COGS 118A, Winter 2020

Supervised Machine Learning Algorithms

Lecture 16: Ensemble Methods

Zhuowen Tu

Ensemble Learning

Empirical Comparisons of Different Algorithms

Caruana and Niculesu-Mizil, ICML 2006

MODEL	1ST	2ND	3RD	4TH	5TH	6TH	7TH	8TH	9TH	10TH
BST-DT	0.580	0.228	0.160	0.023	0.009	0.000	0.000	0.000	0.000	0.000
RF	0.390	0.525	0.084	0.001	0.000	0.000	0.000	0.000	0.000	0.000
BAG-DT	0.030	0.232	0.571	0.150	0.017	0.000	0.000	0.000	0.000	0.000
SVM	0.000	0.008	0.148	0.574	0.240	0.029	0.001	0.000	0.000	0.000
ANN	0.000	0.007	0.035	0.230	0.606	0.122	0.000	0.000	0.000	0.000
KNN	0.000	0.000	0.000	0.009	0.114	0.592	0.245	0.038	0.002	0.000
BST-STMP	0.000	0.000	0.002	0.013	0.014	0.257	0.710	0.004	0.000	0.000
DT	0.000	0.000	0.000	0.000	0.000	0.000	0.004	0.616	0.291	0.089
LOGREG	0.000	0.000	0.000	0.000	0.000	0.000	0.040	0.312	0.423	0.225
NB	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.030	0.284	0.686

Overall rank by mean performance across problems and metrics (based on bootstrap analysis).

BST-DT: boosting with decision tree weak classifier

BAG-DT: bagging with decision tree weak classifier

ANN: neural nets

BST-STMP: boosting with decision stump weak classifier

LOGREG: logistic regression

RF: random forest

SVM: support vector machine

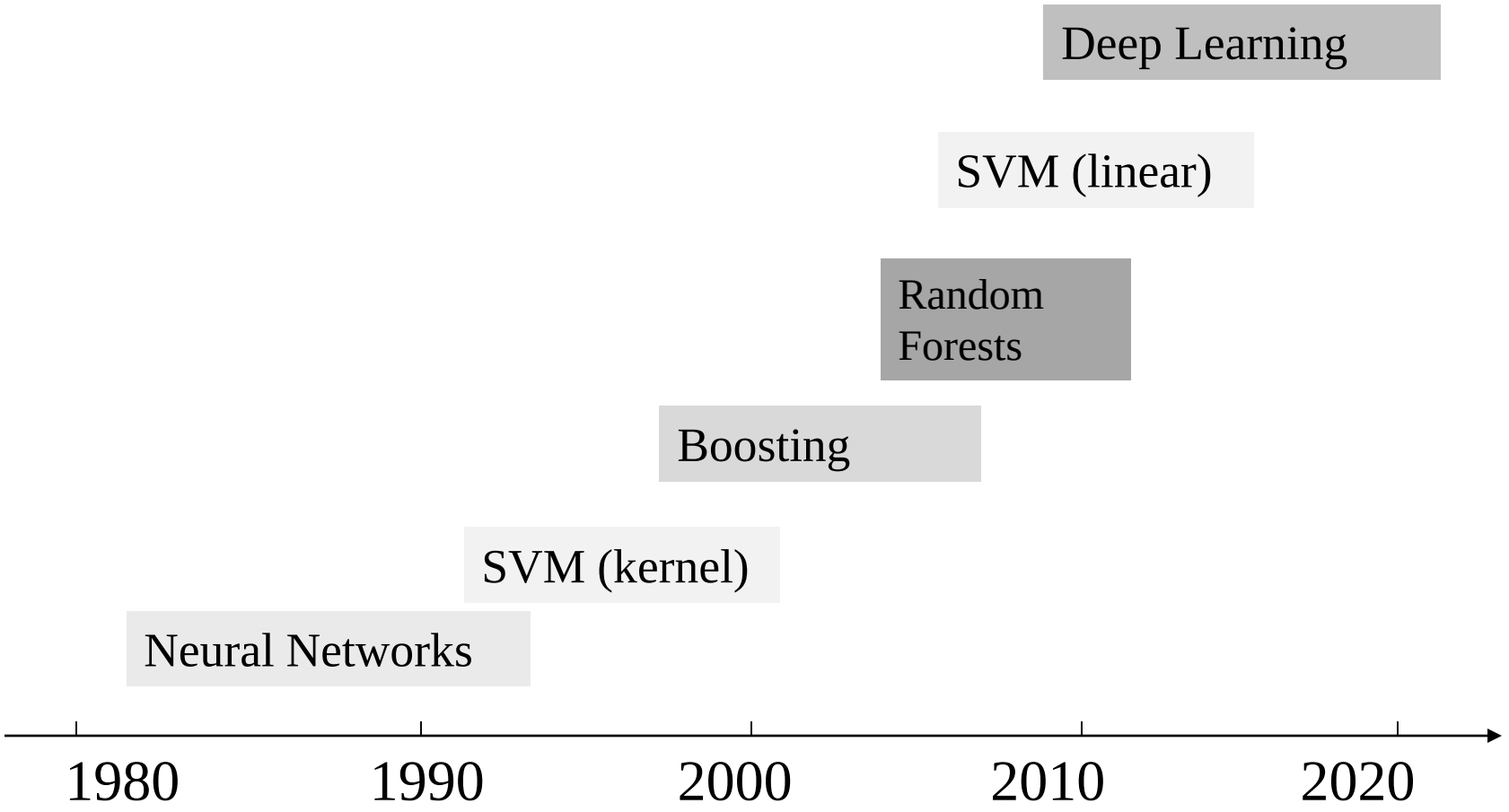
KNN: k nearest neighboorhood

DT: decision tree

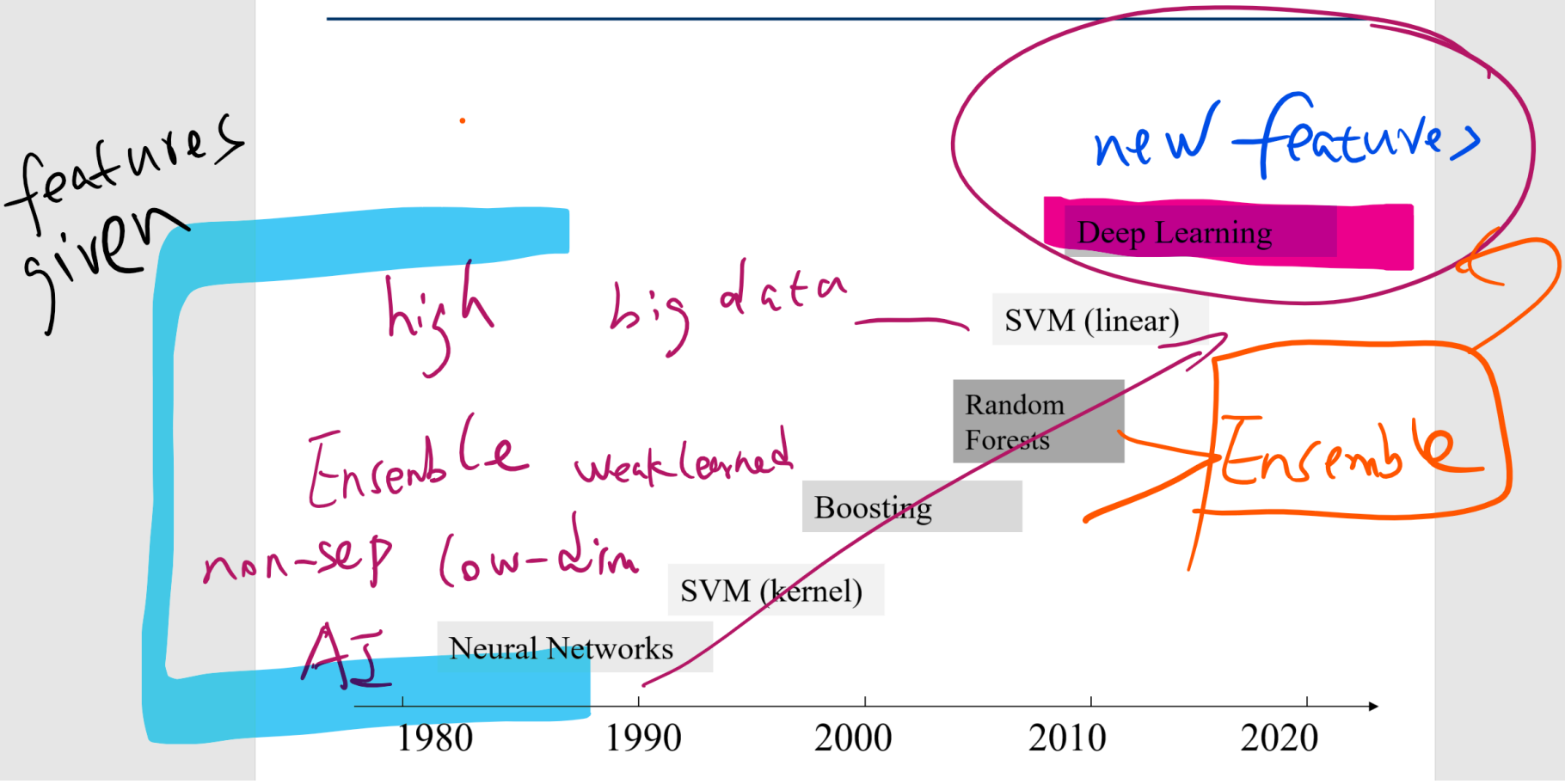
NB: naïve Bayesian

It is informative, but by no means final.

Trends of classification methods



Trends of classification methods



Ensemble Methods

Bagging (Breiman 1994,…)

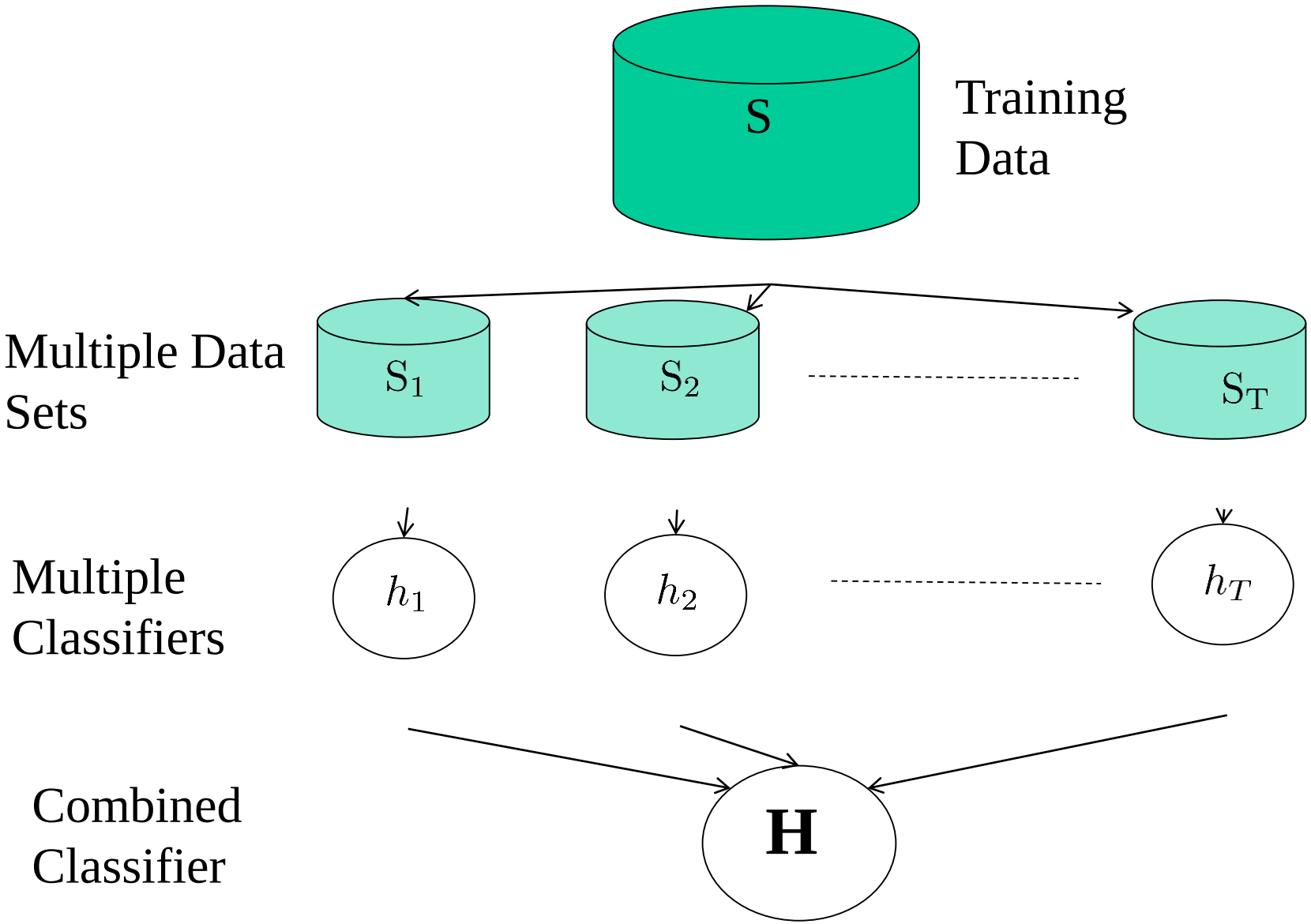
Random forests (Breiman 2001,…)

Boosting (Freund and Schapire 1995, Friedman et al. 1998,…)

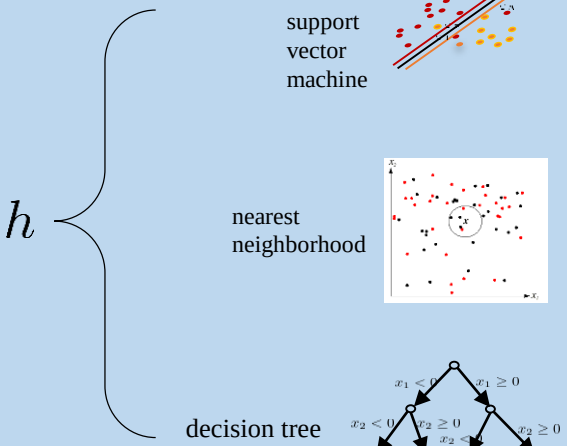
Predict class label for unseen data by aggregating a set of predictions (classifiers learned from the training data).

General Idea

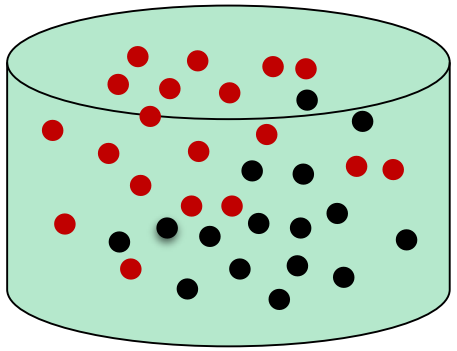
$$S = \{(\mathbf{x}_i, y_i), i = 1..n\}$$



Sampling with replacement

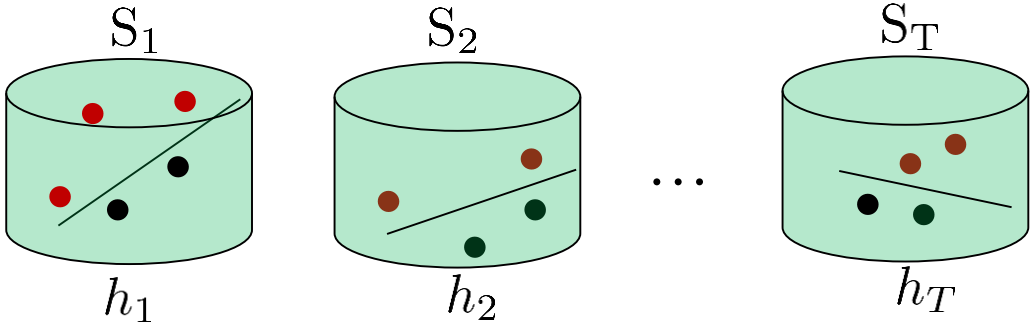


$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$



$y = +1$

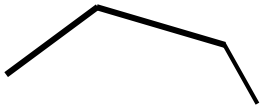
$y = -1$



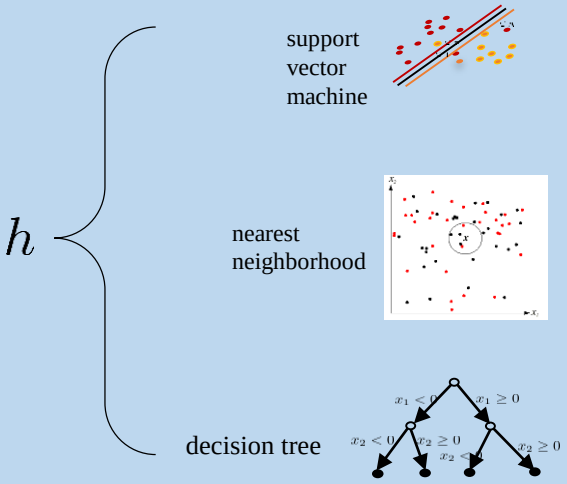
α_1 α_2 α_T

$$H(\mathbf{x}) = sign(\sum_{t=1}^{T} \alpha_t \times h_t(\mathbf{x}))$$

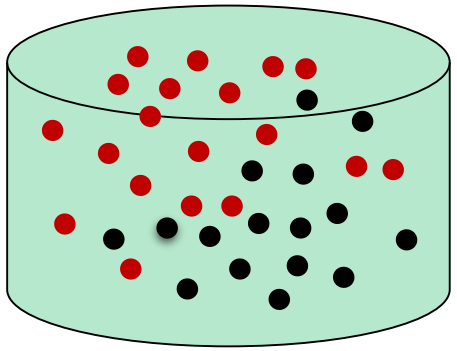
$h_t(\mathbf{x}) \in \{-1, +1\}$



Sampling without replacement

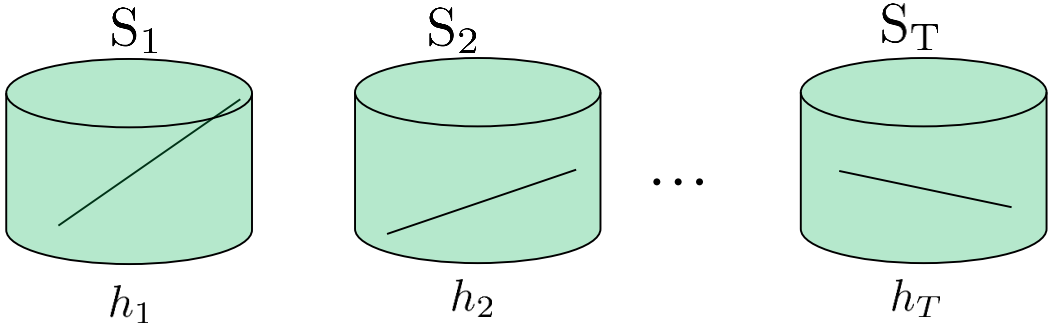


$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$



$y = +1$

$y = -1$



α_1 α_2 α_T

$$H(\mathbf{x}) = sign(\sum_{t=1}^{T} \alpha_t \times h_t(\mathbf{x}))$$

$h_t(\mathbf{x}) \in \{-1, +1\}$

Why do they work?

1. Suppose there are 25 base classifiers.
2. Each classifier has error rate, $\epsilon = 0.35$.
3. Assume independence among classifiers.
4. Probability that the ensemble classifier makes a wrong prediction:

Ensemble classifier: $sign(\sum_{t=1}^{25} h_t)$ $\quad\quad$ $h_t \in \{-1, +1\}$

Possible final error?

A. 0.5?
B. 0.35?
C. 0.2?
D. 0.06?

Why do they work?

1. Suppose there are 25 base classifiers.
2. Each classifier has error rate, $\epsilon = 0.35$.
3. Assume independence among classifiers.
4. Probability that the ensemble classifier makes a wrong prediction:

Ensemble classifier: $sign(\sum_{t=1}^{25} h_t)$ $\quad\quad$ $h_t \in \{-1, +1\}$

Possible final error?

A. 0.5?
B. 0.35?
C. 0.2?
D. 0.06?

1. Suppose there are 25 base classifiers.

2. Each classifier has error rate $\epsilon = 0.35$.

3. Assume independence among classifiers.

4. Probability that the ensemble classifier makes a wrong prediction:

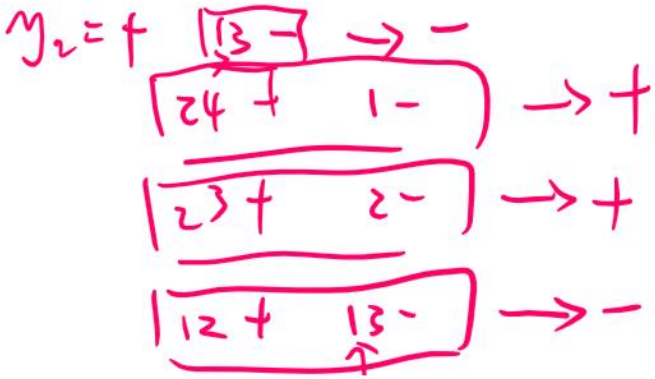
Ensemble classifier: $sign(\sum_{t=1}^{25} h_t)$ $\quad h_t \in \{-1, +1\}$

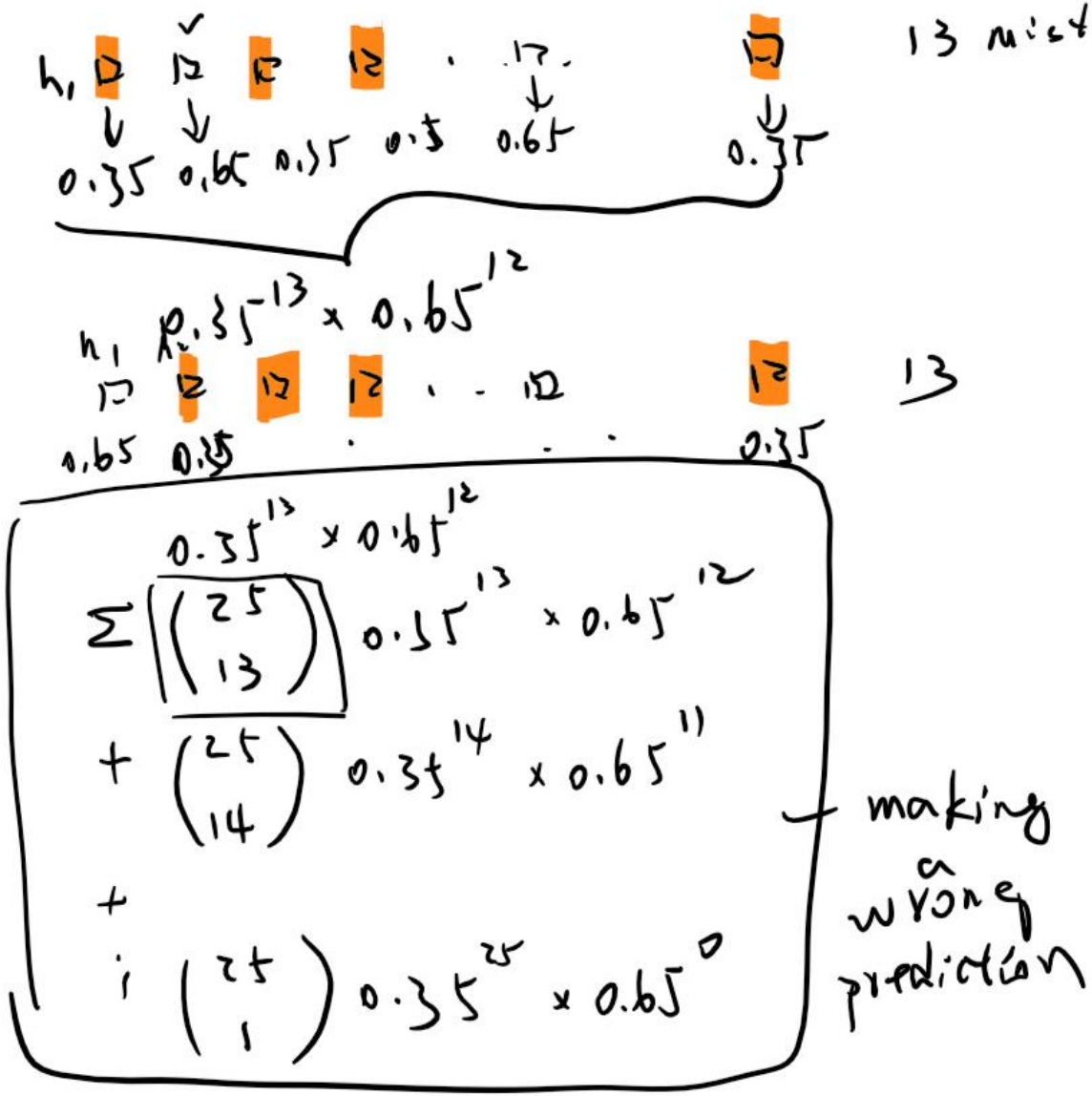
$$error = \sum_{t=13}^{25} \binom{25}{t} \epsilon^t (1-\epsilon)^{25-t} = 0.06$$

$H(X) = sign(\sum_{t=1}^{25} h_t)$

$H(x_i) \neq y_i$ when there are more mistakes than corrected answers.

at least 13 errors




13 mistakes
making a wrong prediction

Bagging (Breiman 1994)

Training:

1. Given a dataset S, at each iteration t, a training set S_t is sampled with replacement from S (i.e. bootstraping).

2. A classifier h_t is learned for each S_t

Classification: given an unseen sample $\mathbf{x}$,

1. Each classifier h_t returns its class prediction

2. The bagged classifier H counts the votes and assigns the class with the most votes to $\mathbf{x}$.

Regression: can be applied to the prediction of continuous values by taking the average value of each prediction.

Bagging

- Bagging works because it reduces variance by voting/averaging
  - In some pathological hypothetical situations the overall error might increase.
  - Usually, the more classifiers the better.
- Problem: we only have one dataset.
- Solution: generate new ones of size n by bootstrapping, i.e. sampling it with replacement
- Can help a lot if data is noisy.

Build Ensemble Classifiers

- Basic idea:

  Build different “experts”, and let them vote

- Advantages:

  Improve predictive performance

  Other types of classifiers can be directly included

  Easy to implement

  No too much parameter tuning

- Disadvantages:

  The combined classifier is not so transparent (black box)

  Not a compact representation

Why do they work?

1. Suppose there are 25 base classifiers.

2. Each classifier has error rate, $\epsilon = 0.35$.

3. Assume independence among classifiers.

4. Probability that the ensemble classifier makes a wrong prediction:

Ensemble classifier: $sign(\sum_{t=1}^{25} h_t)$ $\qquad$ $h_t \in \{-1, +1\}$

$$error = \sum_{t=13}^{25} \binom{25}{t} \epsilon^t (1-\epsilon)^{25-t} = 0.06$$

Why Bagging works?

Let $S = \{(y_i, \mathbf{x}_i, i = 1..n\}$ be the set of training dataset.

Let $\{S_t\}$ be a sequence of training sets containing a sub-set of S.

Let P be the underlying distribution of S.

Bagging replaces the prediction of the model with the majority of the predictions given by the classifiers

$$H_A(\mathbf{x}, P) = E_S(H(\mathbf{x}, S_t))$$

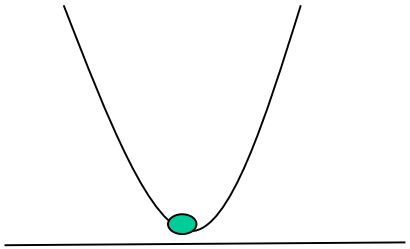
Why Bagging works?

$$H_A(\mathbf{x}, P) = E_S(H(\mathbf{x}, S_t))$$

Direct error: $e = E_S E_{y,\mathbf{x}}[y - H(\mathbf{x}, S)]^2$

Bagging error: $e_A = E_{y,\mathbf{x}}[y - H_A(\mathbf{x}, P)]^2$

Jensens inequality: $E[Z]^2 \leq E[Z^2]$



$$e = E[y^2] - 2E[yH_A] + E_{y,\mathbf{x}}E_S[H^2(\mathbf{x}, S)]$$

$$\geq E[y - H_A]^2 = e_A$$

Why Bagging works?

$$H_A(\mathbf{x}, P) = E_S(H(\mathbf{x}, S_t))$$

Direct error: $e = E_S E_{y,\mathbf{x}}[y - H(\mathbf{x}, S)]^2$

Bagging error: $e_A = E_{y,\mathbf{x}}[y - H_A(\mathbf{x}, P)]^2$

Jensens inequality: $(E[Z])^2 \leq E[Z^2]$

$e = E[y^2] - 2E[yH_A] + E_{y,\mathbf{x}}E_S[H^2(\mathbf{x}, S)]$

$\geq E[y - H_A]^2 = e_A$

f(v) convex

$a f(v_0) + (1-a) f(v_1)$

$\geq f(a v_0 + (1-a) v_1)$

$E[f(v)] \geq f(E(v))$

$f(v) = v^2$

$E[v^2] \geq (E(v))^2$

$E(y^2) - 2E[y] \cdot [H_A]^2$

$(E_{x,y}[H_A])^2$

Bias-variance Decomposition

- Used to analyze how much selection of any specific training set affects performance
- Assume infinitely many classifiers, built from different training sets
- For any learning scheme,
  - Bias = expected error of the combined classifier on new data
  - Variance = expected error due to the particular training set used
- Total expected error ~ bias + variance

When does Bagging work?

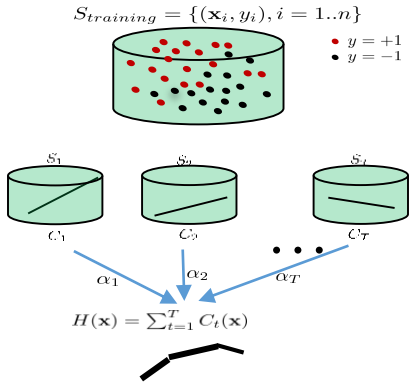
- Learning algorithm is unstable: if small changes to the training set cause large changes in the learned classifier.
- If the learning algorithm is unstable, then Bagging almost always improves performance.
- Some candidates:

  Decision tree, decision stump, regression tree, linear regression, SVMs

Randomization

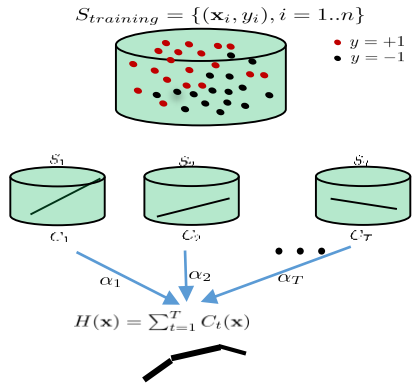
- Can randomize learning algorithms instead of inputs.
- Some algorithms already have random component: e.g. random initialization.
- Most algorithms can be randomized.
  - Pick from the N best options at random instead of always picking the best one.
  - The splitting rule in decision tree.
- Random projection in kNN (Freund and Dasgupta 08).

Which method cannot be used as the base weak classifier for the Bagging classifier?

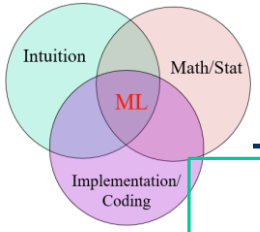


A. SVM
B. Logistic regression classifier
C. KNN classifier
D. Decision tree
E. None of the above

Which method cannot be used as the base weak classifier for the Bagging classifier?



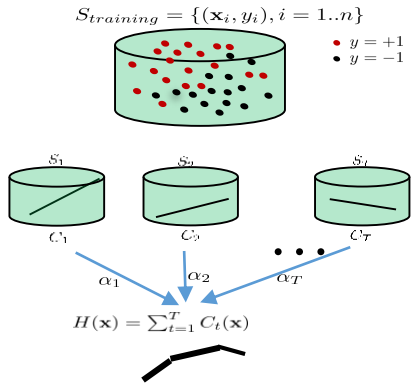
A. SVM
B. Logistic regression classifier
C. KNN classifier
D. Decision tree
E. None of the above

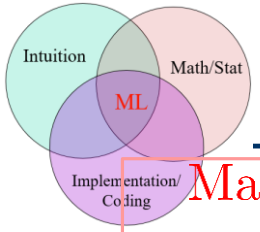


Recap: Bagging

Intuition: Bagging (ensemble) learning works almost all the time by combining a few weak classifiers.

- Any standard classifiers such as SVM, logistic regression, decision tree, can be combined in bagging.
- The more variant the base weak classifiers are, the more apparent the effect will be for bagging.
- Typically, combing 50-200 weak classifiers works the best in bagging.
- The training process can be highly efficient by having the weak classifiers trained in parallel.





Recap: Bagging

Math:

$$H(\mathbf{x}) = \sum_{t=1}^{T} h_t(\mathbf{x})$$

Implementation:

Given a training set S
For $t = 1$ to T do:
    Build subset S_t by sampling with replacement from S
    Learn classifier h_t from S_t
Make predictions according to majority vote of the set of T classifiers.

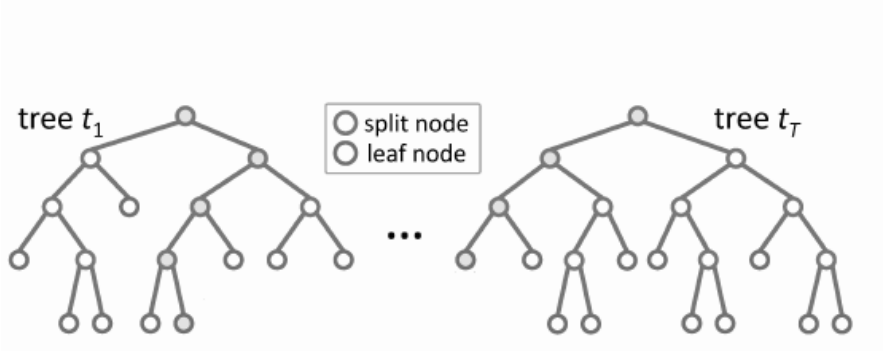
Ensemble Methods

Bagging (Breiman 1994,…)

Random forests (Breiman 2001,…)

Boosting (Freund and Schapire 1995, Friedman et al. 1998,…)

Random Forests

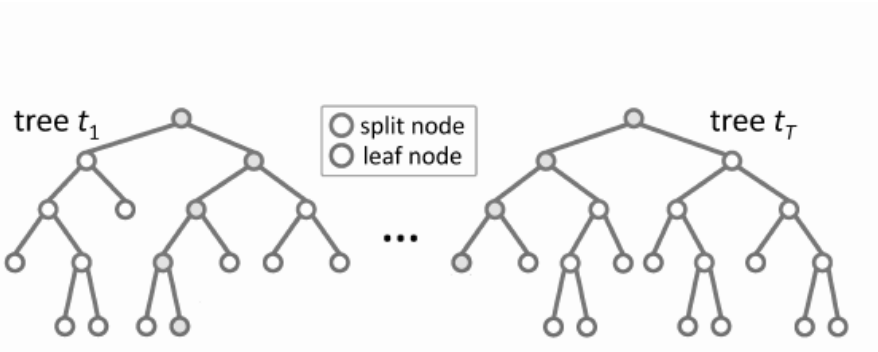


- Random forests (RF) are a combination of tree predictors
- Each tree depends on the values of a random vector sampled independently。
- The generalization error depends on the strength of the individual trees and the correlation between them.
- Using a random selection of features yields results favorable to AdaBoost, and are more robust w.r.t. noise.

The Random Forests Algorithm

Given a training set S
For $t = 1$ to T do:
    Build subset S_t by sampling with replacement from S
    Learn tree h_t from S_t
        At each node:
            Choose best split from random subset of F features
        Each tree grows to the largest extend, and no pruning
Make predictions according to majority vote of the set of T trees.

Random Forests



The overall classifier:

$$H(\mathbf{x}) = sign(\sum_{t=1}^{T} h_t(\mathbf{x}))$$

where each $h_t(\mathbf{x})$ is a decision tree based classifier.

We often observe significantly improved classification performance using random forests over decision-tree.

Features of Random Forests

- It is unexcelled in accuracy among current algorithms.
- It runs efficiently on large data bases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what featuress are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.
- It has methods for balancing error in class population unbalanced data sets.