MACHINE LEARNING & DATA MINING

Decision trees (recap)

Model based on

- directed graph
- with no loops
- with single root node
- each node has:
  - either 0 child nodes (terminal node)
  - or ≥ 2 child nodes (internal node)

Defining a tree

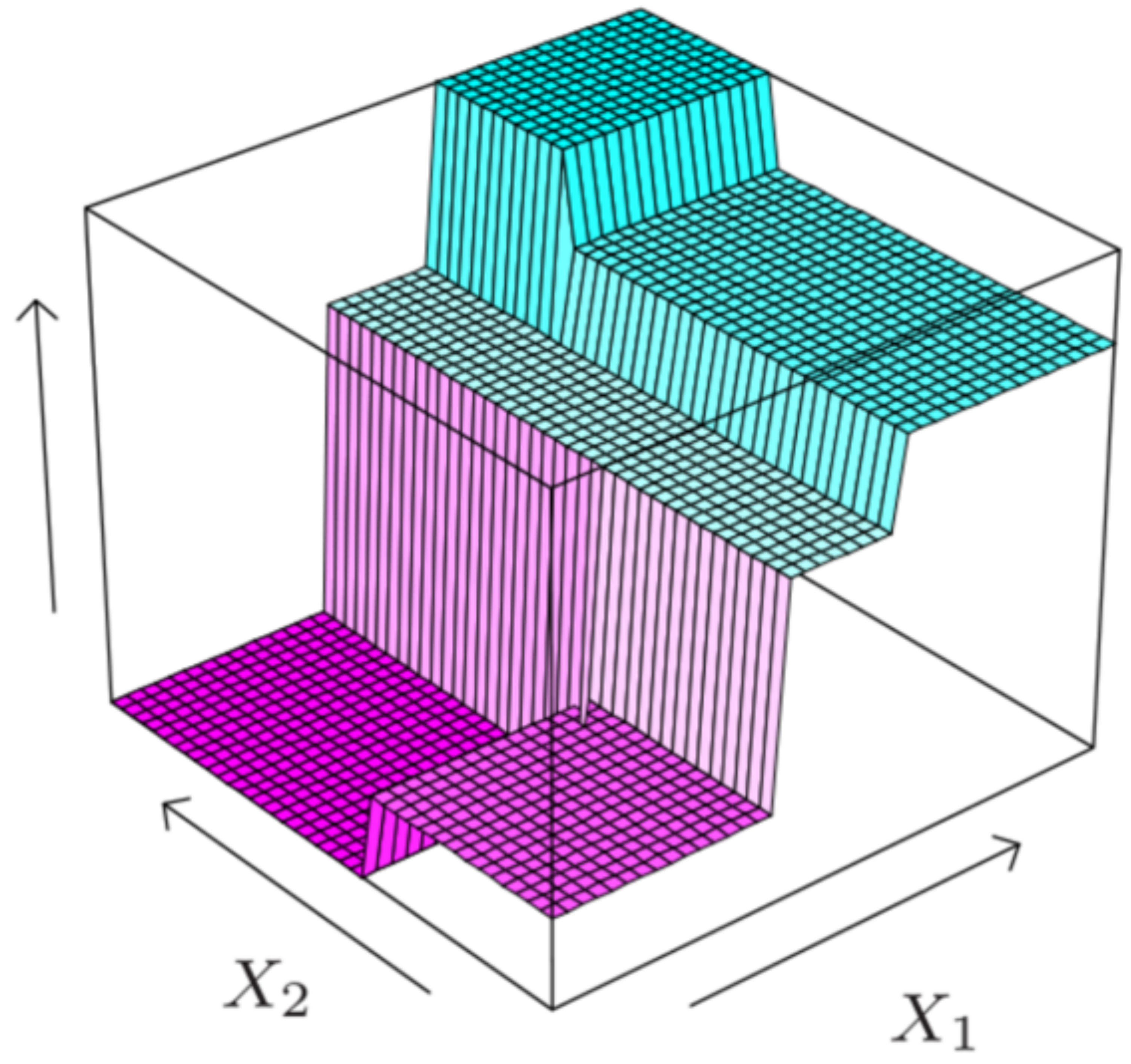
Defining a tree T:

- associate a check-function $Q_t(x)$ for each node t
- assign each child node of t a set of unique values of $Q_t(x)$
- assign each terminal node a prediction value

Defining a tree

Defining a tree T:

- associate a check-function $Q_t(x)$ for each node t
- assign each child node of t a set of unique values of $Q_t(x)$
- assign each terminal node a prediction value



Note:

- decision tree prediction is always a piecewise constant function

CART

«Classification and Regression Trees» (CART):

- $Q_t(x) = x^{i(t)}$ – single feature value
- Only two child nodes based on rule: $I(Q_t(x) > h_t)$

Advantages and limitations

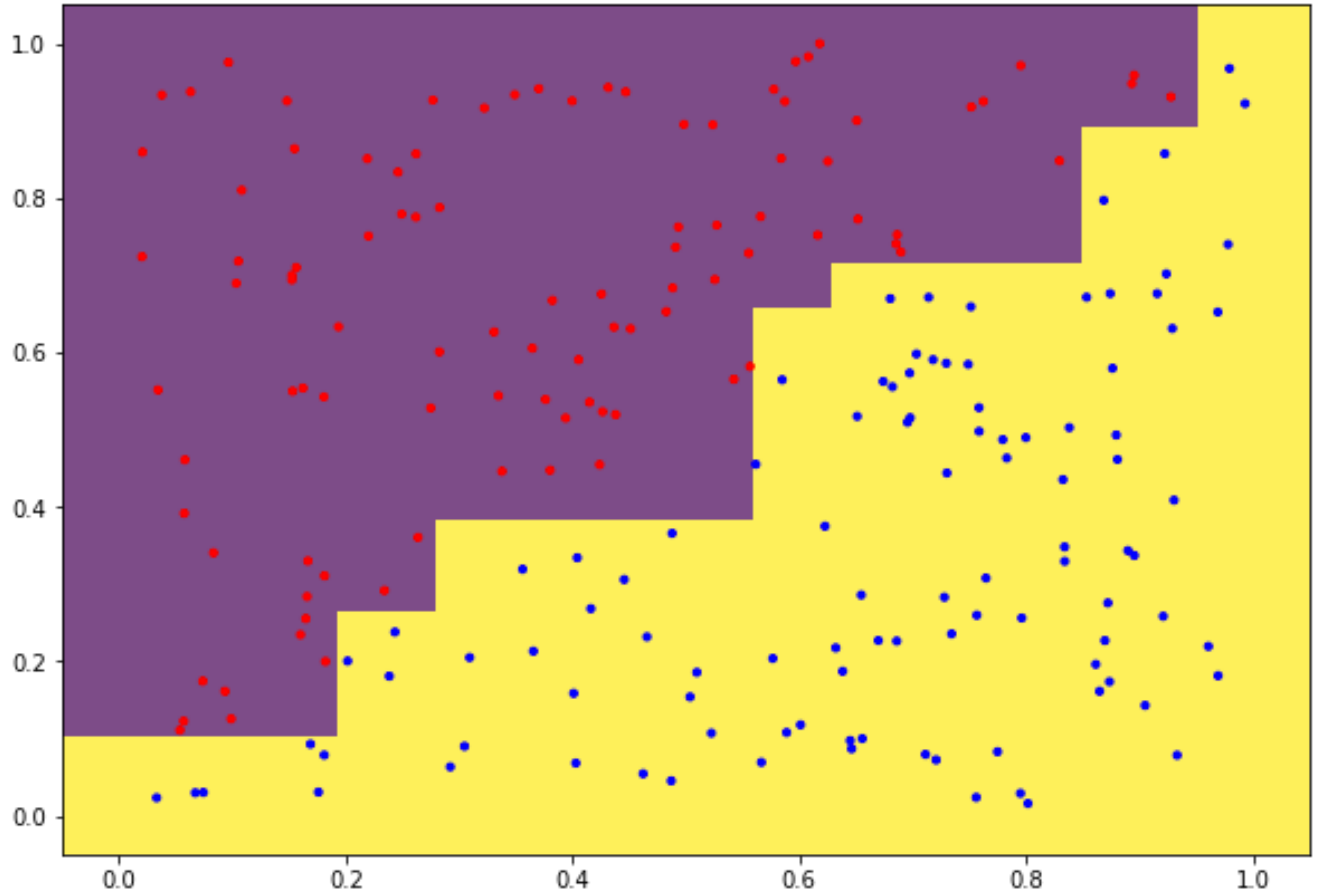
CART splitting rule advantages:

- simplicity
- estimation efficiency
- interpretability

Advantages and limitations

CART splitting rule advantages:

- simplicity
- estimation efficiency
- interpretability



Limitations:

- many nodes needed for boundaries not parallel to axes

Decision trees learning

Decision trees learning

Iteratively growing the tree:

Decision trees learning

Iteratively growing the tree:

- Start from the single root node (containing all the training set)

Decision trees learning

Iterative growing the tree:

- Start from the single root node (containing all the training set)
- For each unchecked terminal node t:

Decision trees learning

Iteratively growing the tree:

- Start from the single root node (containing all the training set)
- For each unchecked terminal node t:
  - Find the best split (feature $i(t)$ and threshold h_t)

Decision trees learning

Iteratively growing the tree:

- Start from the single root node (containing all the training set)
- For each unchecked terminal node t:
  - Find the best split (feature $i(t)$ and threshold h_t)
  - If stopping criterion not fulfilled:

Decision trees learning

Iteratively growing the tree:

- Start from the single root node (containing all the training set)
- For each unchecked terminal node t:
  - Find the best split (feature $i(t)$ and threshold h_t)
  - If stopping criterion not fulfilled:
    - attach two new nodes according to the found split

Decision trees learning

Iteratively growing the tree:

- Start from the single root node (containing all the training set)
- For each unchecked terminal node t:
  - Find the best split (feature $i(t)$ and threshold h_t)
  - If stopping criterion not fulfilled:
    - attach two new nodes according to the found split
- For each terminal node t assign the prediction value

Impurity functions

- Finding the best split based on the reduction of (weighted) impurity
- Impurity functions (regression):

Impurity functions

- Finding the best split based on the reduction of (weighted) impurity
- Impurity functions (regression):

$$I(t) = \frac{1}{|t|} \sum_{(x_i, y_i) \in t} (y_i - \mu)^2$$

Impurity functions

- Finding the best split based on the reduction of (weighted) impurity
- Impurity functions (regression):

$$I(t) = \frac{1}{|t|} \sum_{(x_i, y_i) \in t} (y_i - \mu)^2$$

$$I(t) = \frac{1}{|t|} \sum_{(x_i, y_i) \in t} |y_i - \mu|$$

Impurity functions (classification)

- Let class c probability for node t be:

$$p(c\,|\,t) = \frac{|\{i : (x_i, y_i) \in t, y_i = c\}|}{|t|}$$

Impurity functions (classification)

- Let class c probability for node t be:

$$p(c\,|\,t) = \frac{|\{i : (x_i, y_i) \in t, y_i = c\}|}{|t|}$$

- Possible impurity functions for classification:
  - *Gini criterion* (probability to make mistake when predicting randomly according to $p(c \mid t)$):

$$I(t) = \sum_c p(c\,|\,t) \cdot (1 - p(c\,|\,t)) = 1 - \sum_c [p(c\,|\,t)]^2$$

Impurity functions (classification)

- Let class c probability for node t be:

$$p(c\,|\,t) = \frac{|\{i : (x_i, y_i) \in t, y_i = c\}|}{|t|}$$

- Possible impurity functions for classification:
  - *Entropy* (measure of uncertainty of a random variable):

$$I(t) = -\sum_c p(c\,|\,t) \log p(c\,|\,t)$$

Impurity functions (classification)

- Let class c probability for node t be:

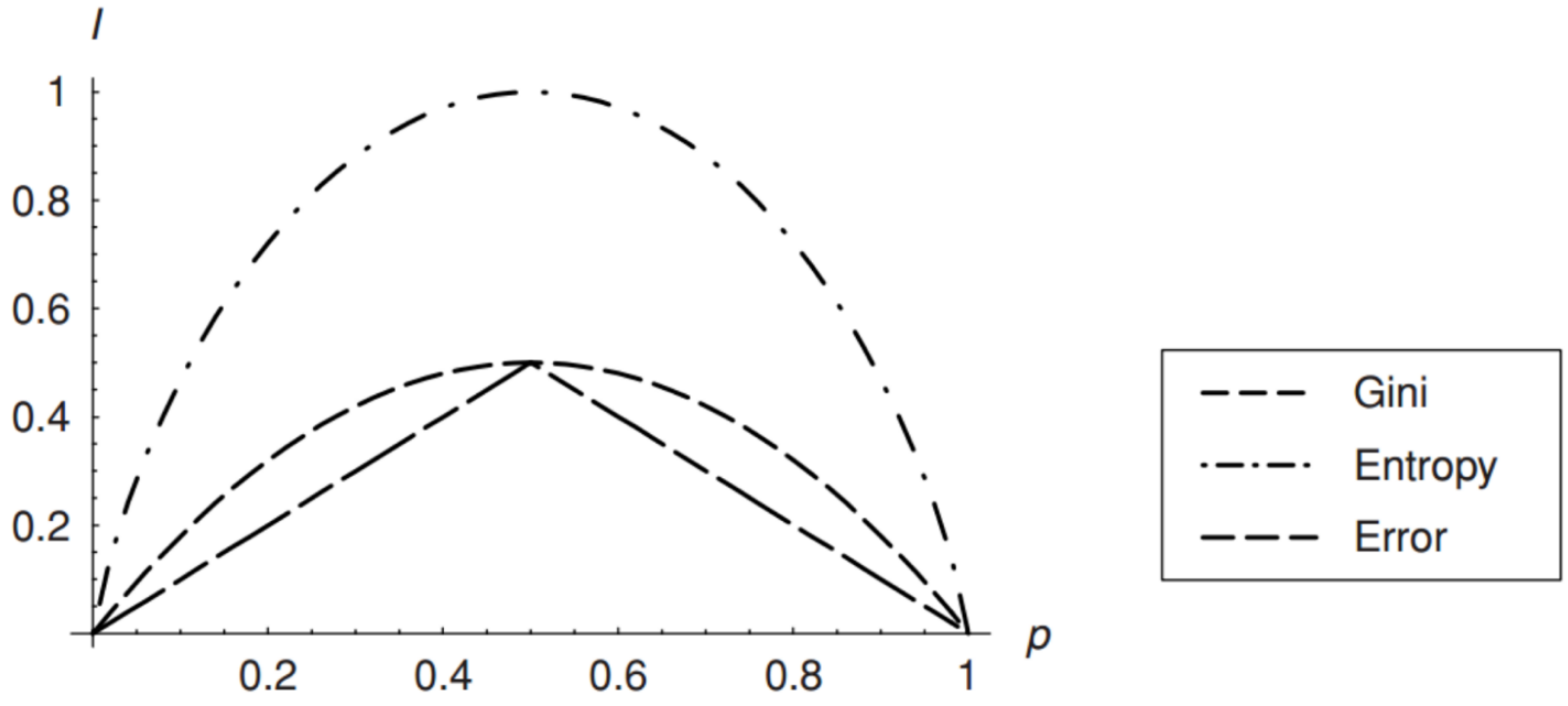
$$p(c\,|\,t) = \frac{|\{i : (x_i, y_i) \in t, y_i = c\}|}{|t|}$$

- Possible impurity functions for classification:
  - *Classification error* (frequency of errors when predicting most probable class):

$$I(t) = 1 - \max_c p(c\,|\,t)$$

Impurity functions (classification)

- For binary classification:



Termination criterion

- Controls the bias-variance tradeoff
- E.g.:
  - depth of the tree
  - number of objects in a node
  - minimal number of objects in each child
  - impurity
  - change of impurity

Analysis of decision trees

Advantages:

- simplicity
- interpretability
- implicit feature selection
  - (e.g. importance based on weighted impurity reduction from particular feature from all nodes)
- good for features of different nature

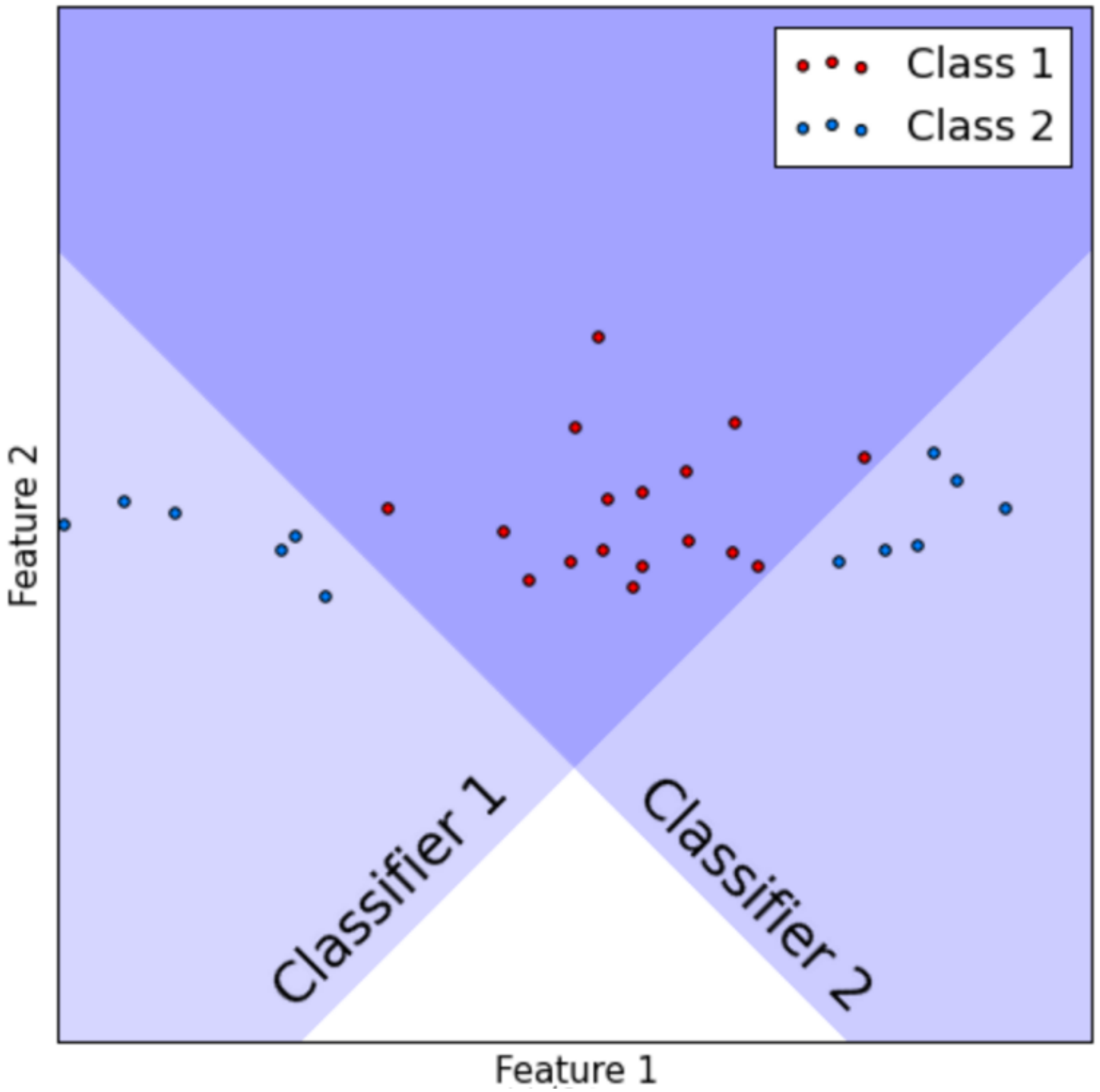
Disadvantages:

- Overfit easily
- Not optimal for boundaries non-parallel to axes
- one step ahead lookup for the best split may be insufficient (e.g. for XOR task)

Ensembling

- Models, using predictions of other models
- Example: stacking

Ensembles for solving underfitting



Ensembles for solving overfitting

- Overfitting <=> high variance
- For regression: average the predictions to reduce variance
- For classification: majority voting

Ambiguity decomposition

- Let ensemble model be defined as a weighted sum of individual models:

$$F(x) = \sum_m w_m f_m(x), \quad w_m \geq 0, \quad \sum_m w_m = 1$$

- Then one can prove:

$$(F(x) - y)^2 = \sum_m w_m \left(f_m(x) - y\right)^2 - \sum_m w_m \left(f_m(x) - F(x)\right)^2$$

Ambiguity decomposition

- Let ensemble model be defined as a weighted sum of individual models:

$$F(x) = \sum_m w_m f_m(x), \quad w_m \geq 0, \quad \sum_m w_m = 1$$

- Then one can prove:

$$\underbrace{(F(x) - y)^2}_{\text{ensemble error}} = \underbrace{\sum_m w_m \left(f_m(x) - y\right)^2}_{\text{base learner error}} - \underbrace{\sum_m w_m \left(f_m(x) - F(x)\right)^2}_{\text{ambiguity}}$$

Bagging

- Bootstrap sample — data sampled from the training set with replacement (typically of the same size as the original set)

Bagging

- Bootstrap sample — data sampled from the training set with replacement (typically of the same size as the original set)

- Bagging = Bootstrap Aggregating = Combining predictions of models trained on bootstrap samples

Bagging

- Bootstrap sample — data sampled from the training set with replacement (typically of the same size as the original set)

- Bagging = Bootstrap Aggregating = Combining predictions of models trained on bootstrap samples

- Random forest = bagged trees + random sampling of features

Bagging

- Bootstrap sample — data sampled from the training set with replacement (typically of the same size as the original set)

- Bagging = Bootstrap Aggregating = Combining predictions of models trained on bootstrap samples

- Random forest = bagged trees + random sampling of features

- Extra random trees = bagged trees + random sampling of features + possible splits randomly sampled for each feature

Forward stagewise additive modeling (FSAM)

- Loss function $L(f, y)$
- Base learners f_m
- Approximate the output as:

$$F_M(x) = f_0(x) + \sum_{m=1}^{M} c_m f_m(x)$$

Forward stagewise additive modeling (FSAM)

- Loss function $L(f, y)$
- Base learners f_m
- Approximate the output as:

$$F_M(x) = f_0(x) + \sum_{m=1}^{M} c_m f_m(x)$$

- Do so in steps:
  - Start from 0, constant or just fit f_0 to data
  - At each step solve:

$$(c_m, f_m) = \arg\min_{c,\, f} \left[\sum_{n=1}^{N} L(F_{m-1}(x_n) + c\, f(x_n), y_n) \right]$$

AdaBoost (classification)

- AdaBoost = FSAM with exponential loss:

$$L = \sum_n \exp\left[-y_n f(x_n)\right], \quad y \in \{-1, 1\}$$

- $f(x_n) \in \{-1, 1\}$
- Minimization can be solved analytically
  - provided individual learners allow for weighted samples

AdaBoost (classification)

$$L(F_m) = \sum_n \exp\left[-y_n \left(F_{m-1}(x_n) + c\, f(x_n)\right)\right] =$$

$$\sum_n \exp\left[-y_n\, F_{m-1}(x_n)\right] \cdot \exp\left[-y_n c\, f(x_n)\right] =$$

$$\sum_n w_n \cdot \exp\left[-y_n c\, f(x_n)\right]$$

- So at each step we train a base learner on the weighted samples, while the coefficient can be determined:

$$c = \frac{1}{2} \log \frac{1 - \text{weighted error rate}}{\text{weighted error rate}}$$

- Early stop when weighted error rate is 0 or > 0.5

Gradient boosting

- FSAM minimization cannot be solved analytically for a general loss function
- Find approximation (linear):

$$L(F(x) + f(x), y) \approx L(F(x), y) + \frac{\partial L(F, y)}{\partial F} f(x)$$

- Gradient shows the direction of maximal increase
- => Fit f(x) to the negative of the gradient
- Then solve for c:

$$L(F(x) + c\, f(x), y) \to \min_{c}$$

Quadratic approximation

$$L(F(x)+f(x),y) \approx L(F(x),y) + \frac{\partial L(F,y)}{\partial F} f(x) + \frac{1}{2}\frac{\partial^2 L(F,y)}{\partial^2 F}(f(x))^2$$

$$= \underbrace{\frac{1}{2}\frac{\partial^2 L(F,y)}{\partial^2 F}}_{\text{weights}} \left(f(x) + \underbrace{\frac{\frac{\partial L(F,y)}{\partial F}}{\frac{\partial^2 L(F,y)}{\partial^2 F}}}_{\text{fitting targets}} \right)^2 + \mathrm{const}(f(x))$$

Discussion

- Ensembling can turn individual algorithm’s weakness into strength
- Averaging a set of base learners with high variance will reduce the variance
  - given individual learners are diverse (ambiguity is large)
- Boosting can overfit easily, a number of regularization techniques can be applied:
  - shrinkage (adding new learners to the sum times a small constant)
  - penalizing the number of leafs (in case of boosted trees) and the magnitude of prediction
  - subsampling / feature subsampling
- Very complex base learners not preferable for boosting
- Number of base learners controls the bias-variance tradeoff