Machine & Deep Learning

Pr Rahhal ERRATTAHI
rahhal.errattahi@um6p.ma
errattahi.r@ucd.ac.ma

Lecture 05

Decision Trees & Ensemble Learning

Lecture 5 Overview

- Decision trees
- Ensemble learning
- Bagging
- Boosting
- Stacking
- Summary

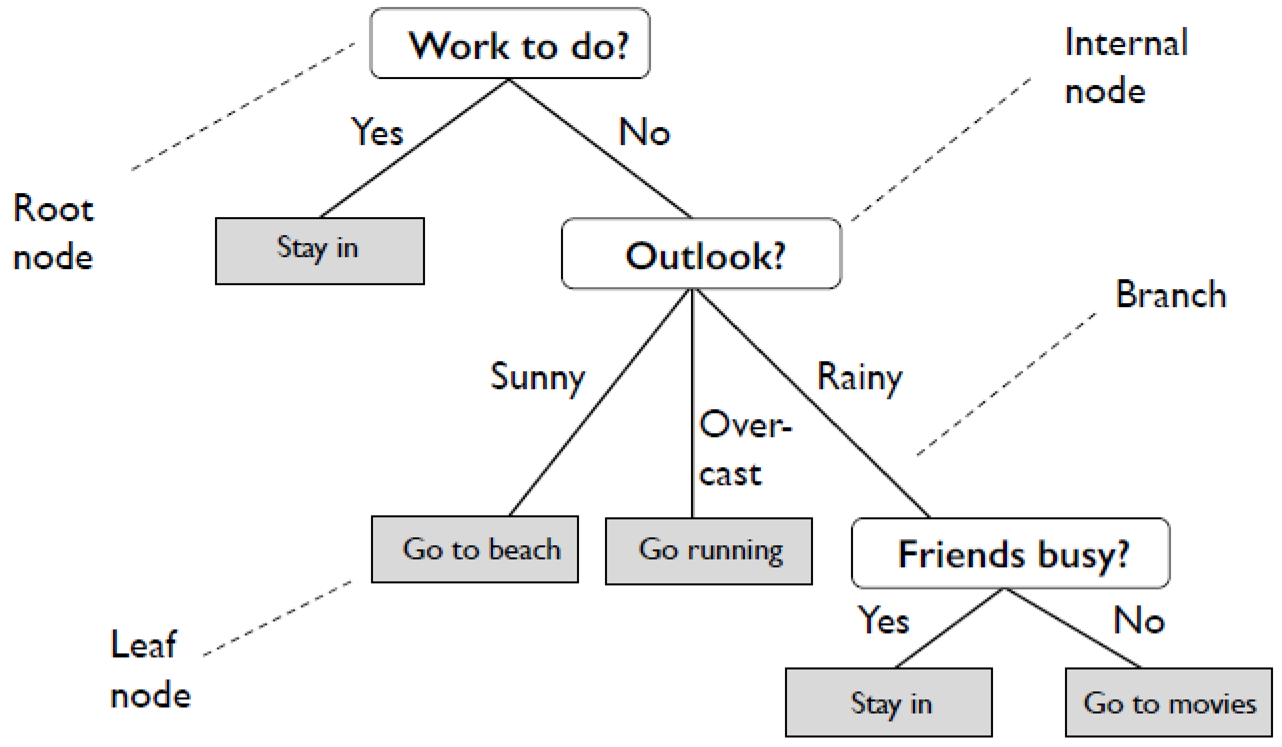
Lecture 5 Overview

- Decision trees
- Ensemble learning
- Bagging
- Boosting
- Stacking
- Summary

Decision trees

- Representation: Tree that splits data points into leaves based on tests
- Evaluation (loss): Heuristic for purity of leaves (Gini index, entropy,…)
- Optimization: Recursive, heuristic greedy search (Hunt’s algorithm)
    - Consider all splits (thresholds) between adjacent data points, for every feature
    - Choose the one that yields the purest leafs, repeat

Decision trees

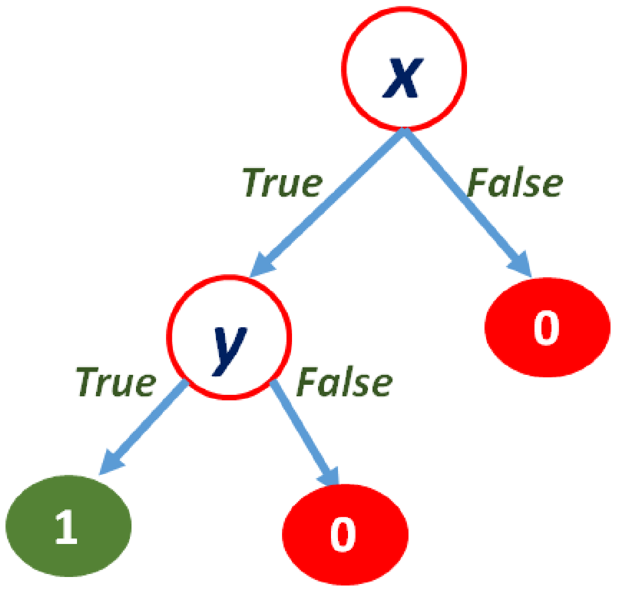


Decision trees

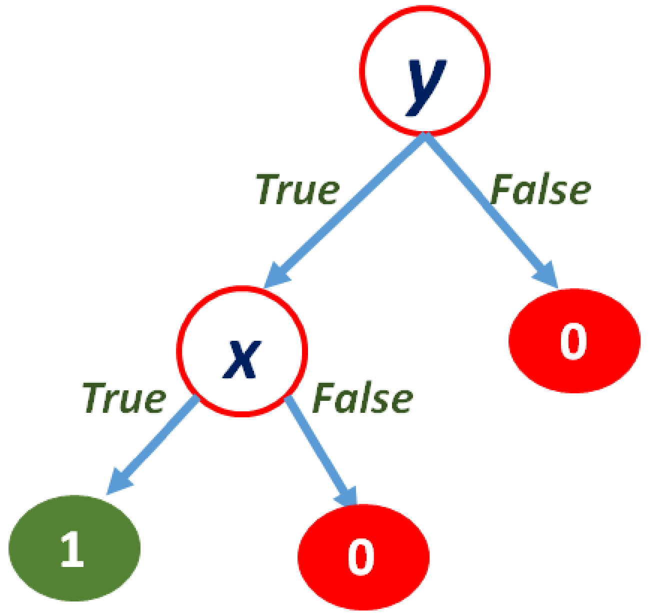
AND $(x \wedge y)$

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

What will be its decision tree?



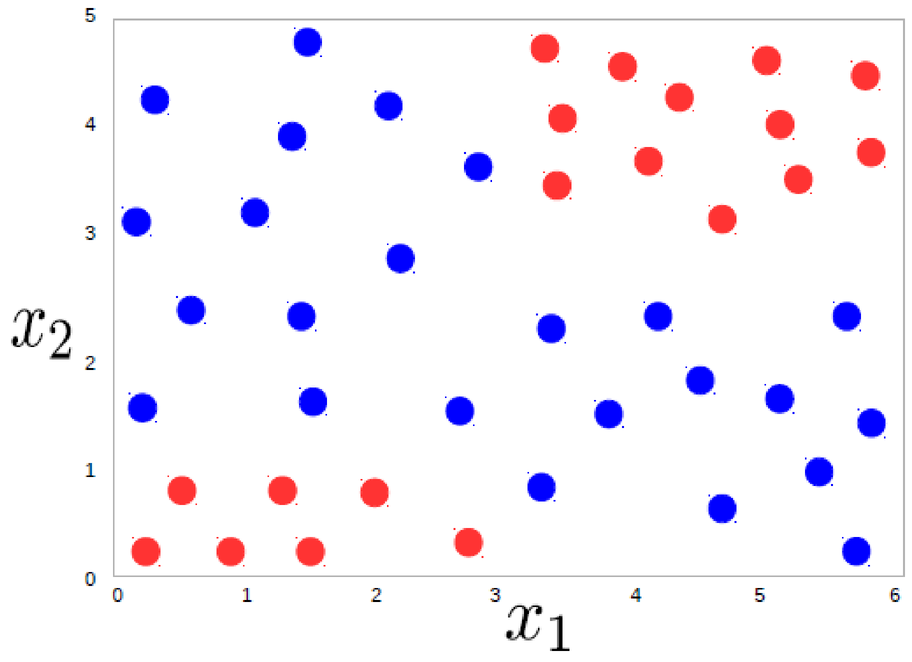
Or!



By choosing different attribute at the top of the tree, algorithm may find better tree representation (not true in this case), but generally it matters.

Decision trees

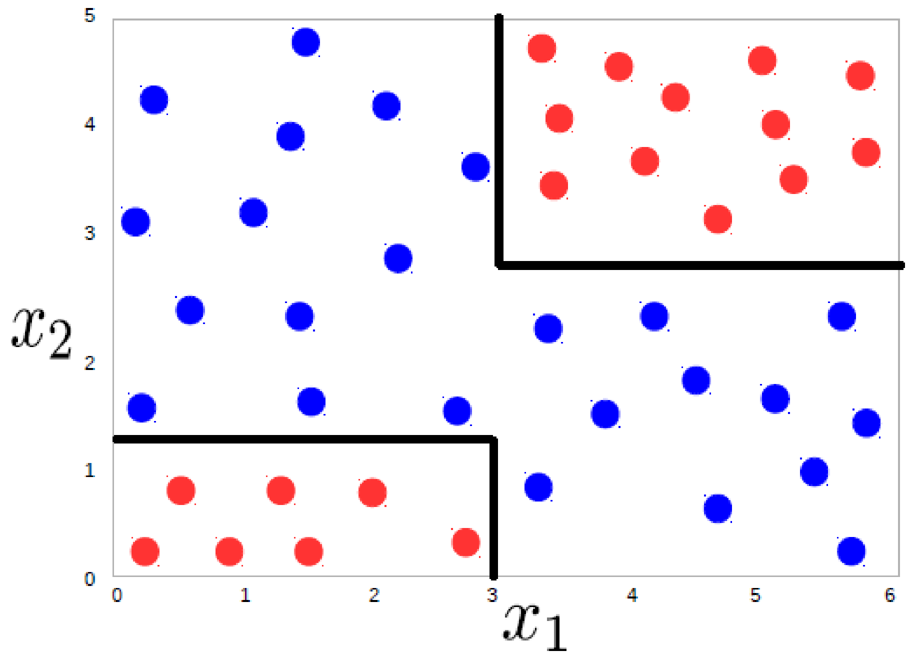
- Consider below presented classification (binary) problem.



- What is the “expected” decision boundary given this training data ?

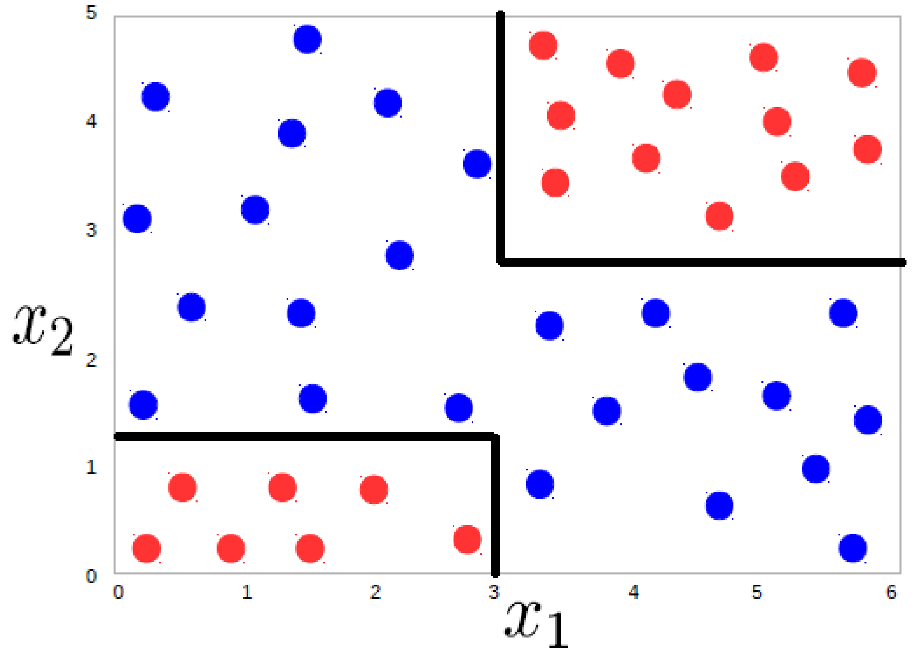
Decision trees

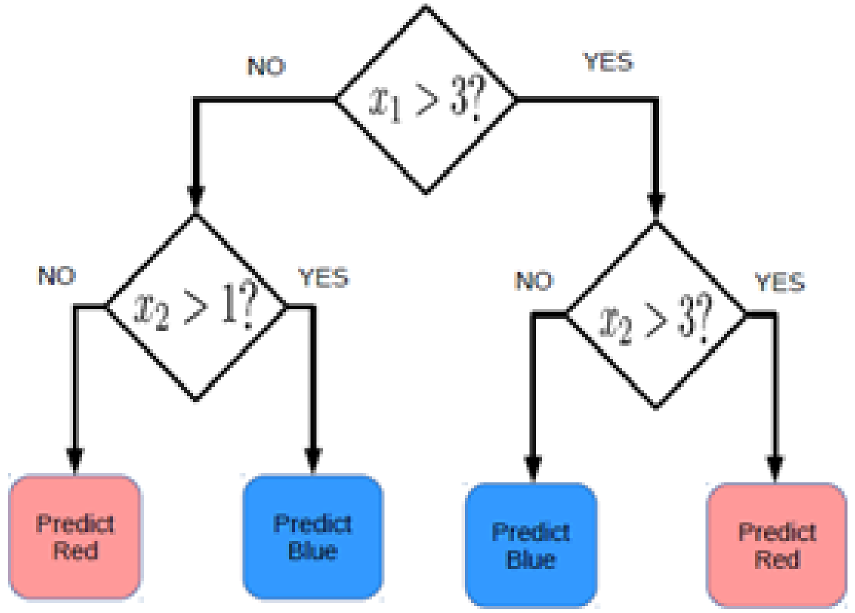
- The “expected” decision boundary given this training data.



- Can you draw corresponding decision tree?

Decision trees





Generic Tree Growing Algorithm

1) Pick the feature that, when parent node is split, results in the largest information gain
2) Stop if child nodes are pure or information gain <= 0
3) Go back to step 1 for each of the two child nodes

Generic Tree Growing Algorithm

- How to split
    - what measurement/criterion as measure of goodness
    - binary vs multi-category split
- When to stop
    - if leaf nodes contain only examples of the same class
    - feature values are all the same for all examples
    - statistical significance test

ID3 -- Iterative Dichotomizer 3

- one of the earlier/earliest decision tree algorithms
- Quinlan, J. R. 1986. Induction of Decision Trees.
- cannot handle numeric features
- no pruning, prone to overfitting
- short and wide trees (compared to CART)
- maximizing information gain/minimizing entropy
- discrete features, binary and multi-category features

C4.5

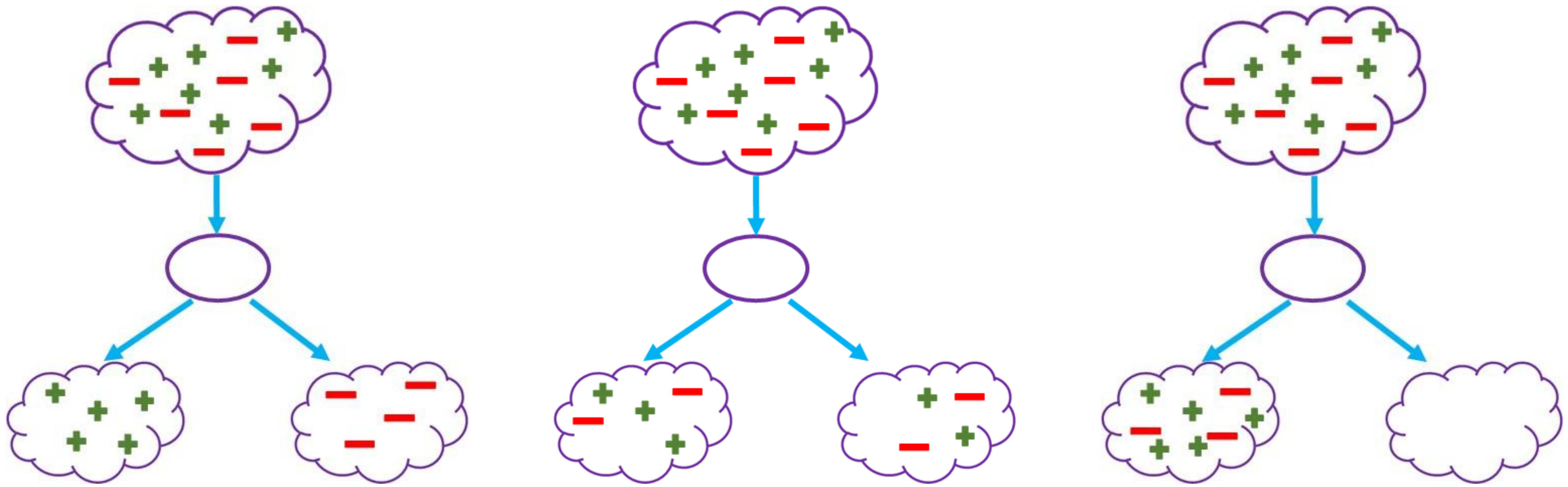
- continuous and discrete features
- Ross Quinlan 1993, Quinlan, J. R. (1993). C4.5: Programming for machine learning.
- continuous is very expensive, because must consider all possible ranges
- handles missing attributes (ignores them in gain compute)
- post-pruning (bottom-up pruning)
- Gain Ratio

CART

- Breiman, L. (1984). Classification and regression trees.
- continuous and discrete features
- strictly binary splits (taller trees than ID3, C4.5)
- binary splits can generate better trees than C4.5, but tend to be larger and harder to interpret
- variance reduction in regression trees
- Gini impurity, twoing criteria in classification trees
- cost complexity pruning

Splitting criteria

- Which “attribute” is the best?



- Prefer splits that makes data “less randomized” after the split

Splitting criteria

What is a good quantitative measure to evaluate effectiveness of an attribute for classification task?

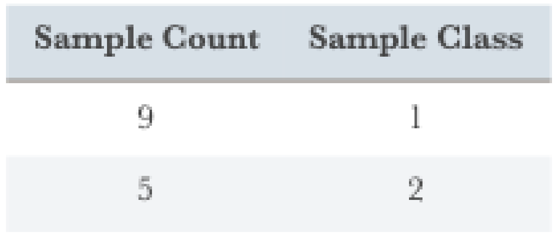
- Information gain measures how well a given attribute / feature separates the training examples according to their target classification.
- ID3 uses information gain (entropy) to measure to select among the candidate attributes at each step while growing the tree.
- “Best attribute” is the one with lowest entropy or highest information gain.

Entropy

Entropy is measure of “randomness”. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one

Given a collection *S*, the entropy of *S* is:

- n-class Entropy -> $E(S) = \sum -(p_i * \log_2 p_i)$
- 2-class Entropy: $E(S) = -(p_1 * \log_2 p_1 + p_2 * \log_2 p_2)$



Sample Count	Sample Class
9	1
5	2

p_1 = 9/(9+5) =9/14 (probability of a sample in the same training class with class 1)
p_2 = 5/14 (probability of a sample in the same training class with class 2)
E = -(9/14 * $\log_2$(9/14) + (5/14) * $\log_2$(5/14))
E = 0.94

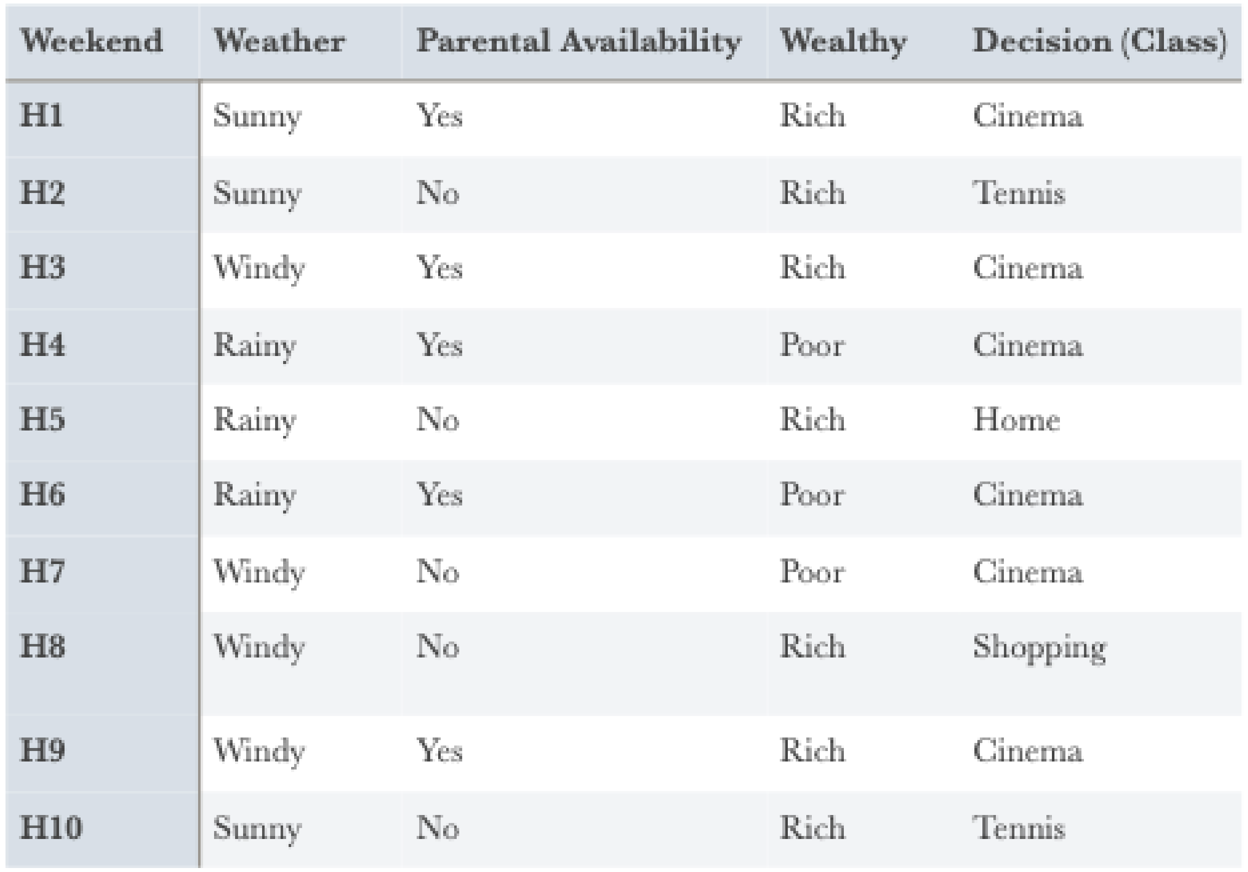
Information Gain

- Given entropy as a measure of the impurity in a collection of training examples, we can now define a measure of the effectiveness of an attribute in classifying the training data. Generally, information gain (IG) is used as this measure of effectiveness.
- The difference in the entropy before and after the split is called information gain.

$$Gain(S,A) = E(before) - G(after\_splitting)$$

$$GAIN(S,A) = E(S) - \sum_{v \in Values(A)} \frac{S_v}{S} E(S_v)$$

Information Gain



Weekend	Weather	Parental Availability	Wealthy	Decision (Class)
H1	Sunny	Yes	Rich	Cinema
H2	Sunny	No	Rich	Tennis
H3	Windy	Yes	Rich	Cinema
H4	Rainy	Yes	Poor	Cinema
H5	Rainy	No	Rich	Home
H6	Rainy	Yes	Poor	Cinema
H7	Windy	No	Poor	Cinema
H8	Windy	No	Rich	Shopping
H9	Windy	Yes	Rich	Cinema
H10	Sunny	No	Rich	Tennis

- E(S) = -((6/10)*log2(6/10) + (2/10)*log2(2/10) + (1/10)*log2(1/10) + (1/10)*log2(1/10))
- E(S) = 1.571
- ***Gain(S,Weather) = ?***

Sunny = 3 (1 Cinema + 2 Tennis)

Windy = 4 (3 Cinema + 1 Shopping)

Rainy = 3 (2 Cinema + 1 Home)

E(S,*sunny*)= - (1/3)*log2(1/3)-(2/3)*log2(2/3) = 0.918

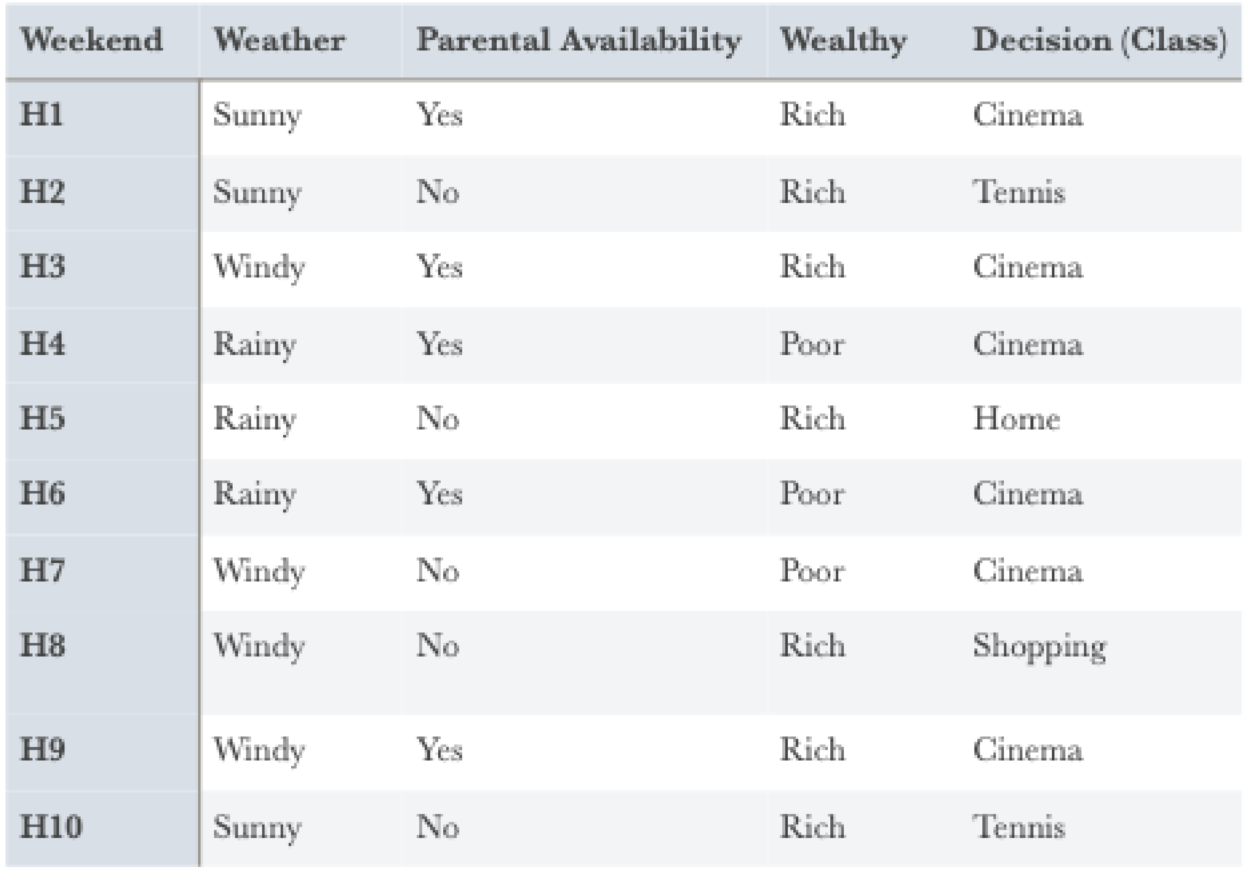
E(S,*windy*)= -(3/4)*log2(3/4)- (1/4)*log2(1/4) = 0.811

E(S*rainy*)= - (2/3)*log2(2/3) - (1/3)*log2(1/3) = 0.918

Gain(S, Weather) = 1.571 — (((1+2)/10)*0.918 + ((3+1)/10)*0.811 + ((2+1)/10)*0.918)

Gain(S, Weather) = 0.70

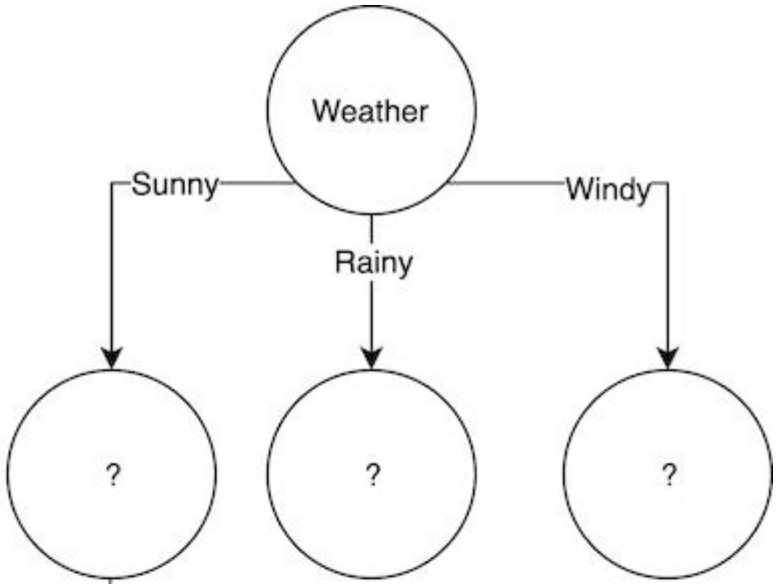
Information Gain

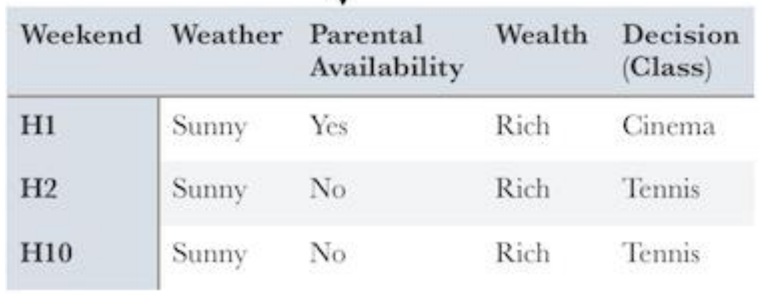


Weekend	Weather	Parental Availability	Wealthy	Decision (Class)
H1	Sunny	Yes	Rich	Cinema
H2	Sunny	No	Rich	Tennis
H3	Windy	Yes	Rich	Cinema
H4	Rainy	Yes	Poor	Cinema
H5	Rainy	No	Rich	Home
H6	Rainy	Yes	Poor	Cinema
H7	Windy	No	Poor	Cinema
H8	Windy	No	Rich	Shopping
H9	Windy	Yes	Rich	Cinema
H10	Sunny	No	Rich	Tennis

- E(S) = -((6/10)*log2(6/10) + (2/10)*log2(2/10) + (1/10)*log2(1/10) + (1/10)*log2(1/10))
- E(S) = 1.571
- **Gain(S, Weather) = 0.70**
- **Gain(S, Parental_Availability) = 0.61**
- **Gain(S, Wealth) = 0.2816**

Information Gain





Weekend	Weather	Parental Availability	Wealth	Decision (Class)
H1	Sunny	Yes	Rich	Cinema
H2	Sunny	No	Rich	Tennis
H10	Sunny	No	Rich	Tennis

Lecture 5 Overview

- Decision trees
- Ensemble learning
- Bagging
- Boosting
- Stacking
- Summary

Ensemble learning

- If different models make different mistakes, can we simply average the predictions?
- Voting Classifier: gives every model a *vote* on the class label
    - Hard vote: majority class wins
    - Soft vote: sum class probabilities $p_{m,c}$ over M models: $\underset{c}{\operatorname{argmax}} \sum_{m=1}^{M} w_c p_{m,c}$
    - Classes can get different weights
- Why does this work?
    - Different models may be good at different 'parts' of data (even if they underfit)
    - Individual mistakes can be 'averaged out' (especially if models overfit)

Ensemble learning

- Bias-variance analysis teaches us that we have two options:
  - If model underfits (high bias, low variance): combine with other low-variance models
    - Need to be different: ‘experts’ on different parts of the data
    - Bias reduction. Can be done with ***Boosting***
  - If model overfits (low bias, high variance): combine with other low-bias models
    - Need to be different: individual mistakes must be different
    - Variance reduction. Can be done with ***Bagging***
- Models must be uncorrelated but good enough (otherwise the ensemble is worse)

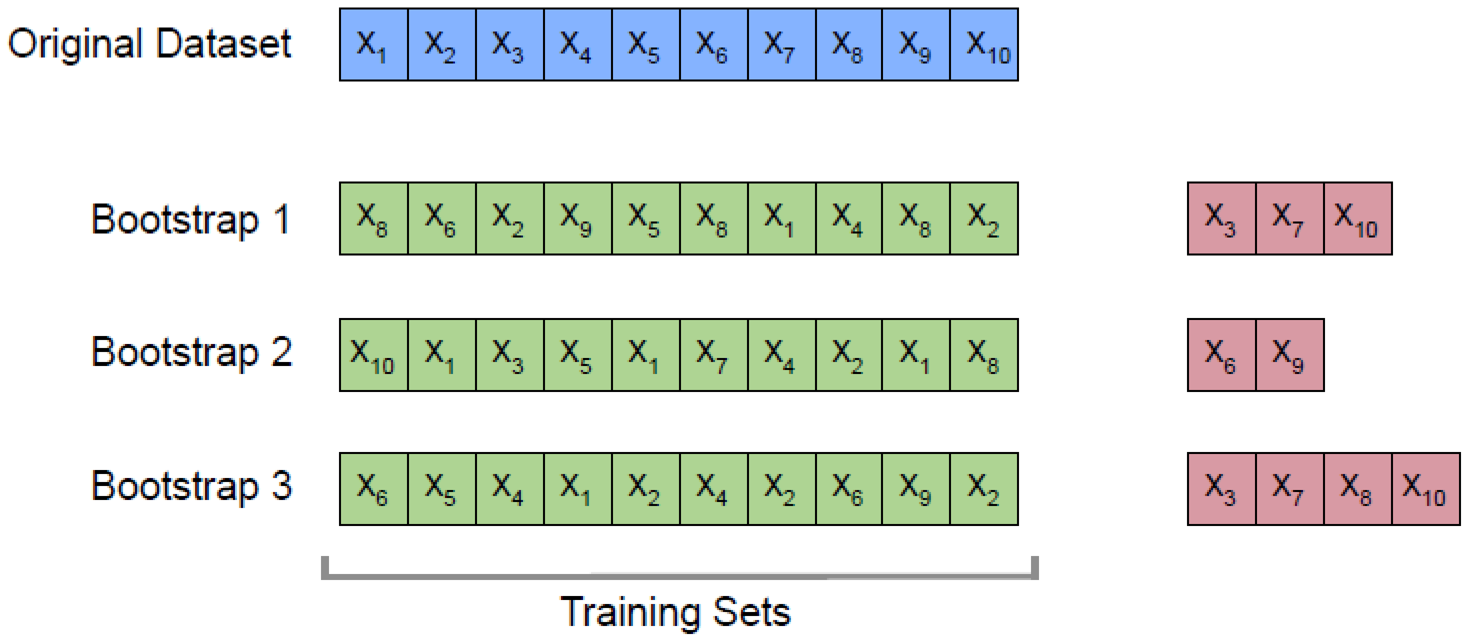
Lecture 5 Overview

- Decision trees
- Ensemble learning
- Bagging
- Boosting
- Stacking
- Summary

Bagging (Bootstrap Aggregating)

- Obtain different models by training the *same* model on *different training samples*
  - Reduce overfitting by averaging out individual predictions (variance reduction)
- In practice: take I bootstrap samples of your data, train a model on each bootstrap
  - Higher I: more models, more smoothing (but slower training and prediction)

Bagging (Bootstrap Aggregating)


Original Dataset
X_1 X_2 X_3 X_4 X_5 X_6 X_7 X_8 X_9 X_{10}
Bootstrap 1
X_8 X_6 X_2 X_9 X_5 X_8 X_1 X_4 X_8 X_2
X_3 X_7 X_{10}
Bootstrap 2
X_{10} X_1 X_3 X_5 X_1 X_7 X_4 X_2 X_1 X_8
X_6 X_9
Bootstrap 3
X_6 X_5 X_4 X_1 X_2 X_4 X_2 X_6 X_9 X_2
X_3 X_7 X_8 X_{10}
Training Sets

Bagging (Bootstrap Aggregating)

- Base models should be unstable: different training samples yield different models
  - E.g. very deep decision trees, or even randomized decision trees
  - Deep Neural Networks can also benefit from bagging (deep ensembles)
- Prediction by averaging predictions of base models
  - Soft voting for classification (possibly weighted)
  - Mean value for regression
- Can produce uncertainty estimates as well
  - By combining class probabilities of individual models (or variances for regression)

Bagging (Bootstrap Aggregating)

Random Forests:

- Uses randomized trees to make models even less correlated (more unstable)
    - At every split, only consider max_features features, randomly selected
- Extremely randomized trees: considers 1 random threshold for random set of features (faster)

Bagging (Bootstrap Aggregating)

- Different implementations can be used. E.g. in scikit-learn:
  - BaggingClassifier: Choose your own base model and sampling procedure
  - RandomForestClassifier: Default implementation, many options
  - ExtraTreesClassifier: Uses extremely randomized trees
- Most important parameters:
  - n_estimators (>100, higher is better, but diminishing returns)
  - max_features
- Easy to parallelize (set n_jobs to -1)
- Fix random_state (bootstrap samples) for reproducibility

Bagging (Bootstrap Aggregating)

- RandomForests don’t need cross-validation: you can use the out-of-bag (OOB) error
- For each tree grown, about 33% of samples are out-of-bag (OOB)
  - Remember which are OOB samples for every model, do voting over these
- OOB error estimates are great to speed up model selection
  - As good as CV estimates, althought slightly pessimistic
- In scikit-learn: oob_error = 1 - clf.oob_score_

Bagging (Bootstrap Aggregating)

- RandomForest are among most widely used algorithms:
  - Don’t require a lot of tuning
  - Typically very accurate
  - Handles heterogeneous features well (trees)
  - Implictly selects most relevant features
- Downsides:
  - less interpretable, slower to train (but parallellizable)
  - don’t work well on high dimensional sparse data (e.g. text)

Lecture 5 Overview

- Decision trees
- Ensemble learning
- Bagging
- **Boosting**
- Stacking
- Summary

Adaptive Boosting (AdaBoost)

- Obtain different models by *reweighting* the training data every iteration
  - Reduce underfitting by focusing on the ‘hard’ training examples
- Increase weights of instances misclassified by the ensemble, and vice versa
- Base models should be simple so that different instance weights lead to different models
  - Underfitting models: decision stumps (or very shallow trees)
  - Each is an ‘expert’ on some parts of the data
- Additive model: Predictions at iteration I are sum of base model predictions

AdaBoost algorithm

- Initialize sample weights: $s_{n,0} = \frac{1}{N}$
- Build a model (e.g. decision stumps) using these sample weights
- Give the *model* a weight w_i related to its weighted error rate $w_{,i} = \lambda\log(\frac{1-\varepsilon}{\varepsilon})$
- Good trees get more weight than bad trees
    - Logit function maps error ε from [0,1] to weight in [-Inf,Inf]
    - Learning rate λ (shrinkage) decreases impact of individual classifiers
        - Small updates are often better but requires more iterations

AdaBoost algorithm

- Update the sample weights
  - Increase weight of incorrectly predicted samples: $s_{n,i+1} = s_{n,i}e^{w_i}$
  - Decrease weight of correctly predicted samples: $s_{n,i+1} = s_{n,i}e^{-w_i}$
  - Normalize weights to add up to 1
- Repeat for I iterations

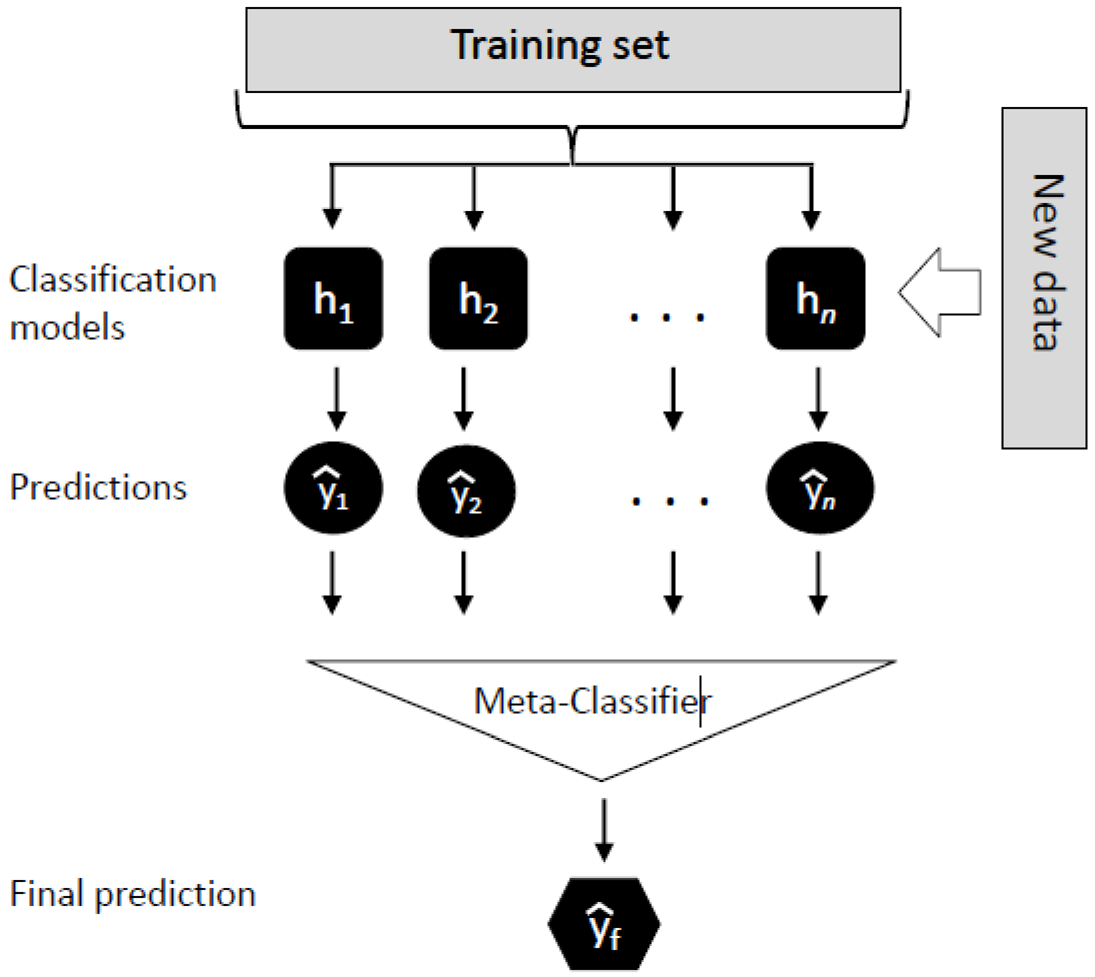
Lecture 5 Overview

- Decision trees
- Ensemble learning
- Bagging
- Boosting
- Stacking
- Summary

Stacking

- Choose M different base-models, generate predictions
- Stacker (meta-model) learns mapping between predictions and correct label
  - Can also be repeated: multi-level stacking
  - Popular stackers: linear models (fast) and gradient boosting (accurate)
- Cascade stacking: adds base-model predictions as extra features
- Models need to be sufficiently different, be experts at different parts of the data
- Can be *very* accurate, but also very slow to predict

Stacking


Training set
New data
Classification models
h_1
h_2
. . .
h_n
Predictions
$\hat{y}_1$
$\hat{y}_2$
. . .
$\hat{y}_n$
Meta-Classifier
Final prediction
$\hat{y}_f$

Lecture 5 Overview

- Decision trees
- Ensemble learning
- Bagging
- Boosting
- Stacking
- Summary

Summary

- Ensembles of voting classifiers improve performance
  - Which models to choose? Consider bias-variance tradeoffs!
- Bagging / RandomForest is a variance-reduction technique
  - Build many high-variance (overfitting) models on random data samples
    - The more different the models, the better
  - Aggregation (soft voting) over many models reduces variance
    - Diminishing returns, over-smoothing may increase bias error
  - Parallellizes easily, doesn’t require much tuning

Summary

- Boosting is a bias-reduction technique
    - Build low-variance models that correct each other’s mistakes
        - By reweighting misclassified samples: AdaBoost
        - By predicting the residual error: Gradient Boosting
    - Additive models: predictions are sum of base-model predictions
        - Can drive the error to zero, but risk overfitting
    - Doesn’t parallelize easily. Slower to train, much faster to predict.
        - XGBoost,LightGBM,… are fast and offer some parallellization
- Stacking: learn how to combine base-model predictions
    - Base-models still have to be sufficiently different

Lab 7 - Ensemble learning