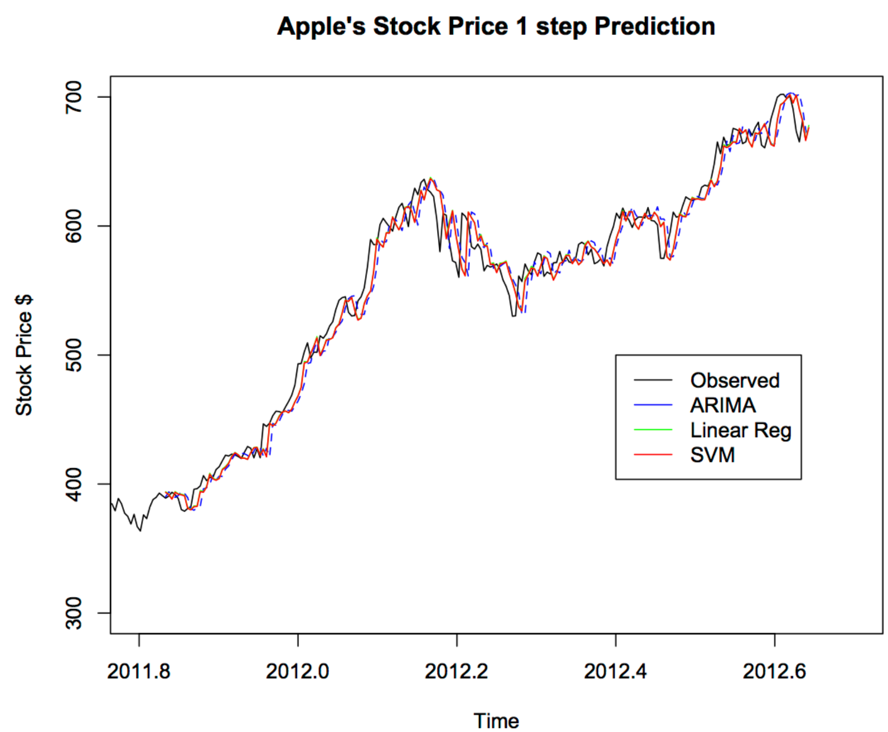
Machine Learning

A Practical Overview

Emma Peng
Aug. 03. 2015

Example

- Predict the price of a stock in 6 months from now, on the basis of company performance measures and economic data



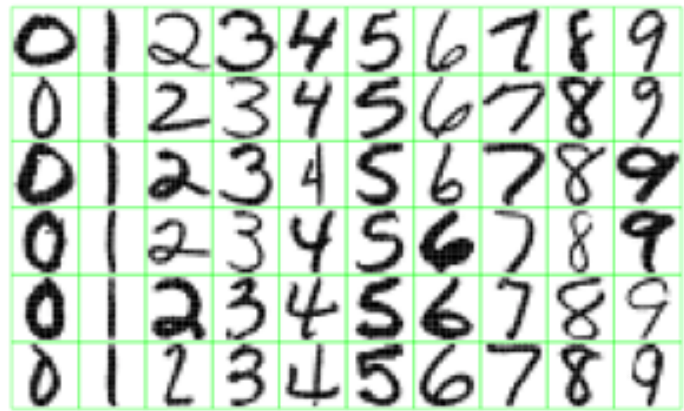
Example

- Classify an email as spam/not spam, base on the frequencies of words appeared in the email

And many more…

- Self driving cars:

- Digit Recognition:



- Recommending System:

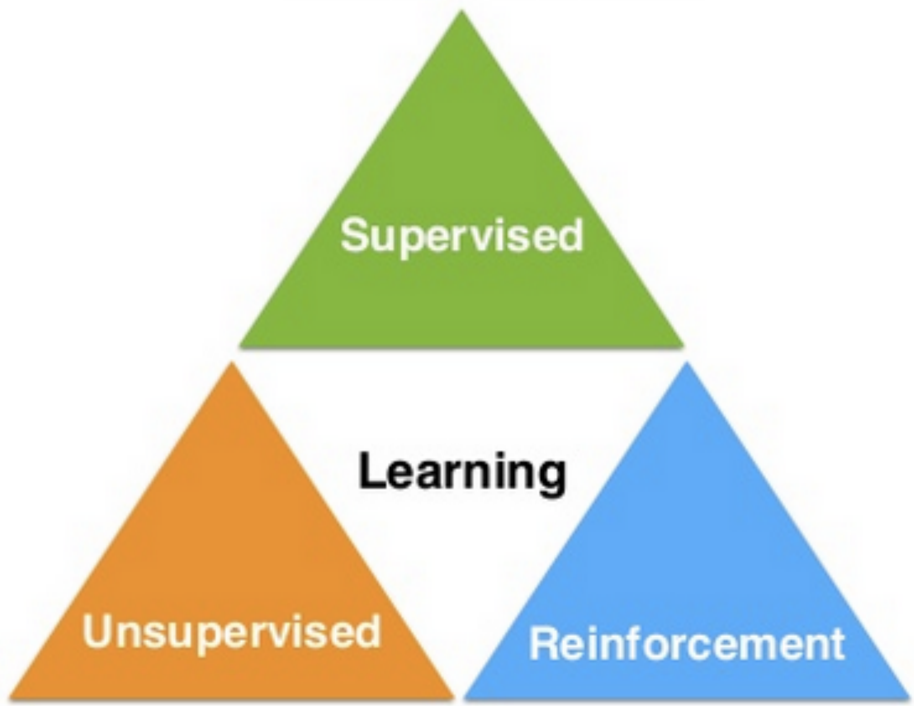
amazon

Agenda

- Overview: Supervised Learning
- Classification
- Common Classifiers

Overview

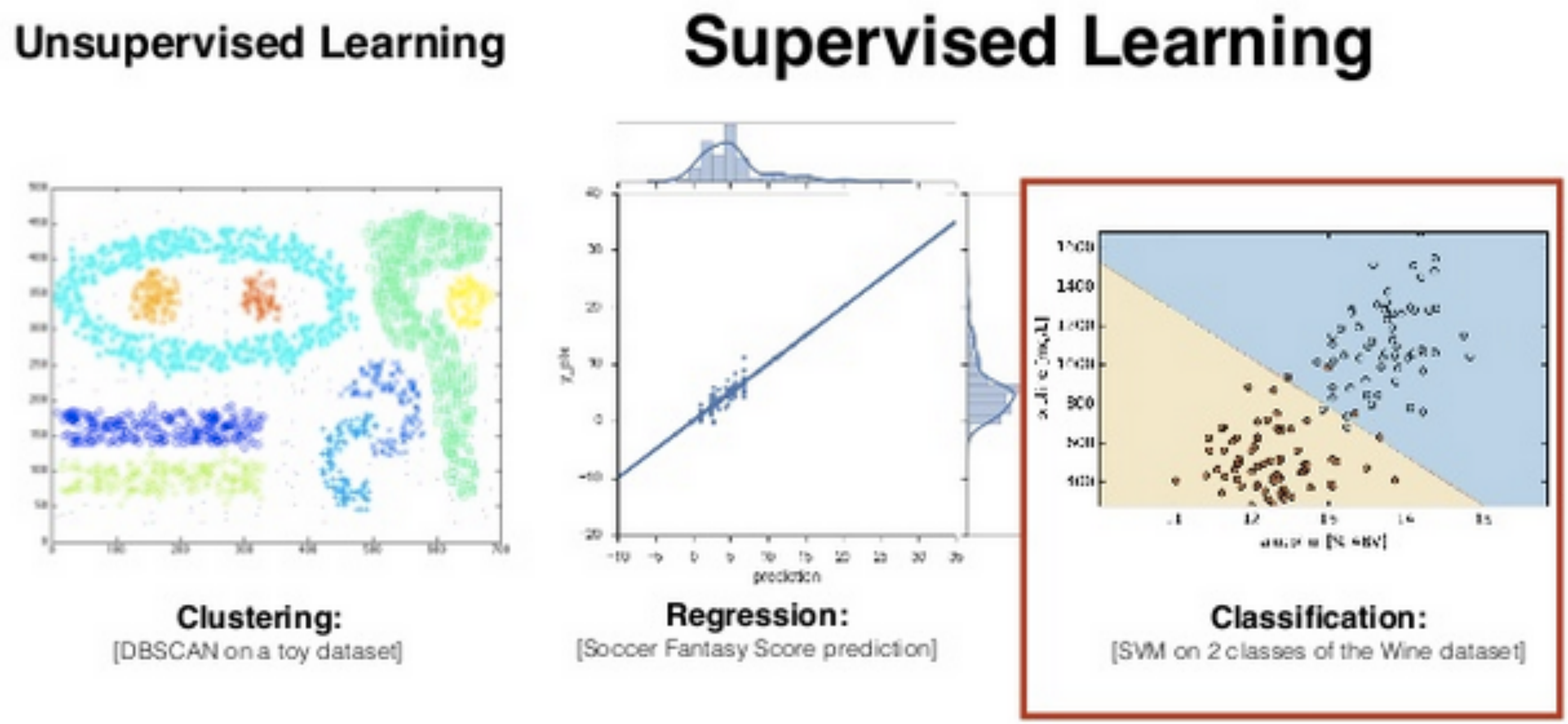
- Labeled data
- Direct feedback
- Predict outcome/future



- No labels
- No feedback
- “Find hidden structure”

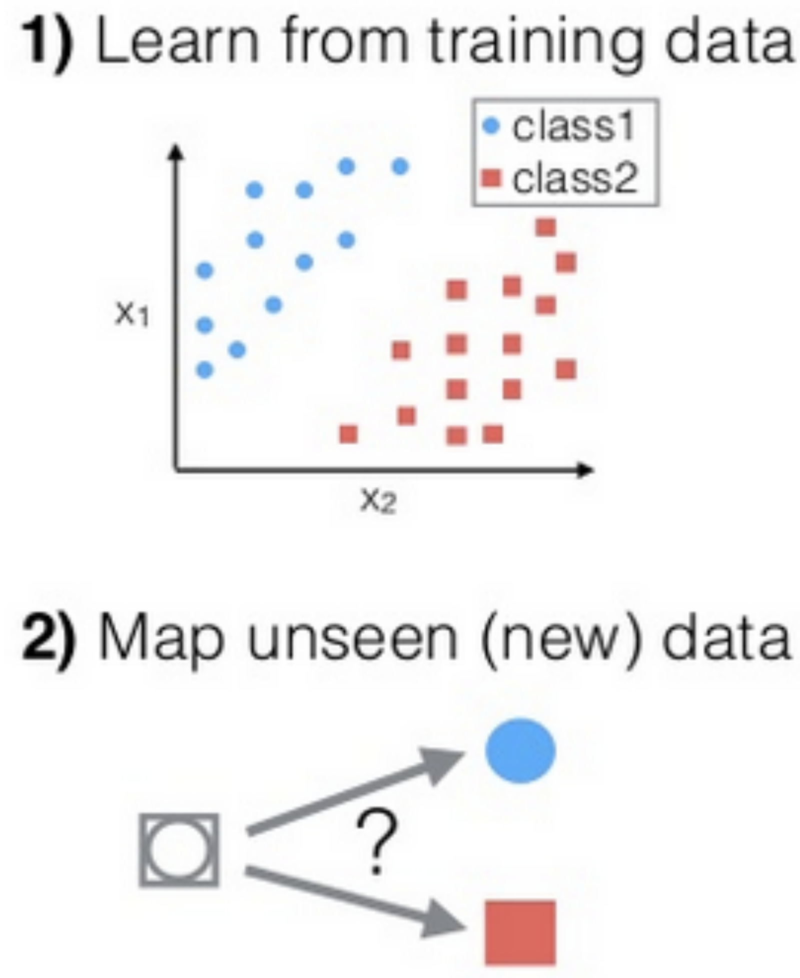
- Decision process
- Reward system
- Learn series of actions

Overview



Today's topic

Classification: 2 steps



Common Classifiers

Neural Networks

$\boxed{\text{Decision Tree}}$

Naive Bayes

$\boxed{\text{Logistic Regression}}$

K-Nearest Neighbors

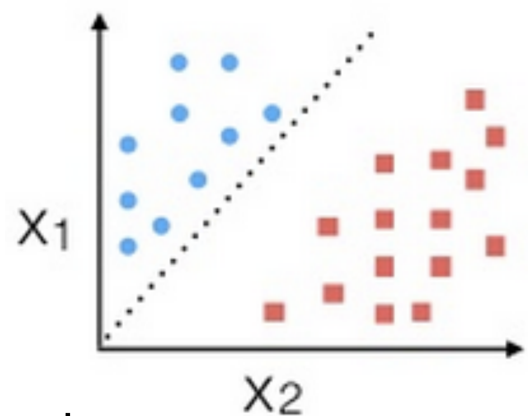
Support Vector Machines

$\boxed{\text{Ensemble Methods: Random Forest, Boosted Trees, etc.}}$

Logistic Regression

- Model (Input X -> Output Y) directly: 2-class case

$$\Pr(G=1|X=x)=\frac{\exp(\beta_0+\beta^Tx)}{1+\exp(\beta_0+\beta^Tx)},$$
$$\Pr(G=2|X=x)=\frac{1}{1+\exp(\beta_0+\beta^Tx)}.$$



- Assumption: Linear decision boundary

$$\log\frac{\Pr(G=1|X=x)}{\Pr(G=2|X=x)}=\beta_0+\beta^Tx.$$

Logistic Regression

- Fitting Logistic Regression Models:
  - Log Likelihood:

$$\ell(\theta) = \sum_{i=1}^{N} \log p_{g_i}(x_i; \theta)$$

where $p_k(x_i; \theta) = \Pr(G = k | X = x_i; \theta).$

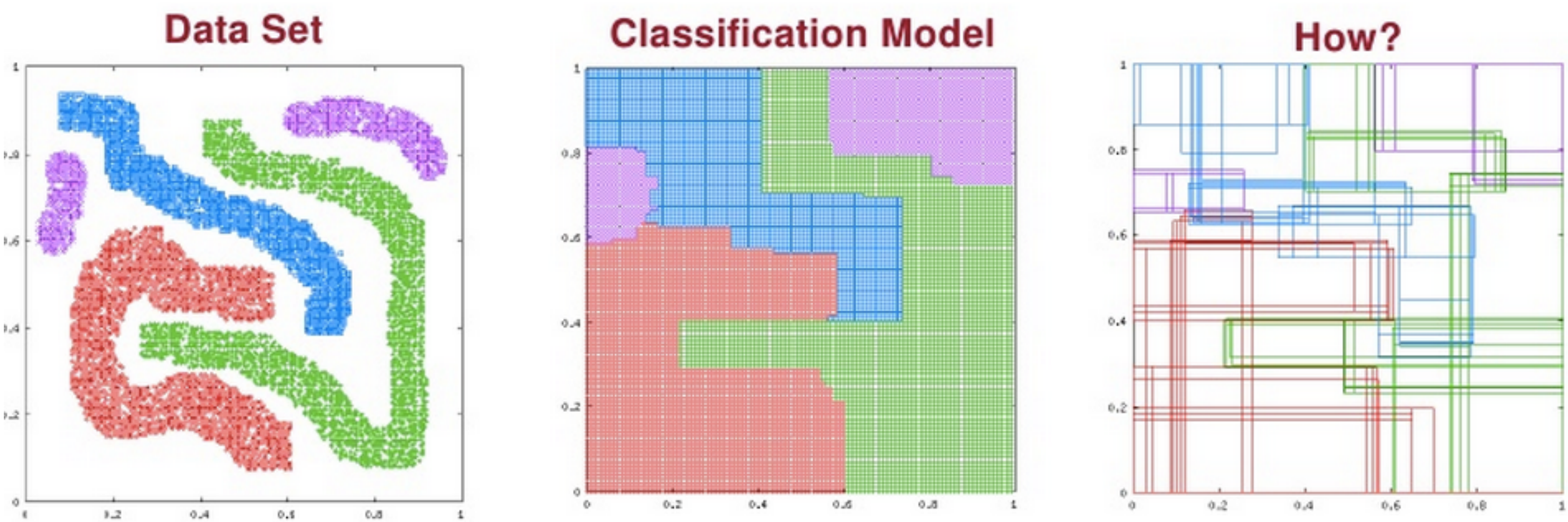
  - Maximize Log Likelihood: e.g. gradient descent

Logistic Regression

- Pros:
  - Simple and fast to train
  - Low variance, robust to noise, less prone to over-fitting
- Cons:
  - Assume linear decision boundary, high bias, can hardly handle categorical features
- Used as our baseline model

Decision Tree

- One of the most widely used technique for classification:



Decision Tree

- Best known algorithm: C4.5 by Ross Quinlan

1. Check for base cases
2. For each attribute *a*
    1. Find the normalized information gain ratio from splitting on *a*
3. Let *a_best* be the attribute with the highest normalized information gain
4. Create a decision *node* that splits on *a_best*
5. Recur on the sublists obtained by splitting on *a_best*, and add those nodes as children of *node*

Implicit feature selection

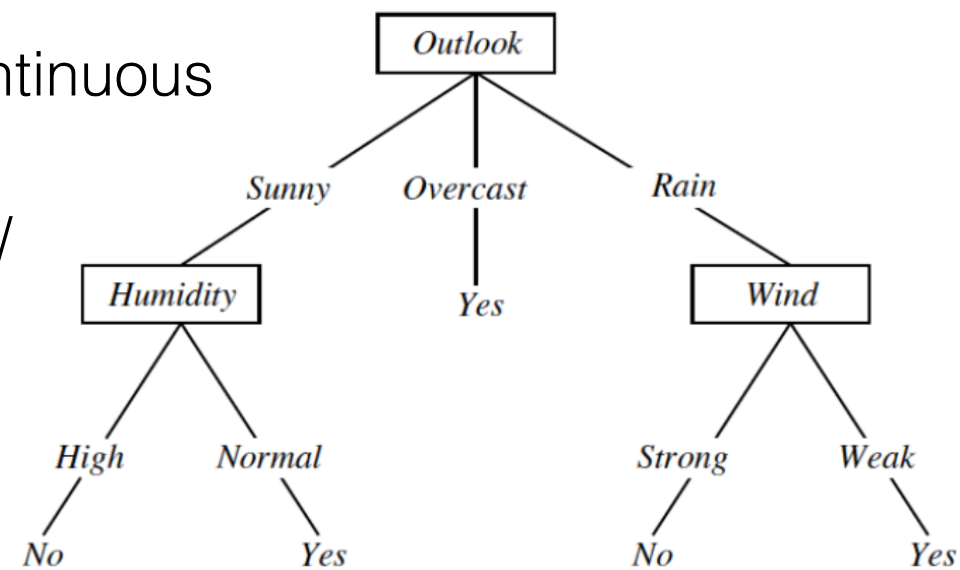
$$\text{Entropy} = \sum_i -p_i \log_k p_i$$

e.g., $2\,(-\,0.5 \log_2(0.5)) = 1$

Information Gain =
entropy(parent) – [avg entropy(children)]

C4.5

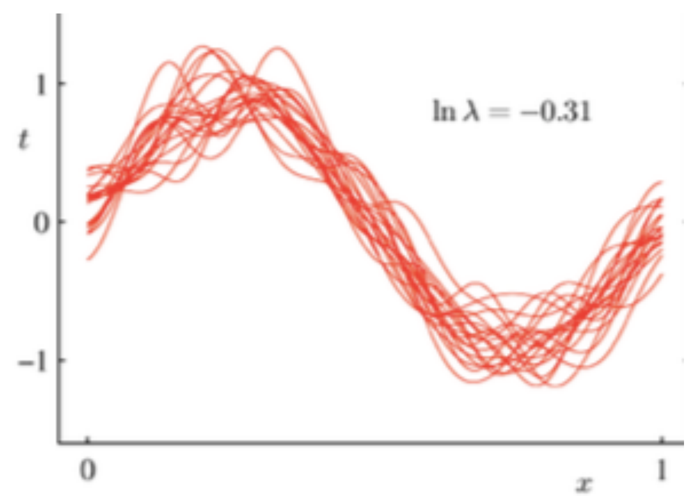
- Pros:
  - Simple to understand and interpret, implicit feature selection
  - Require little data preparation
  - Can deal with categorical/ continuous attributes
  - Avoid over-fitting: pre-pruning/ post-pruning
- Cons:
  - Assume decision surfaces are parallel to axis



Play tennis?

Random Forest

- Rational: the combination of learning models increases the classification accuracy (Bagging)
- Bagging: to average noisy and unbiased models to create a model with low variance (bias-variance tradeoff)



Random Forest

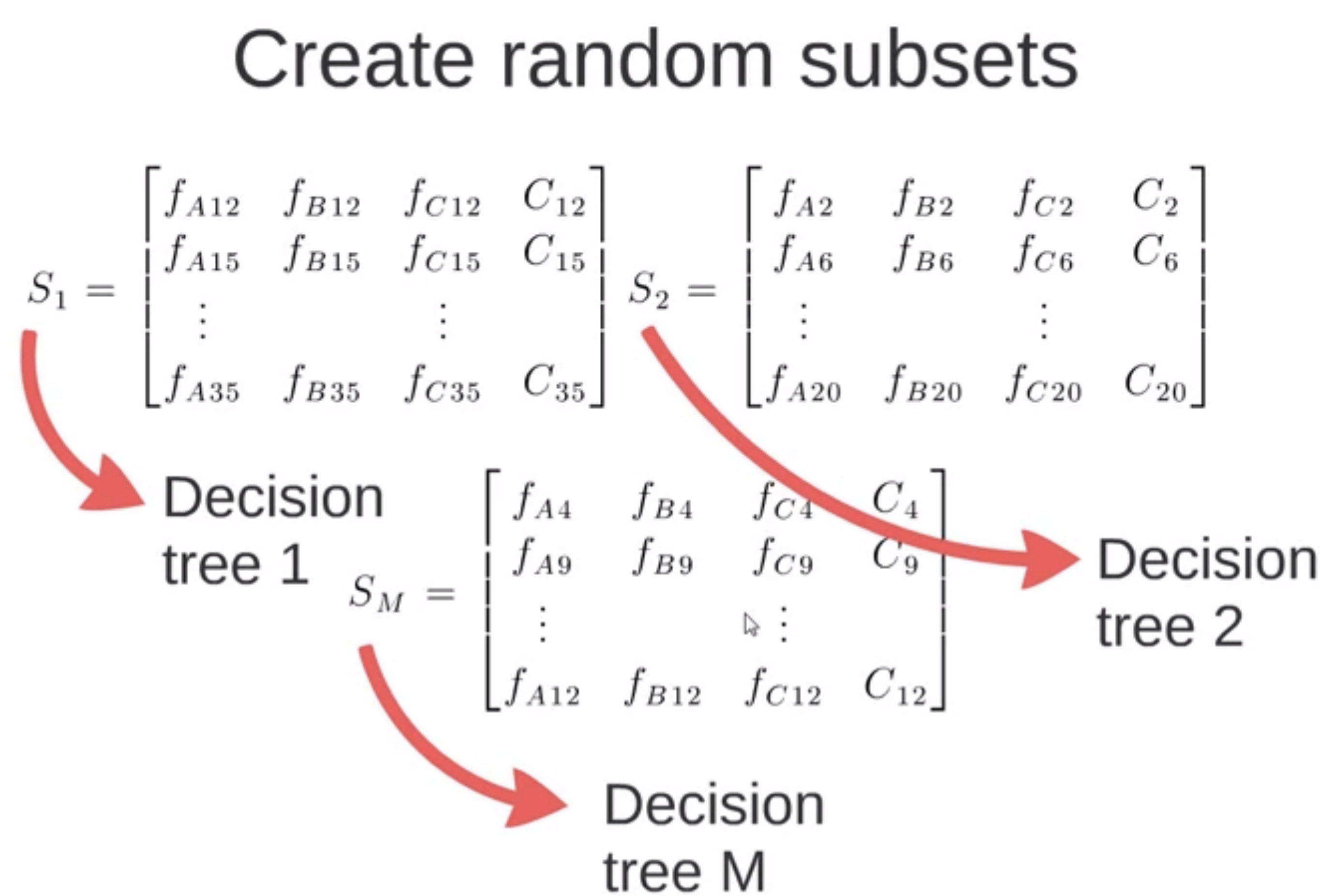
- Works as a large collection of un-correlated decision trees

- Input:

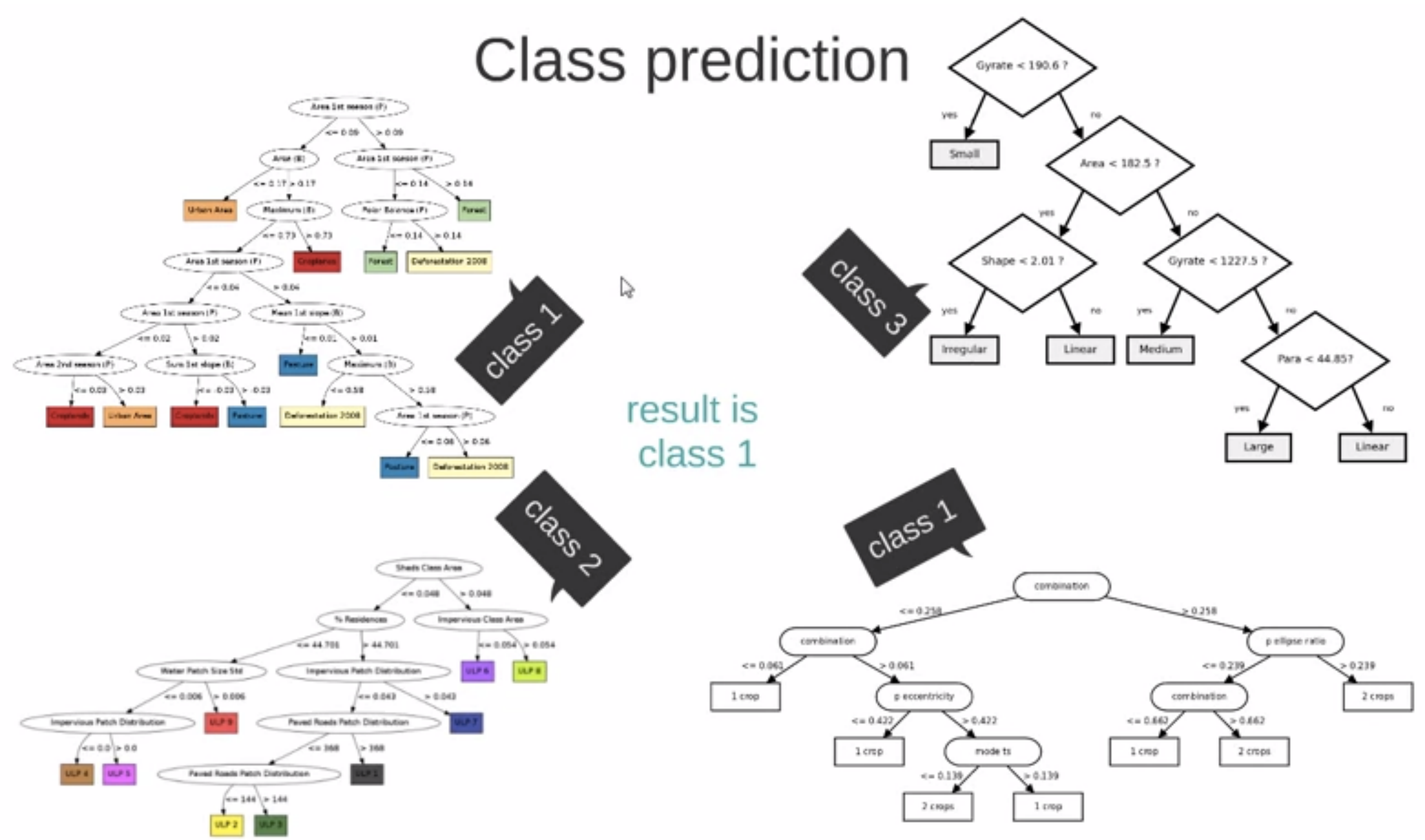
feature A of the 1st sample

$$S = \begin{bmatrix} f_{A1} & f_{B1} & f_{C1} & C_1 \\ \vdots & & \vdots & \\ f_{AN} & f_{BN} & f_{CN} & C_N \end{bmatrix}$$

Random Forest



Random Forest



Random Forest

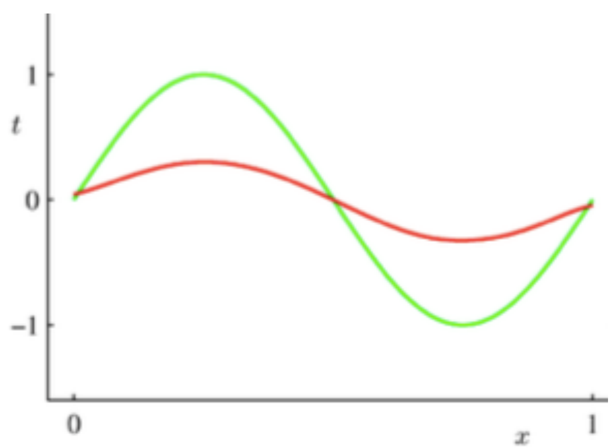
- Pros:
  - Good empirical results
  - Fast to train: decision trees in a forest can be trained in parallel
- Cons:
  - Generally require deep trees

Gradient Boosted Trees

- Boosting:

  e.g. Decision trees/ Stumps

  - Sequentially add new models to the ensemble
  - At each iteration, a new weak, base-learner model is trained with respect to the error of the whole ensemble learned so far
  - Reduce bias of weak models (bias-variance tradeoff)



Algorithm 1

Friedman's Gradient Boost algorithm.

Inputs:

- input data $(x, y)^{N}_{i=1}$
- number of iterations M
- choice of the loss-function $\Psi(y, f)$
- choice of the base-learner model $h(x, \theta)$

Algorithm:

1: initialize $\widehat{f}_0$ with a constant
2: **for** $t = 1$ to M **do**
3: compute the negative gradient $g_t(x)$
4: fit a new base-learner function $h(x, \theta_t)$
5: find the best gradient descent step-size ρ_t:

$$\rho_t = \arg\min_{\rho} \sum_{i=1}^{N} \Psi\left[y_i,\ \widehat{f}_{t-1}(x_i) + \rho h(x_i,\ \theta_t)\right]$$

6: update the function estimate:

$$\widehat{f}_t \leftarrow \widehat{f}_{t-1} + \rho_t h(x,\ \theta_t)$$

7: **end for**

Gradient Boosted Trees

- Pros:
  - Good empirical results
  - Require shallower trees
- Cons:
  - Can overfit with too many trees
  - Slower to train

Thank you!