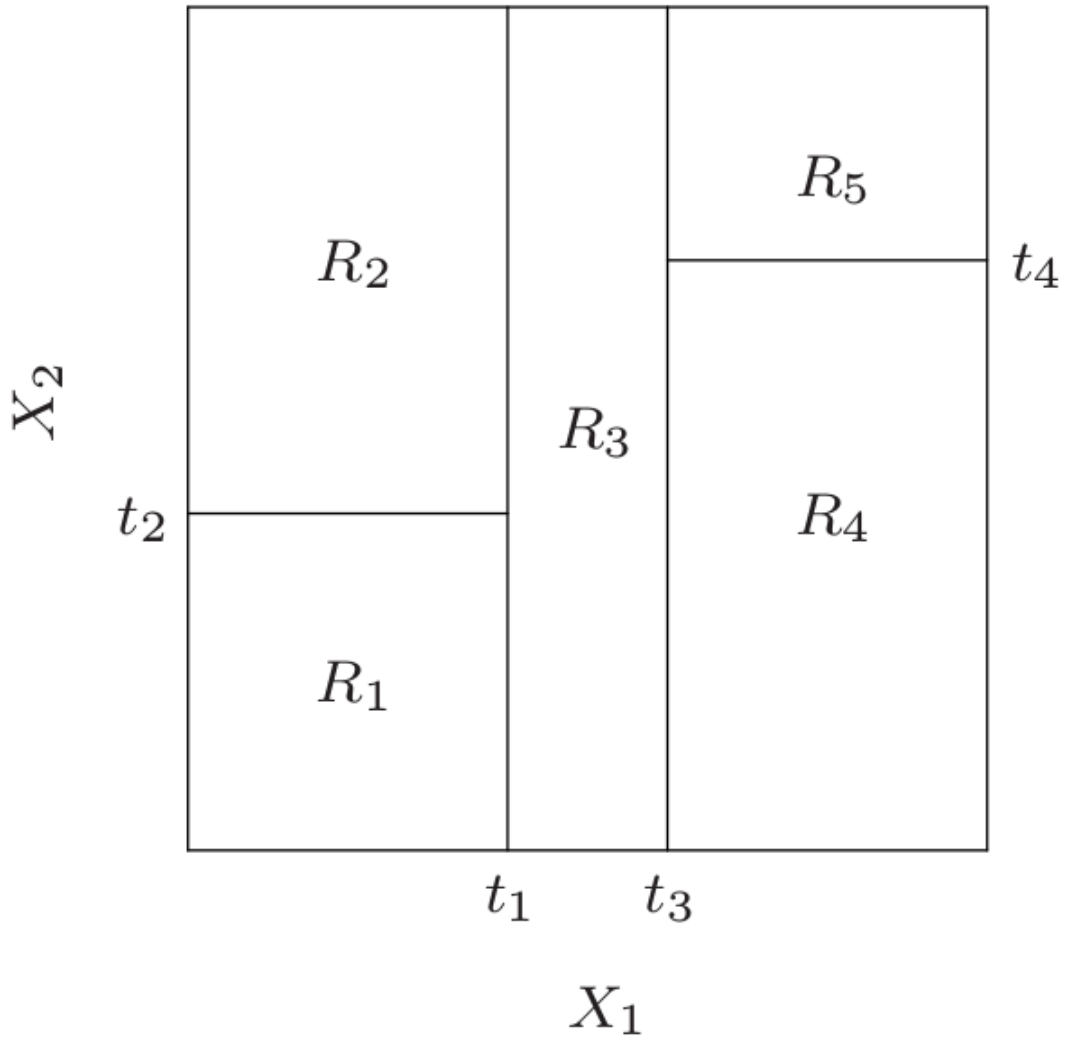
- Use an algorithm that **naturally handles multiple classes.**
  - Decision trees and their variants
  - Neural networks

Decision trees: The 20Q game



Partition of the feature space

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$



- **Classification:**

  c_m = majority vote in region.

- **Regression:**

  c_m = average value in region.

Partition of the feature space

- A decision tree is a **recursive partition** of the training set into smaller and smaller subsets.
- **Purity:** a node is **pure** if all samples at that node have the same class label.
- **CART: Classification And Regression Trees**

  Recursive procedure to split a training set and organize it into a tree.

CART: Design choices

- **Binary** or **multi-way** splits?
- Which **feature(s)** to use at each node?

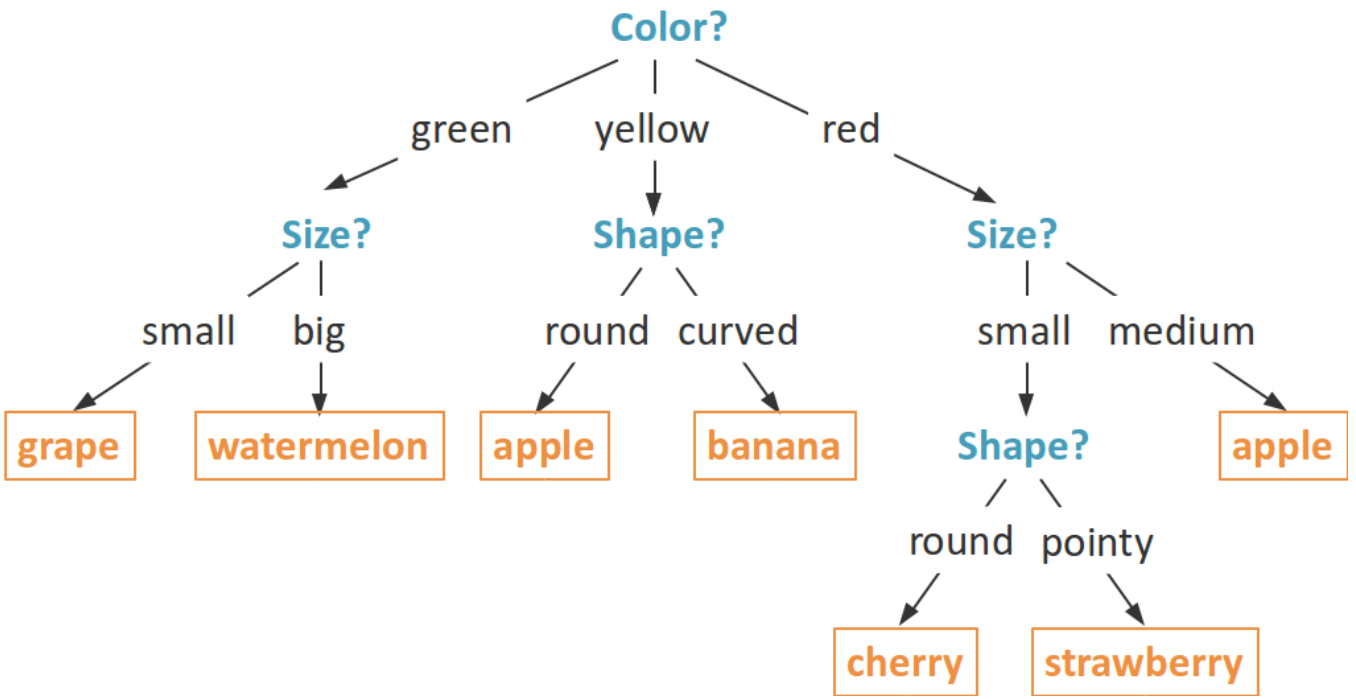
  i.e. how to split?
- When to **stop** growing a tree?

Binary vs. non-binary trees

Binary & multi-value splits

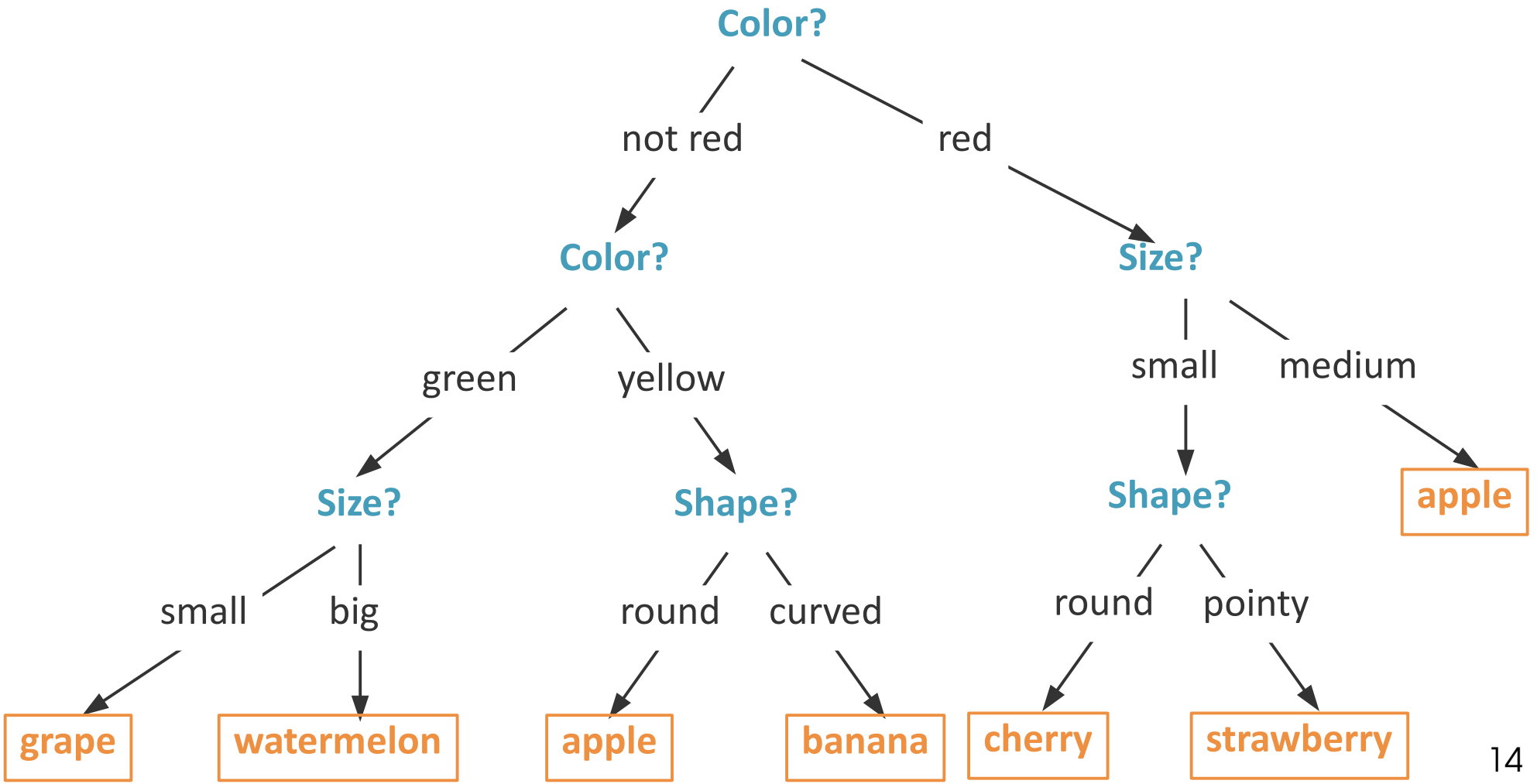
A tree with arbitrary branching factor can always be **equivalently represented** by a binary tree.

Find a binary tree that's equivalent to:



Binary & multi-value splits

A tree with arbitrary branching factor can always be **equivalently represented** by a binary tree.



How to grow a tree

How to grow a tree

- **Monothetic trees**
  - Use 1 feature per node
  - The decision boundary is

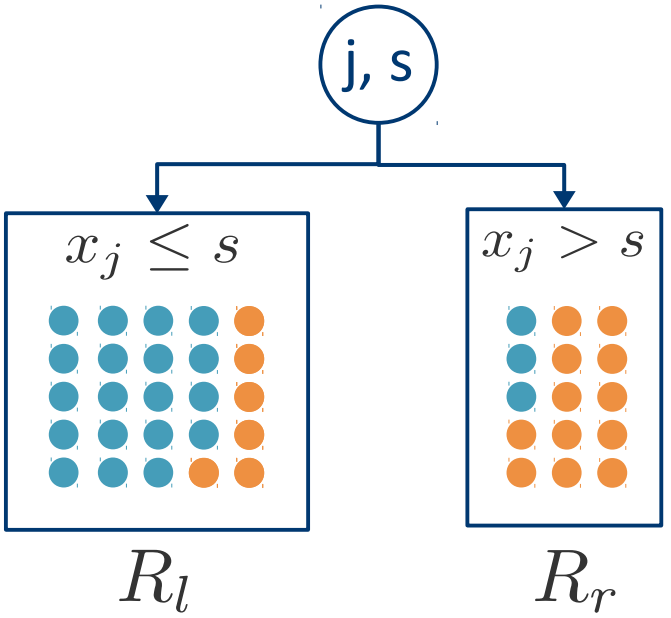
How to grow a tree

- **Monothetic trees**
  - Use 1 feature per node
  - The decision boundary is orthogonal to the axes.

How to grow a tree

- **Splitting variable** (j) and **splitting point** (s) define 2 regions:

$$R_l = \{x : x_j \leq s\} \text{ and } R_r = \{x : x_j > s\}$$



How to grow a tree

- **Splitting variable** (j) and **splitting point** (s) define 2 regions:

$$R_l = \{x : x_j \leq s\} \text{ and } R_r = \{x : x_j > s\}$$

- **Regression tree:** choose j and s to **minimize SE:**

$$\min_{j,s} \left(\sum_{i:x_i \in R_l(j,s)} (y_i - c_l)^2 + \sum_{i:x_i \in R_r(j,s)} (y_i - c_r)^2 \right)$$

How to grow a tree

- **Splitting variable** (j) and **splitting point** (s) define 2 regions:

$$R_l = \{x : x_j \leq s\} \text{ and } R_r = \{x : x_j > s\}$$

- **Classification tree:** choose j and s to **minimize impurity.**

$$\min_{j,s} \left(\frac{|R_l(j,s)|}{n} \times \text{Imp}(R_l(j,s)) + \frac{|R_r(j,s)|}{n} \times \text{Imp}(R_r(j,s)) \right)$$

- Greedy algorithm / local optimization.

How to grow a tree

- **Splitting variable** (j) and **splitting point** (s) define 2 regions:

$$R_l = \{x : x_j \leq s\} \text{ and } R_r = \{x : x_j > s\}$$

- **Classification tree:** choose j and s to **maximize the drop in impurity.**
- **Impurity:**
  - Classification error
  - Entropy
  - Gini impurity.

OPTIONAL

Impurity: Classification error

- Minimum probability that a training point will be misclassified at node (s,j)

$$\mathrm{Imp}(R_m) = 1 - \max_k \hat{p}_{mk}$$

proportion of training instances from class k in Rm

- If all examples from one class belong to Rm, then Imp(Rm) = 0
- If we have 2 balanced classes, and instances are randomly split at (s, j), then Imp(Rm) = 0.5

OPTIONAL

Impurity: Entropy

- Information theory: Shannon's entropy

proportion of training instances from class k in Rm

$$\mathrm{Imp}(R_m) = -\sum_k \hat{p}_{mk} \log_2 \hat{p}_{mk}$$

- If all examples from one class belong to Rm, then Imp(Rm) = 0
- If we have 2 balanced classes, and instances are randomly split at (s, j), then Imp(Rm) = 1

OPTIONAL

Gini impurity

$$\mathrm{Imp}(R_m) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

proportion of training instances from class k in Rm

- If all examples from one class belong to Rm, then Imp(Rm) = 0
- If we have 2 balanced classes, and instances are randomly split at (s, j), then Imp(Rm) = 0.5

OPTIONAL

Gini impurity

$$\mathrm{Imp}(R_m) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

$$GI(j,s) = \frac{|R_l(j,s)|}{n} \times \mathrm{Imp}(R_l(j,s)) + \frac{|R_r(j,s)|}{n} \times \mathrm{Imp}(R_r(j,s))$$

- If the **split respects the overall distribution:** $\hat{p}_{mk} = \frac{n_k}{n} \quad \forall k$
- All regions are identically distributed and have Gini impurity:

$$\mathrm{Imp}(R_m) = \sum_{k=1}^{K} \frac{n_k}{n}\left(1 - \frac{n_k}{n}\right) = 1 - \sum_{k=1}^{K} \frac{n_k^2}{n^2}$$

- If the dataset is **balanced** $n_k = \frac{n}{K} \quad \forall k$

  then

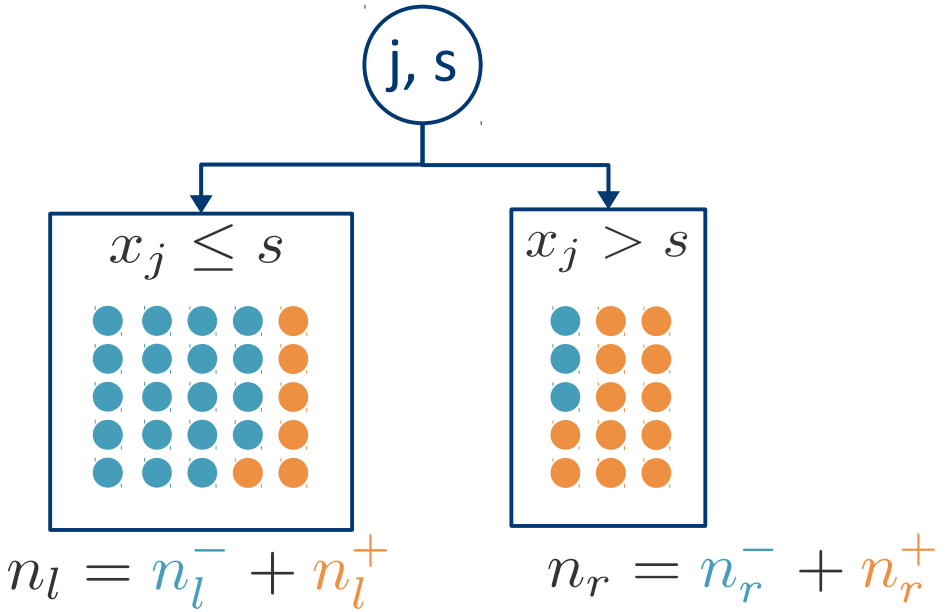
$$\mathrm{Imp}(R_m) = 1 - \frac{1}{K}$$

OPTIONAL

Gini impurity (K=2)

$$\text{Imp}(R_m) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

$$GI(j,s) = \frac{|R_l(j,s)|}{n} \times \text{Imp}(R_l(j,s)) + \frac{|R_r(j,s)|}{n} \times \text{Imp}(R_r(j,s))$$



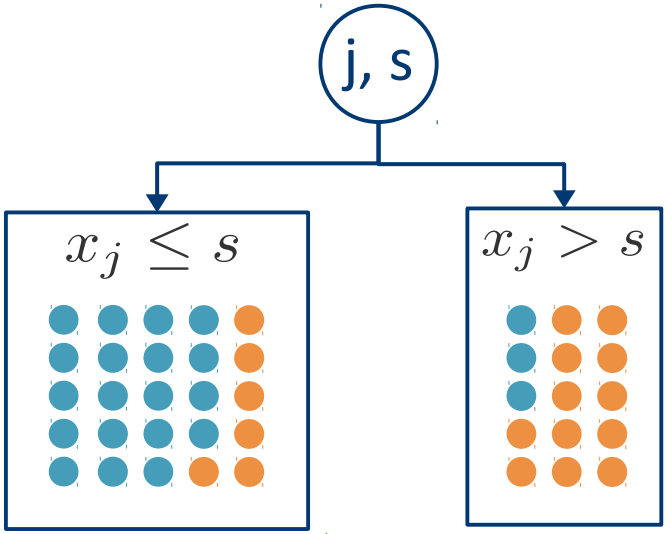
- Rewrite GI(j, s) using $n_l, n_l^+, n_l^-, n_r, n_r^+, n_r^-$

OPTIONAL

Gini impurity (K=2)

$$\text{Imp}(R_m) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

$$GI(j,s) = \frac{|R_l(j,s)|}{n} \times \text{Imp}(R_l(j,s)) + \frac{|R_r(j,s)|}{n} \times \text{Imp}(R_r(j,s))$$



$$n_l + n_r = n$$

$$n_l^- + n_r^- = n^-$$

$$n_l^+ + n_r^+ = n^+$$

$$n_l = n_l^- + n_l^+ \qquad n_r = n_r^- + n_r^+$$

$$GI(j,s) = \frac{n_l}{n}\left(\frac{n_l^-}{n_l}\left(1 - \frac{n_l^-}{n_l}\right) + \frac{n_l^+}{n_l}\left(1 - \frac{n_l^+}{n_l}\right)\right) + \frac{n_r}{n}\left(\frac{n_r^-}{n_r}\left(1 - \frac{n_r^-}{n_r}\right) + \frac{n_r^+}{n_r}\left(1 - \frac{n_r^+}{n_r}\right)\right)$$

$$GI(j,s) = \frac{2}{n}\left(\frac{n_l^- n_l^+}{n_l} + \frac{n_r^- n_r^+}{n_r}\right)$$

When to stop growing a tree

When to stop growing a tree

- **Large tree** might overfit
- **Small tree** might underfit
- Strategy:
  - grow the tree until a **minimum node size** (# training points in the region) is reached;
  - prune the tree: **cost-complexity pruning**.

OPTIONAL

When to stop growing a tree

– prune the tree: **cost-complexity pruning**.

number of regions in T

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T|$$

pruned tree

number of training instances in Rm

Error on Rm

α: trade-off between model complexity and goodness of fit.

Advantages & drawbacks of trees

- :-) Trees are **easy to explain.**
- :-) Trees seem to **mirror human-decision making.**
- :-) Trees can be **displayed graphically** and **easily interpreted.**
- :-) Trees can easily handle **quantitative variables.**
- :-) Trees naturally handle **multi-class problems.**

- :-(Trees generally do not have very good **predictive accuracy.**

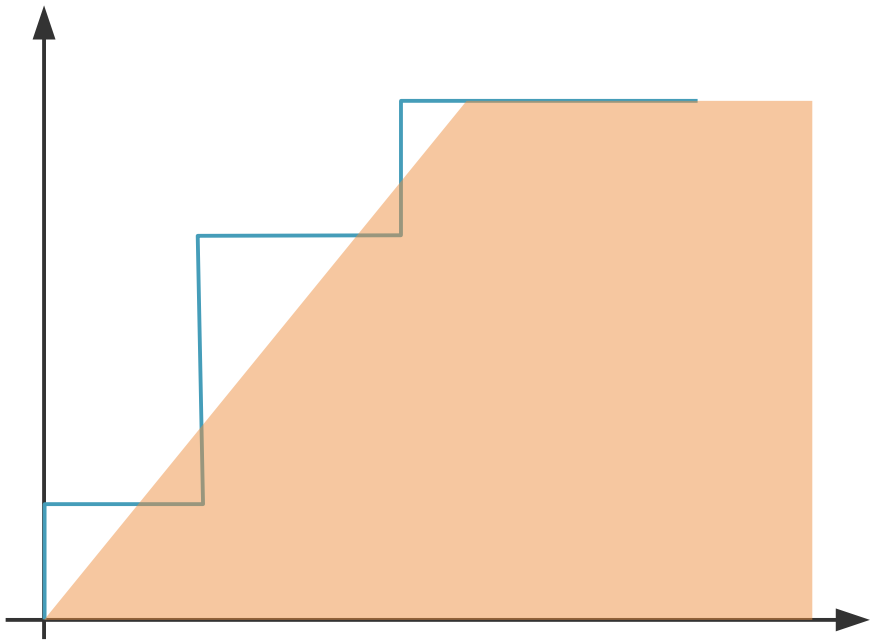
Forests

Building forests

- Idea: **Aggregating many weak learners can substantially increase their performance.**
- **Ensemble learning**

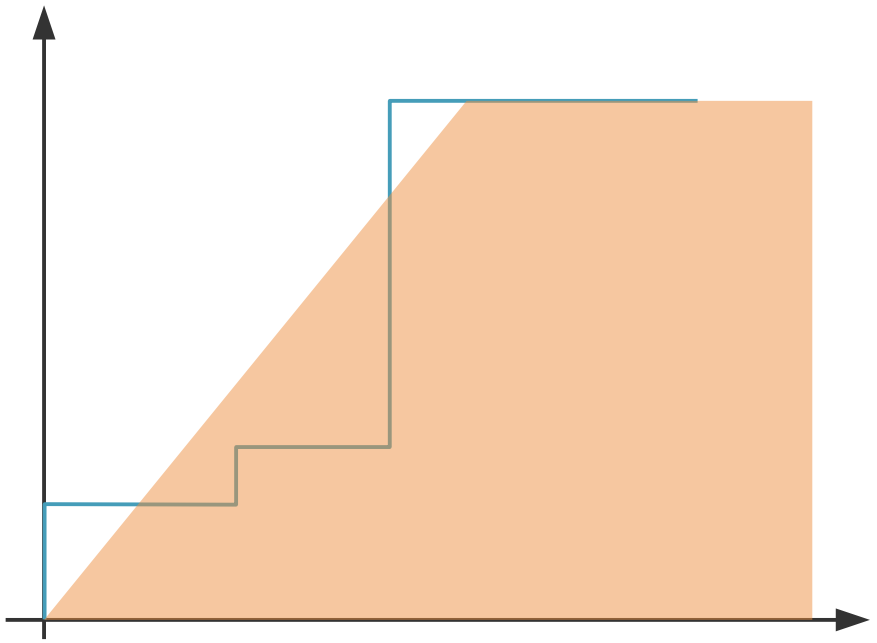
Ensemble learning

- **Wisdom of crowds:** Average out the uncorrelated errors of individual classifiers.
- **Example:** Learn a diagonal separation from "staircase" decision boundaries.



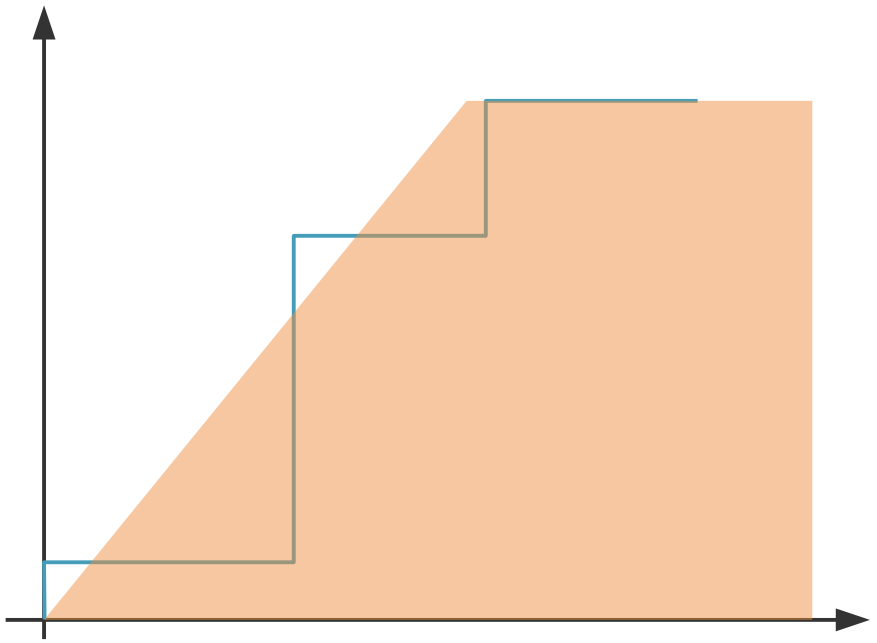
Ensemble learning

- **Wisdom of crowds:** Average out the uncorrelated errors of individual classifiers.
- **Example:** Learn a diagonal separation from "staircase" decision boundaries.



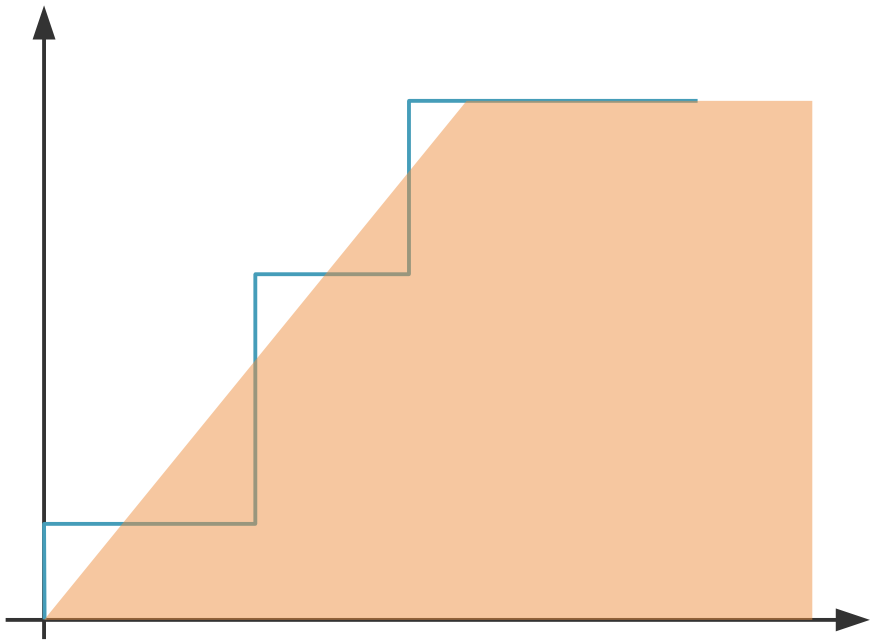
Ensemble learning

- **Wisdom of crowds:** Average out the uncorrelated errors of individual classifiers.
- **Example:** Learn a diagonal separation from "staircase" decision boundaries.



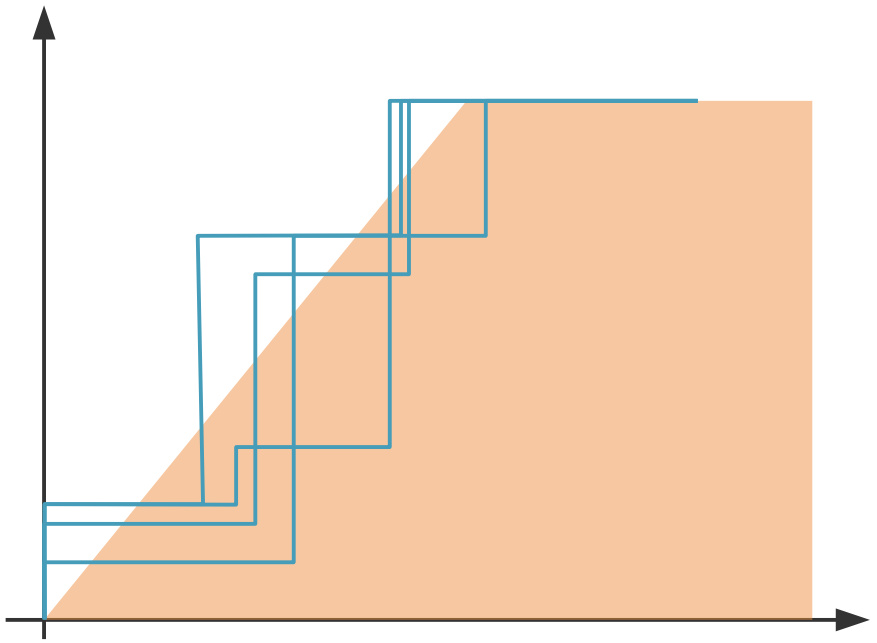
Ensemble learning

- **Wisdom of crowds:** Average out the uncorrelated errors of individual classifiers.
- **Example:** Learn a diagonal separation from "staircase" decision boundaries.



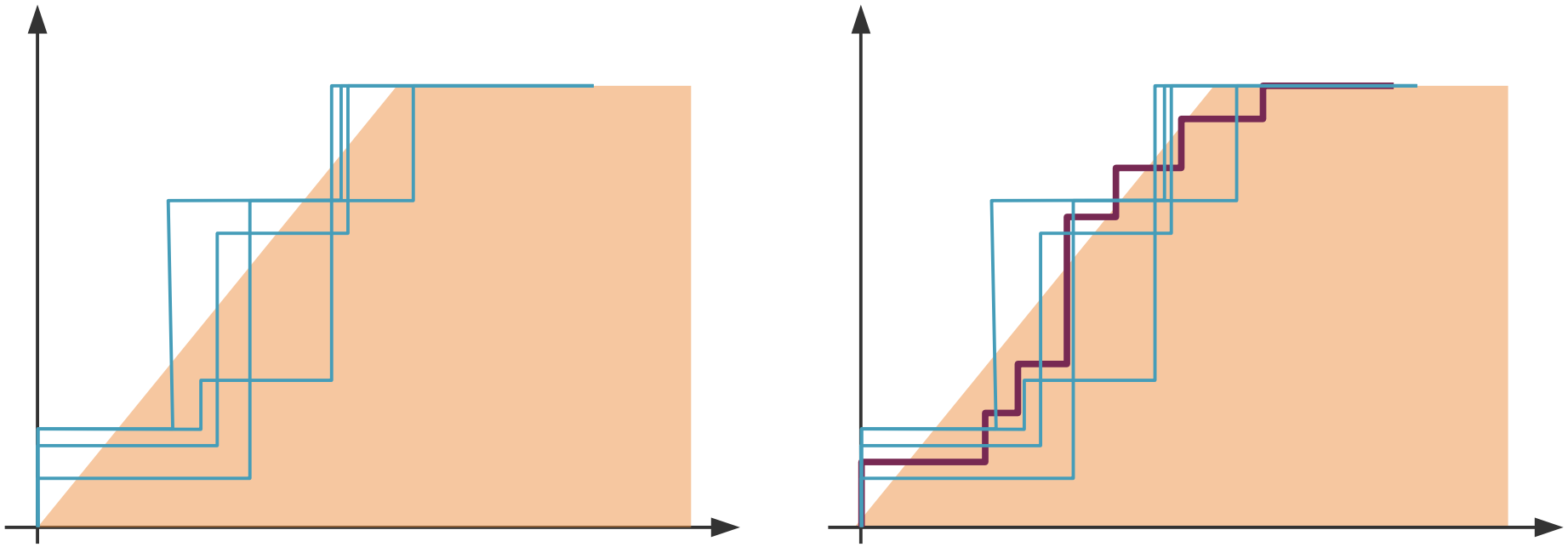
Ensemble learning

- **Wisdom of crowds:** Average out the uncorrelated errors of individual classifiers.
- **Example:** Learn a diagonal separation from "staircase" decision boundaries.



Ensemble learning

- **Wisdom of crowds:** Average out the uncorrelated errors of individual classifiers.
- **Example:** Learn a diagonal separation from "staircase" decision boundaries.



Building ensembles

- **Subsample** the training data
  - **Bagging** [Breiman 1996]: bootstrap resampling
  - **Boosting** [Schapire 1990]: resample based on performance

Combining learners

- **Non-trainable combination:**
  - **Voting** (classification)
  - **Averaging** (regression)
- **Trainable combination:**
  - **Weighted averaging:** based on performance on a validation set.
  - **Meta-learner:** the outputs of the individual learners are features for another learning algorithm.

Bagging trees

- **Bagging:**
  - Take repeated samples from the training data (bootstrap)
  - Build one predictor from each of these samples
  - **Final prediction:** average (regression) or majority vote (classification)

Random forests [Breiman 2001]

- Similar to bagging trees
- One trick to **decorrelate** the trees:

  Before splitting, first **randomly sample q** (out of p) **variables** among which the one over which to split must be chosen.

  Typically $q = \sqrt{p}$.
- Very good predictive power in practice!

$y \in \{-1, +1\}$

AdaBoost

[Schapire & Freund 1997]

- weak learners = **stumps** (only one split)
- Give more weight to the more difficult samples
- At iteration m:
  - learn f_m from data weighted by $\{w_i^{m-1}\}_{i=1,\dots n}$

$$\epsilon_m = \sum_{i=1}^{n} w_i^m \delta_{f_m(\boldsymbol{x}^i) \neq y^i}$$

weighted error

$$\alpha_m = \frac{1}{2} \log \frac{1 - \epsilon_m}{\epsilon_m}$$

  - update weights: $w_i^m = \frac{1}{Z_m} w_i^{m-1} \exp\left(-\alpha_m y^i f_m(\boldsymbol{x}^i)\right)$

exponential loss

such that the weights sum to 1

- **Final decision function:**

$$f : \boldsymbol{x} \mapsto \sum_{m=1}^{M} \alpha_m f_m(\boldsymbol{x})$$

$y \in \{-1, +1\}$

Gradient Boosting [Friedman 2001]

- At iteration m:
  - learn f_m that minimizes a loss function for predictor

$$F_m = \sum_{l=1}^{m} \alpha_l f_l = F_{m-1} + \alpha_m f_m$$

    by gradient descent
  - **Exponential loss:** equivalent to **AdaBoost**

$$\mathcal{L}(y, f(\boldsymbol{x})) = \exp(-y f(\boldsymbol{x}))$$

  - Other possible losses:
    - **cross-entropy:**

$$\mathcal{L}(y, f(\boldsymbol{x})) = \log(1 + \exp^{-y f(\boldsymbol{x})}) = -\sum_{k=0}^{K-1} y \log(f_k(\boldsymbol{x}))$$

= logistic loss

$y \in \{0, 1\}$

multiclass

= binomial divergence loss

    - **least-squares:**

$$\mathcal{L}(y, f(\boldsymbol{x})) = \frac{1}{2}(y - f(\boldsymbol{x}))^2$$

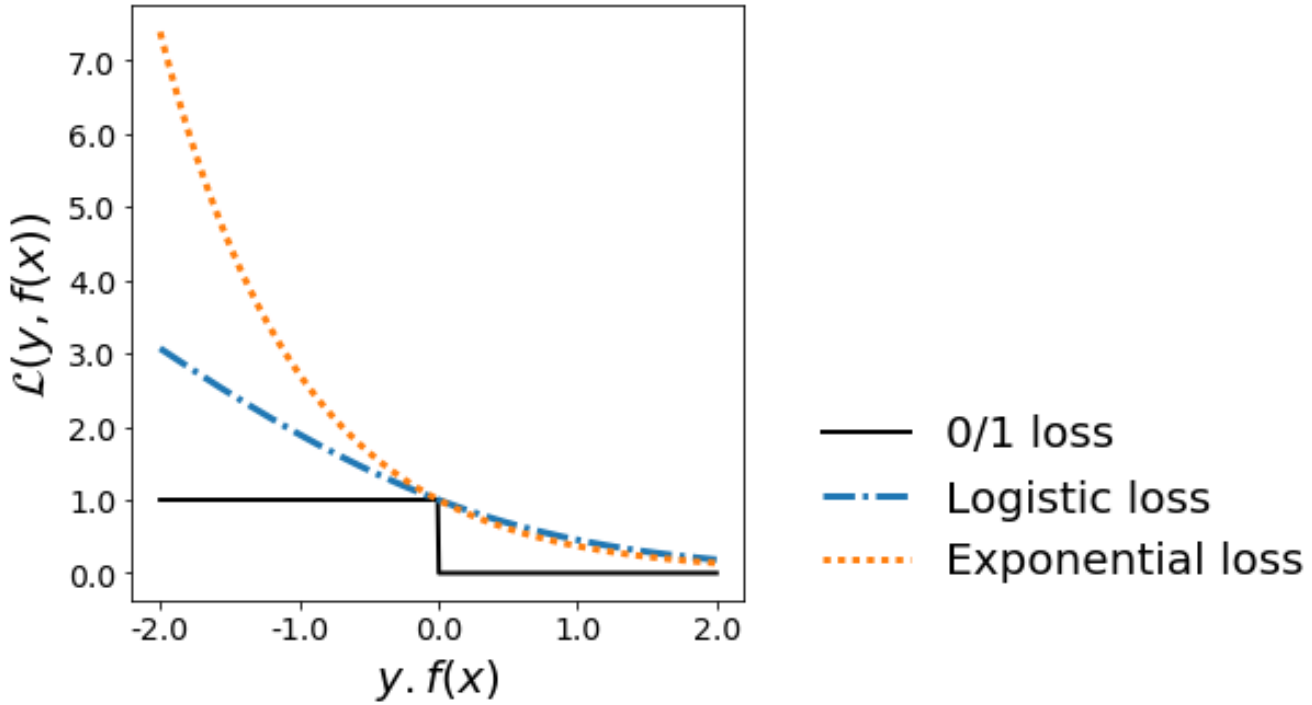
$y \in \{-1, +1\}$

Gradient Boosting [Friedman 2001]

- At iteration m:
  - learn f_m that minimizes a loss function for predictor

$$F_m = \sum_{l=1}^{m} \alpha_l f_l = F_{m-1} + \alpha_m f_m$$

    by gradient descent



Summary

- Decision trees are **easy to interpret.**
- Decision trees elegantly deal with
  - **Quantitative variables**
  - **Multiple classes**
  - **Multimodal distributions.**
- Decision trees have **limited predictive power,** but this can be addressed thanks to **ensemble methods**
  - **Boosting**
  - **Bagging**
  - **Random forests.**

References

- *A Course in Machine Learning.*
  http://ciml.info/dl/v0_99/ciml-v0_99-all.pdf
  - **Decision trees:** Chap 1.3
  - **Boosting :** Chap 13.2
  - **Random forests:** Chap 13.3
- *The Elements of Statistical Learning.*
  http://web.stanford.edu/~hastie/ElemStatLearn/
  - **Decision trees:** Chap 9.2
  - **Boosting:** Chap 10.1 – 10.10
  - **Random forests:** Chap 15.1 – 15.2
- **A complete tutorial on tree-based modeling**

  https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/#ten