

Ensemble methods

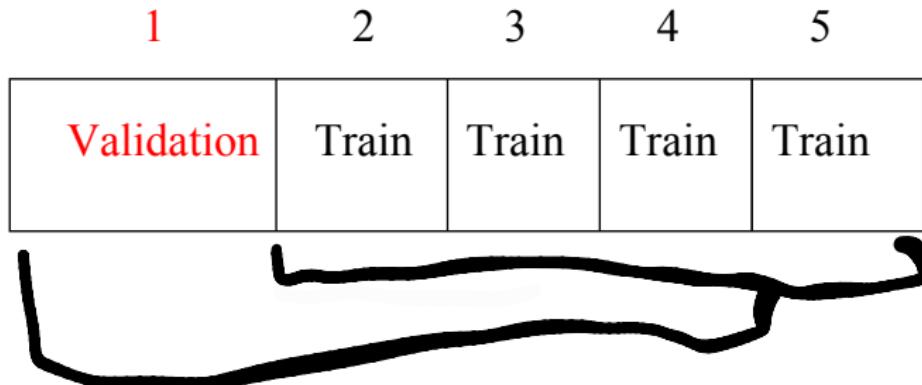
Resampling methods for validation

K -fold Cross-validation

- *Widely used approach* for estimating test error.
- Estimates can be used to select best model, and to give an idea of the test error of the final chosen model.
- Idea is to randomly divide the data into K equal-sized parts. We leave out part k , fit the model to the other $K - 1$ parts (combined), and then obtain predictions for the left-out k th part.
- This is done in turn for each part $k = 1, 2, \dots, K$, and then the results are combined.

K -fold Cross-validation in detail

Divide data into K roughly equal-sized parts ($K = 5$ here)



The details

- Let the K parts be C_1, C_2, \dots, C_K , where C_k denotes the indices of the observations in part k . There are n_k observations in part k : if N is a multiple of K , then $n_k = n/K$.
- Compute

$$\text{CV}_{(K)} = \sum_{k=1}^K \frac{n_k}{n} \text{MSE}_k$$

where $\text{MSE}_k = \sum_{i \in C_k} (y_i - \hat{y}_i)^2 / n_k$, and \hat{y}_i is the fit for observation i , obtained from the data with part k removed.

- Setting $K = n$ yields n -fold or *leave-one out cross-validation* (LOOCV).

The Bootstrap

- The *bootstrap* is a flexible and powerful statistical tool that can be used to quantify the uncertainty associated with a given estimator or statistical learning method.
- For example, it can provide an estimate of the standard error of a coefficient, or a confidence interval for that coefficient.

A simple example

- Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns of X and Y , respectively, where X and Y are random quantities.
- We will invest a fraction α of our money in X , and will invest the remaining $1 - \alpha$ in Y .
- We wish to choose α to minimize the total risk, or variance, of our investment. In other words, we want to minimize $\text{Var}(\alpha X + (1 - \alpha)Y)$.
- One can show that the value that minimizes the risk is given by

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}},$$

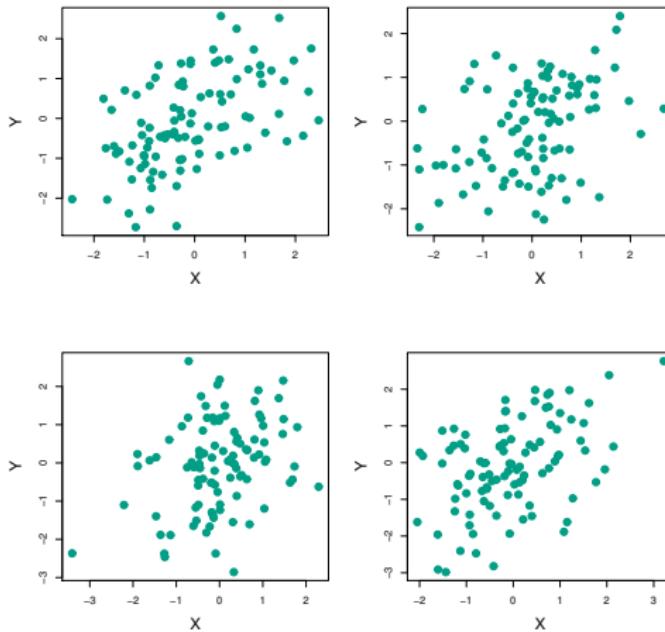
where $\sigma_X^2 = \text{Var}(X)$, $\sigma_Y^2 = \text{Var}(Y)$, and $\sigma_{XY} = \text{Cov}(X, Y)$.

Example continued

- But the values of σ_X^2 , σ_Y^2 , and σ_{XY} are unknown.
- We can compute estimates for these quantities, $\hat{\sigma}_X^2$, $\hat{\sigma}_Y^2$, and $\hat{\sigma}_{XY}$, using a data set that contains measurements for X and Y .
- We can then estimate the value of α that minimizes the variance of our investment using

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}.$$

Example continued



Each panel displays 100 simulated returns for investments X and Y. From left to right and top to bottom, the resulting estimates for α are 0.576, 0.532, 0.657, and 0.651.

Example continued

- To estimate the standard deviation of $\hat{\alpha}$, we repeated the process of simulating 100 paired observations of X and Y , and estimating α 1,000 times.
- We thereby obtained 1,000 estimates for α , which we can call $\hat{\alpha}_1, \hat{\alpha}_2, \dots, \hat{\alpha}_{1000}$.
- The left-hand panel of the Figure on slide 29 displays a histogram of the resulting estimates.
- For these simulations the parameters were set to $\sigma_X^2 = 1$, $\sigma_Y^2 = 1.25$, and $\sigma_{XY} = 0.5$, and so we know that the true value of α is 0.6 (indicated by the red line).

Example continued

- The mean over all 1,000 estimates for α is

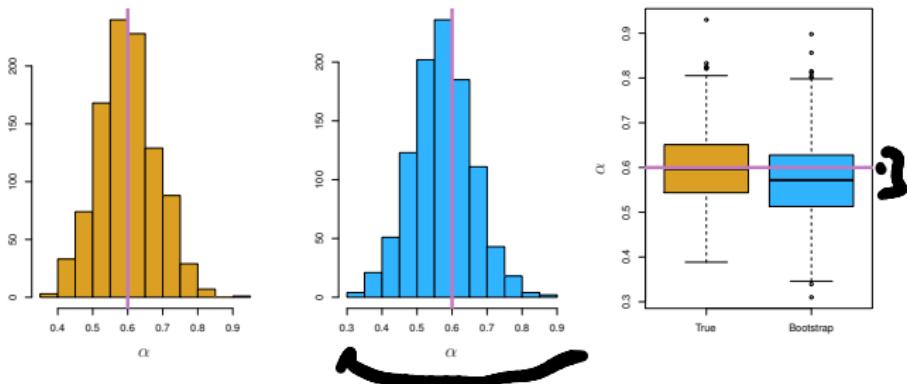
$$\bar{\alpha} = \frac{1}{1000} \sum_{r=1}^{1000} \hat{\alpha}_r = 0.5996,$$

very close to $\alpha = 0.6$, and the standard deviation of the estimates is

$$\sqrt{\frac{1}{1000 - 1} \sum_{r=1}^{1000} (\hat{\alpha}_r - \bar{\alpha})^2} = 0.083.$$

- This gives us a very good idea of the accuracy of $\hat{\alpha}$: $SE(\hat{\alpha}) \approx 0.083$.
- So roughly speaking, for a random sample from the population, we would expect $\hat{\alpha}$ to differ from α by approximately 0.08, on average.

Results

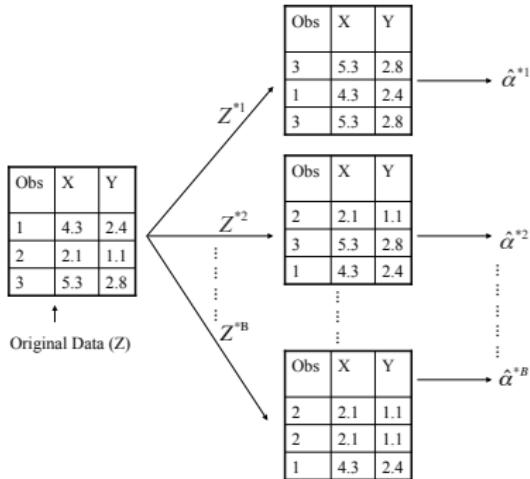


Left: A histogram of the estimates of α obtained by generating 1,000 simulated data sets from the true population. *Center:* A histogram of the estimates of α obtained from 1,000 bootstrap samples from a single data set. *Right:* The estimates of α displayed in the left and center panels are shown as boxplots. In each panel, the pink line indicates the true value of α .

Now back to the real world

- The procedure outlined above cannot be applied, because for real data we cannot generate new samples from the original population.
- However, the bootstrap approach allows us to use a computer to mimic the process of obtaining new data sets, so that we can estimate the variability of our estimate without generating additional samples.
- Rather than repeatedly obtaining independent data sets from the population, we instead obtain distinct data sets by repeatedly sampling observations from the original data set *with replacement*.
- Each of these “bootstrap data sets” is created by sampling *with replacement*, and is the *same size* as our original dataset. As a result some observations may appear more than once in a given bootstrap data set and some not at all.

Example with just 3 observations



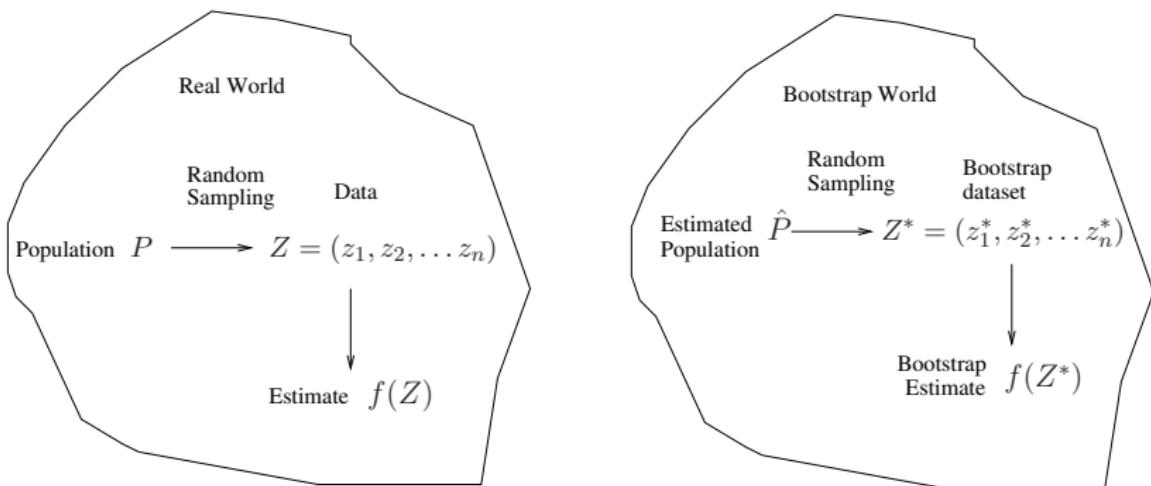
A graphical illustration of the bootstrap approach on a small sample containing $n = 3$ observations. Each bootstrap data set contains n observations, sampled with replacement from the original data set. Each bootstrap data set is used to obtain an estimate of α

- Denoting the first bootstrap data set by Z^{*1} , we use Z^{*1} to produce a new bootstrap estimate for α , which we call $\hat{\alpha}^{*1}$
- This procedure is repeated B times for some large value of B (say 100 or 1000), in order to produce B different bootstrap data sets, $Z^{*1}, Z^{*2}, \dots, Z^{*B}$, and B corresponding α estimates, $\hat{\alpha}^{*1}, \hat{\alpha}^{*2}, \dots, \hat{\alpha}^{*B}$.
- We estimate the standard error of these bootstrap estimates using the formula

$$\text{SE}_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1} \sum_{r=1}^B (\hat{\alpha}^{*r} - \bar{\hat{\alpha}}^*)^2}.$$

- This serves as an estimate of the standard error of $\hat{\alpha}$ estimated from the original data set. See center and right panels of Figure on slide 29. Bootstrap results are in blue. For this example $\text{SE}_B(\hat{\alpha}) = 0.087$.

A general picture for the bootstrap



The bootstrap in general

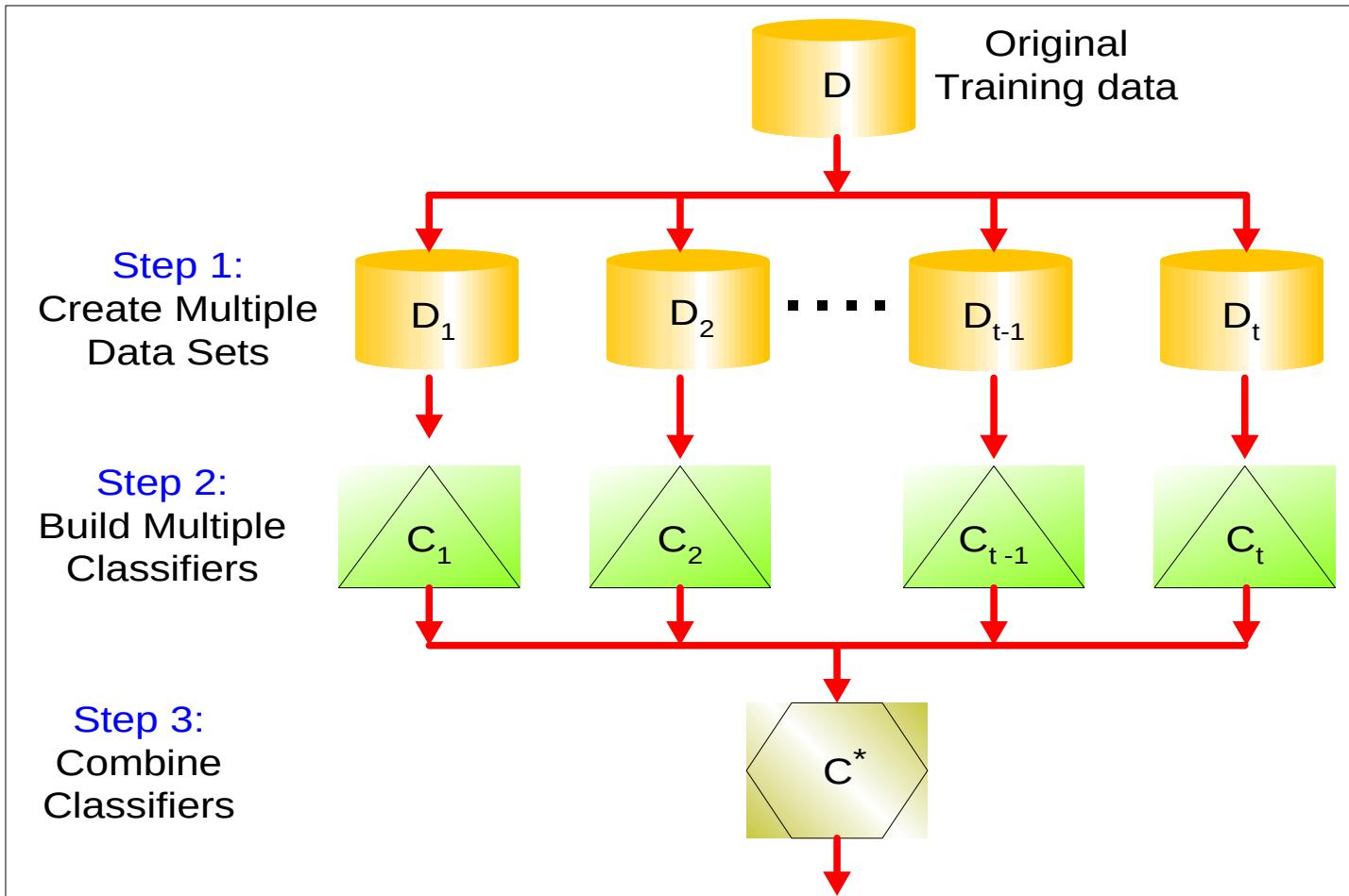
- In more complex data situations, figuring out the appropriate way to generate bootstrap samples can require some thought.
- For example, if the data is a time series, we can't simply sample the observations with replacement (*why not?*).
- We can instead create blocks of consecutive observations, and sample those with replacements. Then we paste together sampled blocks to obtain a bootstrap dataset.

Other uses of the bootstrap

- Primarily used to obtain standard errors of an estimate.
- Also provides approximate confidence intervals for a population parameter. For example, looking at the histogram in the middle panel of the Figure on slide 29, the 5% and 95% quantiles of the 1000 values is (.43, .72).
- This represents an approximate 90% confidence interval for the true α . *How do we interpret this confidence interval?*
- The above interval is called a *Bootstrap Percentile* confidence interval. It is the simplest method (among many approaches) for obtaining a confidence interval from the bootstrap.

Resampling methods for learning

General Idea



Why does it work?

- Suppose there are 25 base classifiers
 - Each classifier has error rate, $\varepsilon = 0.35$
 - Assume classifiers are independent
 - Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

Some ensemble methods for Machine Learning

- How to generate an ensemble of classifiers?
 1. Bagging (**Bootstrap aggregation**)
 2. Boosting

Bagging

- Sampling with replacement

| Data ID | Training Data | | | | | | | | | |
|-------------------|---------------|---|----|----|---|---|----|----|---|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Original Data | | | | | | | | | | |
| Bagging (Round 1) | 7 | 8 | 10 | 8 | 2 | 5 | 10 | 10 | 5 | 9 |
| Bagging (Round 2) | 1 | 4 | 9 | 1 | 2 | 3 | 2 | 7 | 3 | 2 |
| Bagging (Round 3) | 1 | 8 | 5 | 10 | 5 | 5 | 9 | 6 | 3 | 7 |

- Build classifier on each bootstrap sample
- Each sample has probability $\underbrace{(1 - 1/n)^n}$ of being selected as test data
- Training data = $\underbrace{1 - (1 - 1/n)^n}$ of the original data

The 0.632 bootstrap

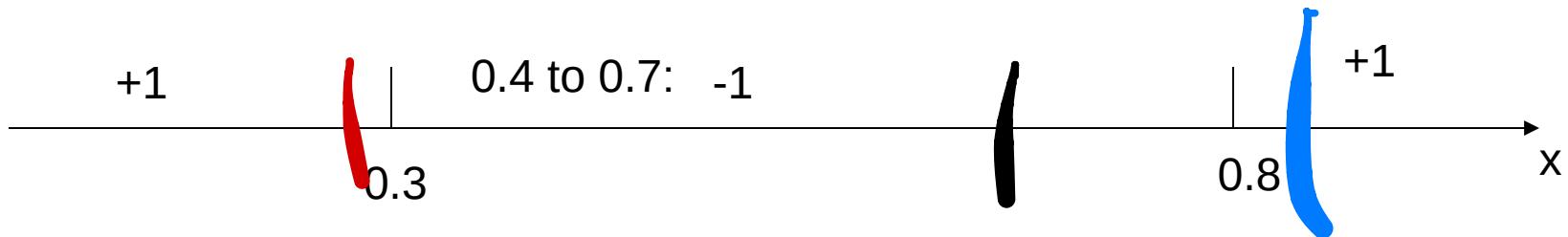
- This method is also called the *0.632 bootstrap*
 - A particular training data has a probability of $1 - 1/n$ of *not* being picked
 - Thus its probability of ending up in the test data (not selected) is:

$$\left(1 - \frac{1}{n}\right)^n \approx e^{-1} = 0.368$$

- This means the training data will contain approximately 63.2% of the instances

Example of Bagging

Assume that the training data is:



Goal: find a collection of 10 simple thresholding classifiers that collectively can classify correctly.

-Each simple (or weak) classifier is:

($x \leq K \rightarrow \text{class} = +1$ or -1 depending on
which value yields the lowest error; where K
is determined by entropy minimization)

Bagging Round 1:

| | | | | | | | | | | |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| x | 0.1 | 0.2 | 0.2 | 0.3 | 0.4 | 0.4 | 0.5 | 0.6 | 0.9 | 0.9 |
| y | 1 | 1 | 1 | 1 | -1 | -1 | -1 | -1 | 1 | 1 |

$$x \leq 0.35 \Rightarrow y = 1$$

$$x > 0.35 \Rightarrow y = -1$$

.

Bagging Round 2:

| | | | | | | | | | | |
|---|-----|-----|-----|-----|-----|-----|-----|---|---|---|
| x | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.8 | 0.9 | 1 | 1 | 1 |
| y | 1 | 1 | 1 | -1 | -1 | 1 | 1 | 1 | 1 | 1 |

$$x \leq 0.65 \Rightarrow y = 1$$

$$x > 0.65 \Rightarrow y = 1$$

.

Bagging Round 3:

| | | | | | | | | | | |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| x | 0.1 | 0.2 | 0.3 | 0.4 | 0.4 | 0.5 | 0.7 | 0.7 | 0.8 | 0.9 |
| y | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 |

$$x \leq 0.35 \Rightarrow y = 1$$

$$x > 0.35 \Rightarrow y = -1$$

.

Bagging Round 4:

| | | | | | | | | | | |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| x | 0.1 | 0.1 | 0.2 | 0.4 | 0.4 | 0.5 | 0.5 | 0.7 | 0.8 | 0.9 |
| y | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 |

$$x \leq 0.3 \Rightarrow y = 1$$

$$x > 0.3 \Rightarrow y = -1$$

Bagging Round 5:

| | | | | | | | | | | |
|---|-----|-----|-----|-----|-----|-----|-----|---|---|---|
| x | 0.1 | 0.1 | 0.2 | 0.5 | 0.6 | 0.6 | 0.6 | 1 | 1 | 1 |
| y | 1 | 1 | 1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 |

$$x \leq 0.35 \Rightarrow y = 1$$

$$x > 0.35 \Rightarrow y = -1$$

Bagging Round 6:

| | | | | | | | | | | |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|---|
| x | 0.2 | 0.4 | 0.5 | 0.6 | 0.7 | 0.7 | 0.7 | 0.8 | 0.9 | 1 |
| y | 1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 |

$$x \leq 0.75 \Rightarrow y = -1$$

$$x > 0.75 \Rightarrow y = 1$$

Bagging Round 7:

| | | | | | | | | | | |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|---|
| x | 0.1 | 0.4 | 0.4 | 0.6 | 0.7 | 0.8 | 0.9 | 0.9 | 0.9 | 1 |
| y | 1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 | 1 | 1 |

$$x \leq 0.75 \Rightarrow y = -1$$

$$x > 0.75 \Rightarrow y = 1$$

Bagging Round 8:

| | | | | | | | | | | |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|---|
| x | 0.1 | 0.2 | 0.5 | 0.5 | 0.5 | 0.7 | 0.7 | 0.8 | 0.9 | 1 |
| y | 1 | 1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 |

$$x \leq 0.75 \Rightarrow y = -1$$

$$x > 0.75 \Rightarrow y = 1$$

Bagging Round 9:

| | | | | | | | | | | |
|---|-----|-----|-----|-----|-----|-----|-----|-----|---|---|
| x | 0.1 | 0.3 | 0.4 | 0.4 | 0.6 | 0.7 | 0.7 | 0.8 | 1 | 1 |
| y | 1 | 1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 |

$$x \leq 0.75 \Rightarrow y = -1$$

$$x > 0.75 \Rightarrow y = 1$$

Bagging Round 10:

| | | | | | | | | | | |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| x | 0.1 | 0.1 | 0.1 | 0.1 | 0.3 | 0.3 | 0.8 | 0.8 | 0.9 | 0.9 |
| y | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

$$x \leq 0.05 \Rightarrow y = -1$$

$$x > 0.05 \Rightarrow y = 1$$

Figure 5.35. Example of bagging.

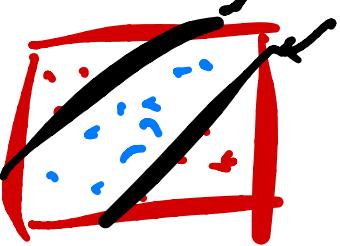
Bagging (applied to training data)

| Round | x=0.1 | x=0.2 | x=0.3 | x=0.4 | x=0.5 | x=0.6 | x=0.7 | x=0.8 | x=0.9 | x=1.0 |
|------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 1 | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 3 | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| 4 | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| 5 | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| 6 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 |
| 7 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 |
| 8 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 |
| 9 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Sum | 2 | 2 | 2 | -6 | -6 | -6 | -6 | 2 | 2 | 2 |
| Sign | 1 | 1 | 1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 |
| True Class | 1 | 1 | 1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 |

Figure 5.36. Example of combining classifiers constructed using the bagging approach.

Accuracy of ensemble classifier: 100% ☺

Bagging- Summary



- Works well if the base classifiers are unstable (complement each other)
- Increased accuracy because it ***reduces the variance*** of the individual classifier
- Does not focus on any particular instance of the training data
 - Therefore, less susceptible to model overfitting when applied to noisy data
- What if we want to focus on a particular instances of training data?

Bagging- Summary

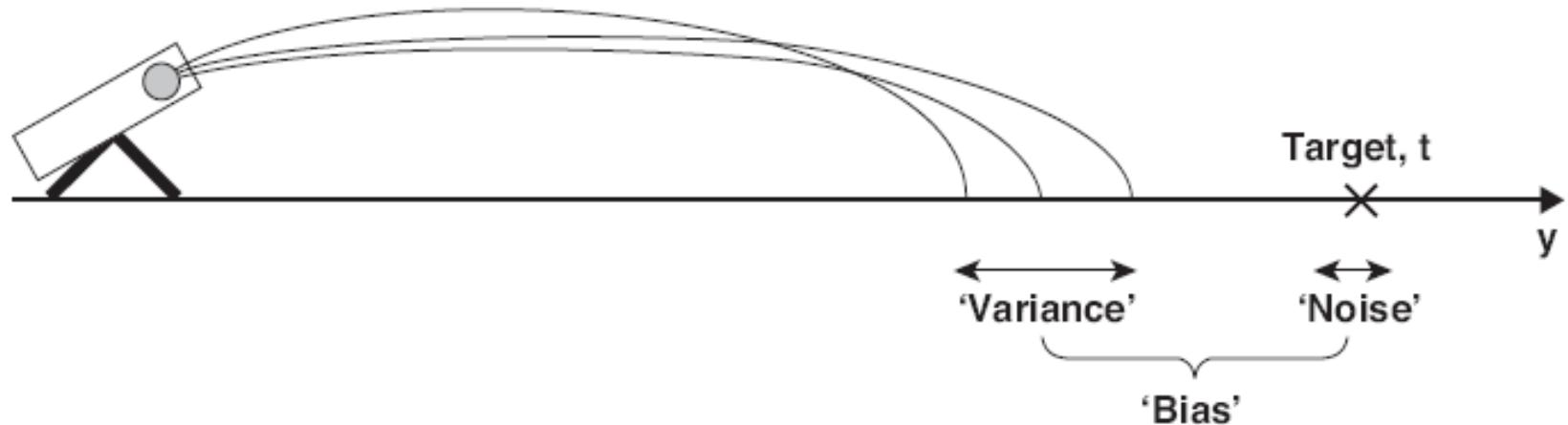
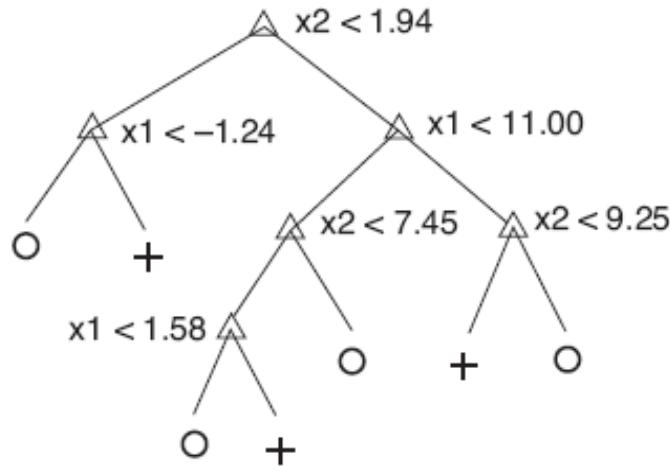


Figure 5.32. Bias-variance decomposition.

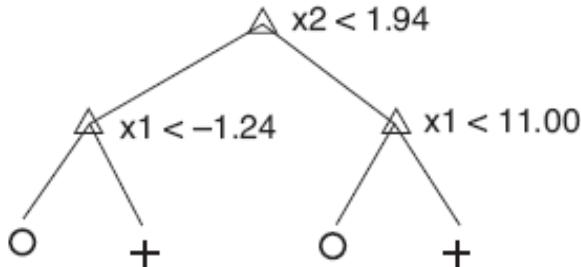
In general,

- **Bias** is contributed to by the training error; a complex model has low bias.
- **Variance** is caused by future error; a complex model has High variance.
- Bagging reduces the variance in the base classifiers.

Bagging- Summary



(a) Decision tree T_1



(b) Decision tree T_2

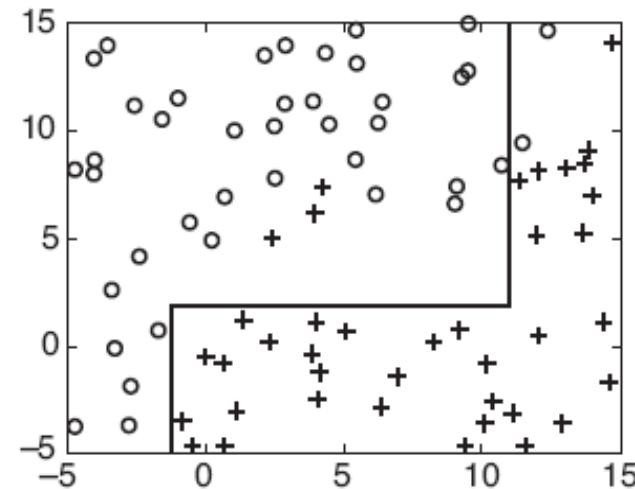
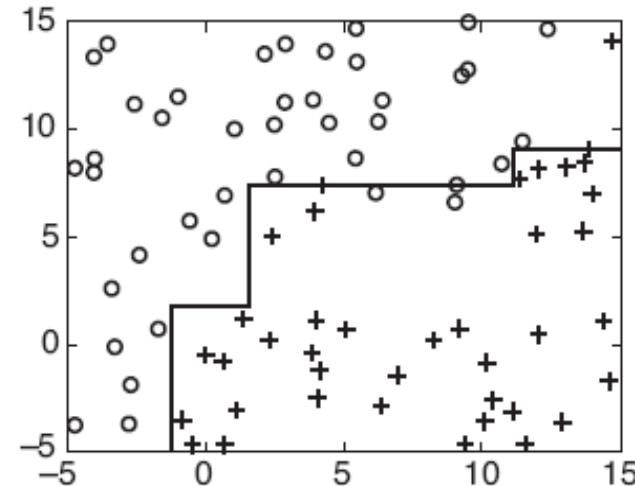
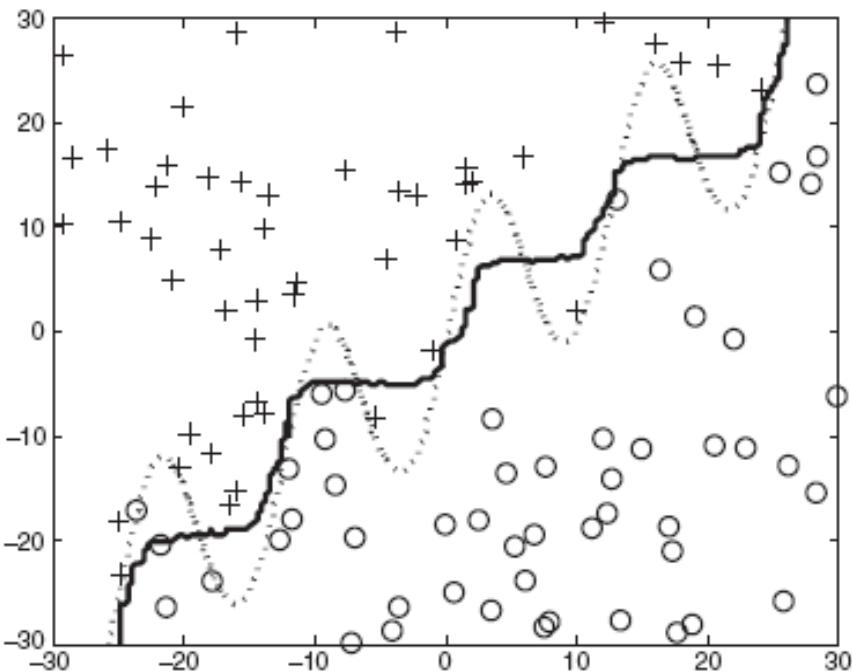
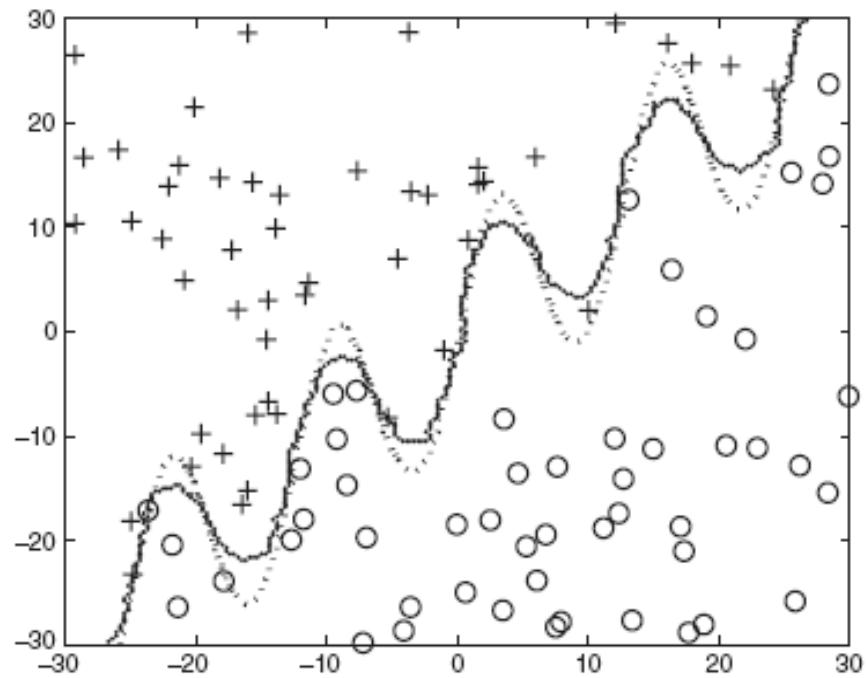


Figure 5.33. Two decision trees with different complexities induced from the same training data.

Bagging- Summary



(a) Decision boundary for decision tree.



(b) Decision boundary for 1-nearest neighbor.

Figure 5.34. Bias of decision tree and 1-nearest neighbor classifiers.

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
 - Initially, all N records are assigned equal weights
 - Unlike bagging, weights may change at the end of a boosting round

Boosting

- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

| Original Data | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|--------------------|---|---|---|----|---|---|---|----|---|----|
| Boosting (Round 1) | 7 | 3 | 2 | 8 | 7 | 9 | 4 | 10 | 6 | 3 |
| Boosting (Round 2) | 5 | 4 | 9 | 4 | 2 | 5 | 1 | 7 | 4 | 2 |
| Boosting (Round 3) | 4 | 4 | 8 | 10 | 4 | 5 | 4 | 6 | 3 | 4 |

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

Boosting

- Equal weights are assigned to each training instance ($1/N$ for round 1) at first round
- After a classifier C_i is learned, the weights are adjusted to allow the subsequent classifier
- C_{i+1} to “pay more attention” to data that were misclassified by C_i .
- Final boosted classifier C^* combines the votes of each individual classifier
 - Weight of each classifier’s vote is a function of its accuracy
- Adaboost – popular boosting algorithm

Adaboost (Adaptive Boost)

- Input:
 - Training set D containing N instances
 - T rounds
 - A classification learning scheme
- Output:
 - A composite model

Adaboost: Training Phase

- Training data D contain N labeled data (X_1, y_1) , (X_2, y_2) , (X_3, y_3) , ..., (X_N, y_N)
- Initially assign equal weight $\frac{1}{d}$ to each data
- To generate T base classifiers, we need T rounds or iterations
- Round i, data from D are sampled with replacement , to form D_i (size N)
- Each data's chance of being selected in the next rounds depends on its weight
 - Each time the new sample is generated directly from the training data D with different sampling probability according to the weights; these weights are not zero

Adaboost: Training Phase

- Base classifier C_i , is derived from training data of D_i
- Error of C_i is tested using D_i
- Weights of training data are adjusted depending on how they were classified
 - Correctly classified: Decrease weight
 - Incorrectly classified: Increase weight
- Weight of a data indicates how hard it is to classify it (directly proportional)

Adaboost: Testing Phase

- The lower a classifier error rate, the more accurate it is, and therefore, the higher its weight for voting should be
- Weight of a classifier C_i 's vote is
- Testing:
 - For each class c , sum the weights of each classifier that assigned class c to X (unseen data)
 - The class with the highest sum is the WINNER!

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_i}{\varepsilon_i} \right)$$

$$C^*(x_{test}) = \operatorname{argmax}_y \sum_{i=1}^T \alpha_i \delta(C_i(x_{test}) = y)$$

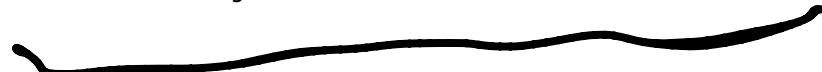
Example: Error and Classifier Weight in AdaBoost

- Base classifiers: C_1, C_2, \dots, C_T



- Error rate: (i = index of classifier, j =index of instance)

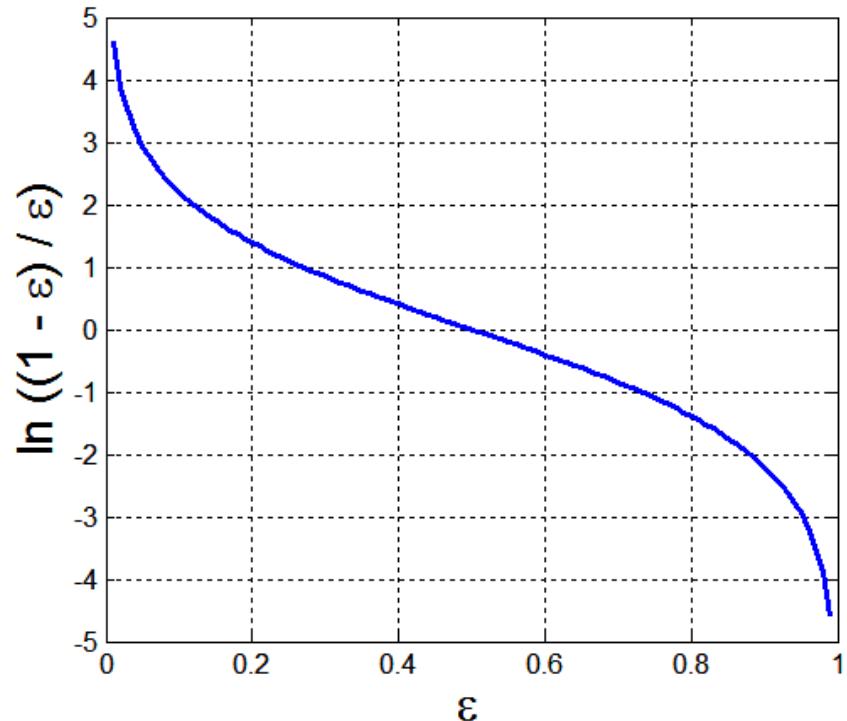
$$\varepsilon_i = \frac{1}{N} \sum_{j=1}^N w_j \delta(C_i(x_j) \neq y_j)$$



- Importance of a classifier:

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_i}{\varepsilon_i} \right)$$





Example: Data Instance Weight in AdaBoost

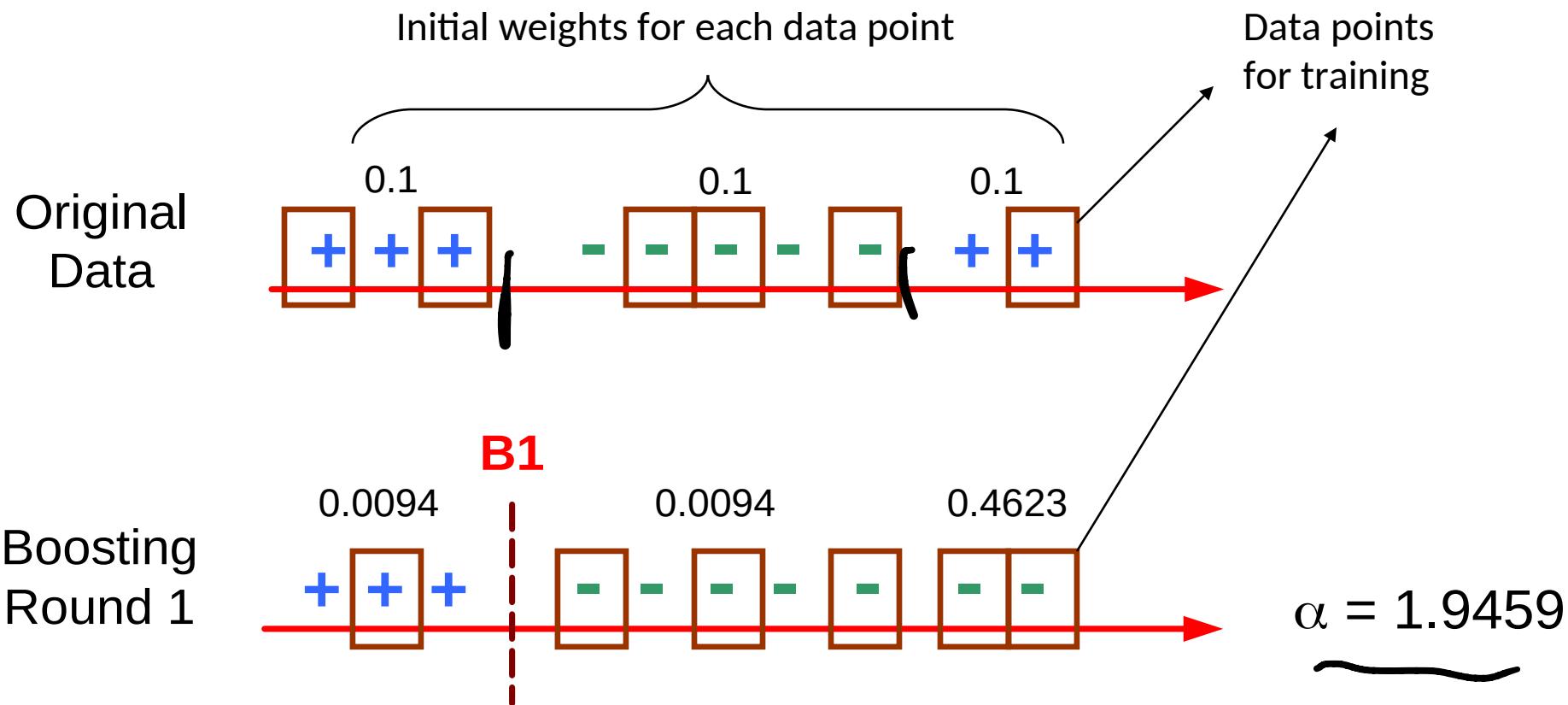
- Assume: N training data in D , T rounds, (x_j, y_j) are the training data, C_i , a_i are the classifier and weight of the i^{th} round, respectively.
- Weight update on all training data in D :

$$w_j^{(i+1)} = \frac{w_j^{(i)}}{Z_i} \begin{cases} \exp^{-\alpha_i} & \text{if } C_i(x_j) = y_j \\ \exp^{\alpha_i} & \text{if } C_i(x_j) \neq y_j \end{cases}$$

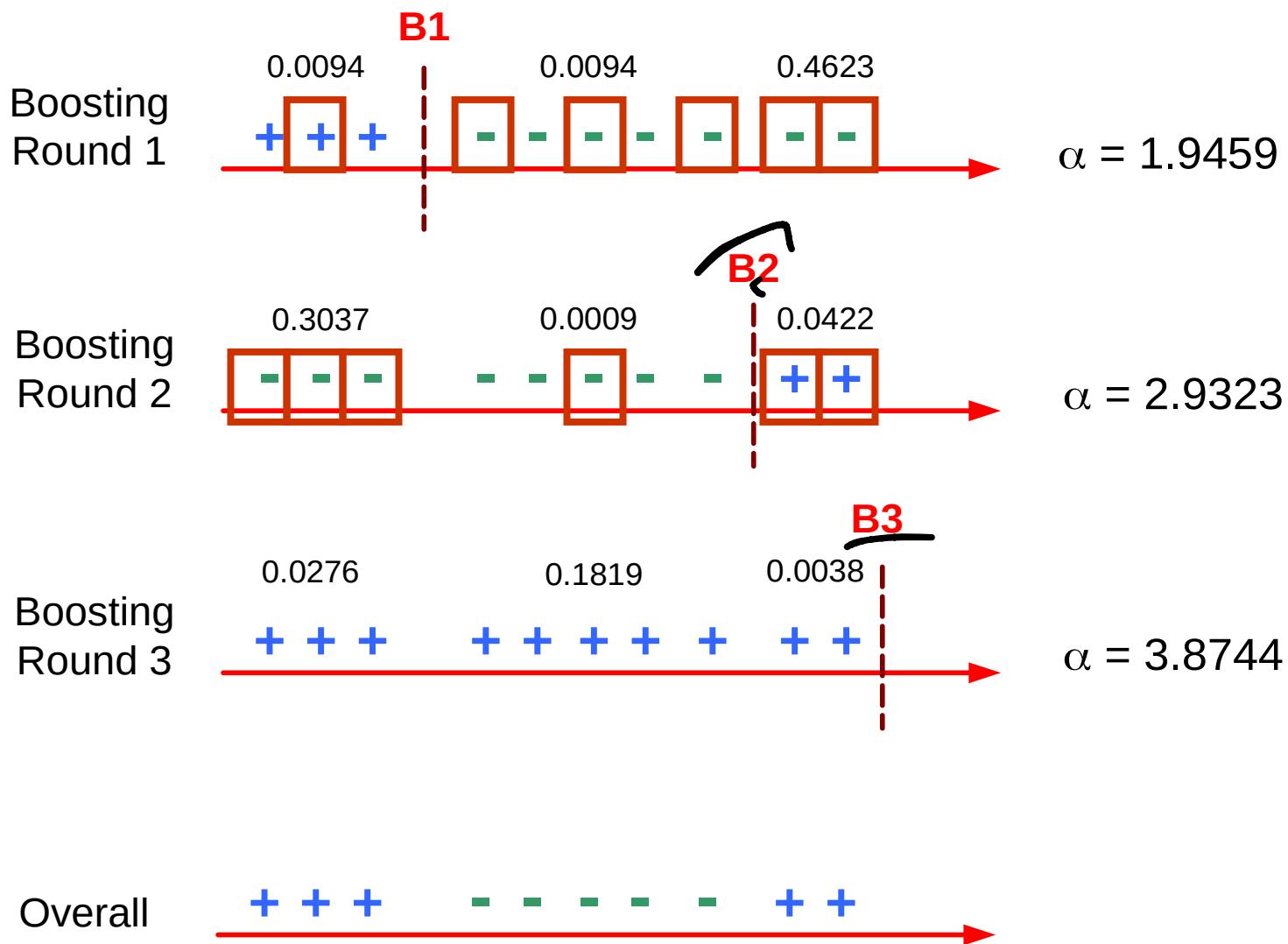
where Z_i is the normalization factor

$$C*(x_{test}) = \operatorname{argmax}_y \sum_{i=1}^T \alpha_i \delta(C_i(x_{test}) = y)$$

Illustrating AdaBoost



Illustrating AdaBoost



Random Forests

- Ensemble method specifically designed for decision tree classifiers
- Random Forests grows many trees
 - Ensemble of unpruned decision trees
 - Each base classifier classifies a “new” vector of attributes from the original data
 - Final result on classifying a new instance: voting. Forest chooses the classification result having the most votes (over all the trees in the forest)

Random Forests

- Introduce two sources of randomness:
“Bagging” and “Random input vectors”
 - Bagging method: each tree is grown using a bootstrap sample of training data
 - Random vector method: **At each node**, best split is chosen from a random sample of m attributes instead of all attributes

Random Forests

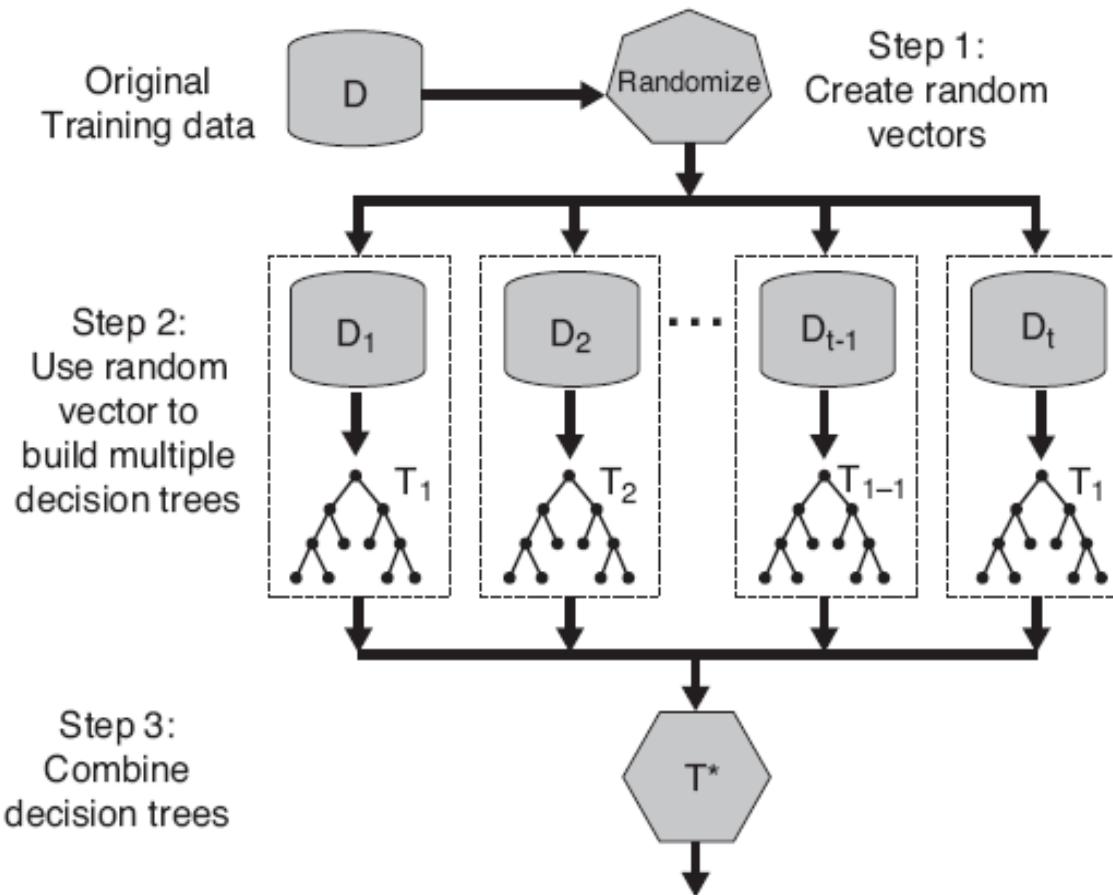


Figure 5.40. Random forests.

Methods for Growing the Trees

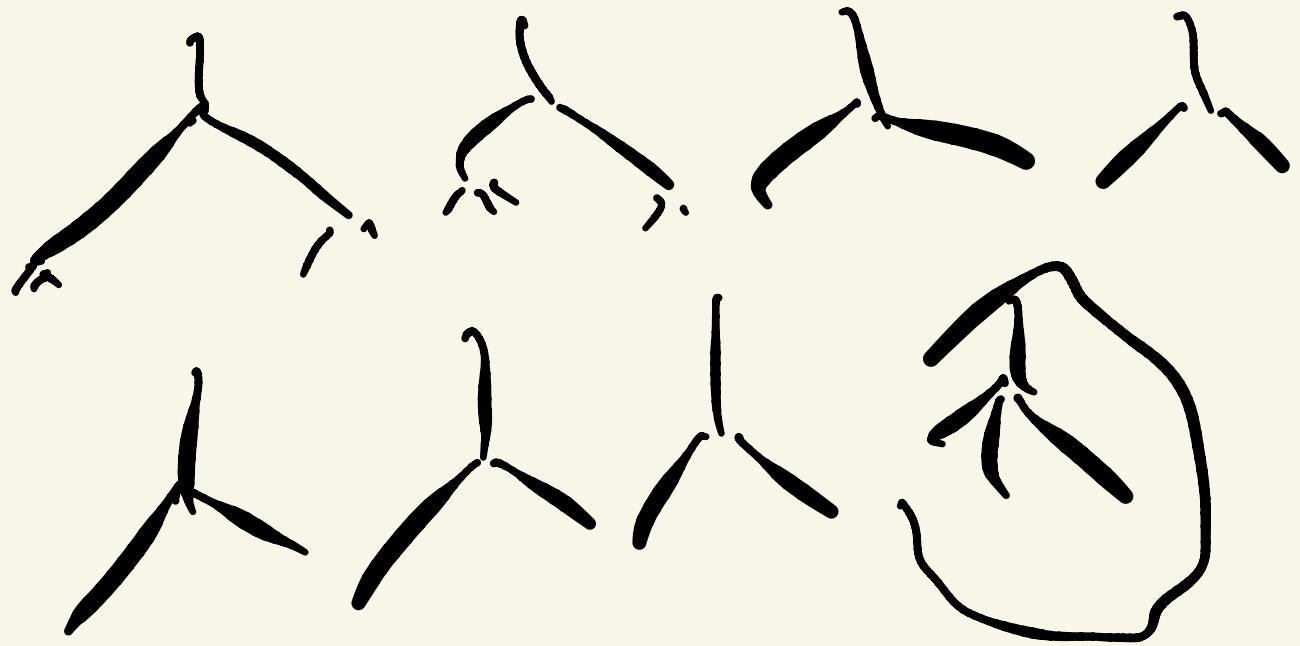
- Fix a $m \leq M$. At each node
 - Method 1:
 - Choose m attributes randomly, compute their information gains, and choose the attribute with the largest gain to split
 - Method 2:
 - (When M is not very large): select L of the attributes randomly. Compute a linear combination of the L attributes using weights generated from $[-1, +1]$ randomly. That is, new $A = \sum(W_i * A_i)$, $i=1..L$.
 - Method 3:
 - Compute the information gain of all M attributes. Select the top m attributes by information gain. Randomly select one of the m attributes as the splitting node.

Random Forest Algorithm: method 1 in previous slide

- M input features in training data, a number $m \ll M$ is specified such that **at each node**, m features are selected at random out of the M and the best split on these m features is used to split the node. (In weather data, M=4, and m is between 1 and 4)
- m is held constant during the forest growing
- Each tree is grown to the largest extent possible (deep tree, overfit easily), and there is no pruning

Generalization Error of Random Forests (page 291 of Tan book)

- It can be proven that the generalization Error $\leq \rho(1-s^2)/s^2, \left(1 - \frac{1}{s^2}\right)$
 - ρ is the average correlation among the trees
 - s is the strength of the tree classifiers
 - Strength is defined as *how certain* the classification results are on the training data on average
 - How certain is measured $\Pr(C1|X) - \Pr(C2|X)$, where $C1, C2$ are class values of two highest probability in decreasing order for input instance X .
- Thus, higher diversity and accuracy is good for performance



Summary:

- Bootstrap: Accurate standard deviation from resampling
- Bagging: Bootstrap with aggregation, use of resample for prediction
- Boosting: From weak to strong predictors
 - Adaboost algorithm
- Random forest algorithm