1

Machine Learning

Ensemble learning

林嘉文 (Chia-Wen Lin)
清華大學電機系
cwlin@ee.nthu.edu.tw

6/6/2022

No Free Lunch Theorem

No Free Lunch Theorem

- No Free Lunch Theorem: There is **NO** overall superior or inferior classification, if no prior assumptions about the problem are made
- Generate a group of base-learners which when combined has higher accuracy
- Different learners use different
  - Algorithms
  - Hyperparameters
  - Representations /Modalities/Views
  - Training sets
  - Subproblems
- Diversity vs accuracy

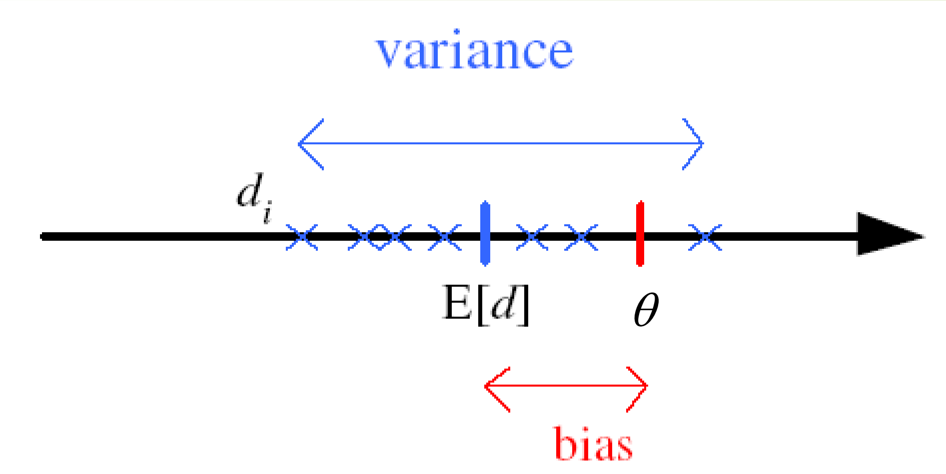
Bias & Variance

Unknown function f
Estimator $g_i = g(X_i)$ on sample X_i

Bias: $b_f(g) = \mathbb{E}[g] - f$
Variance: $\mathbb{E}[(g - \mathbb{E}[g])^2]$

Mean square error:

$$r(g, f) = \mathbb{E}[(g - f)^2]$$
$$= (\mathbb{E}[g] - f)^2 + \mathbb{E}[(g - \mathbb{E}[g])^2]$$
$$= Bias^2 + Variance$$

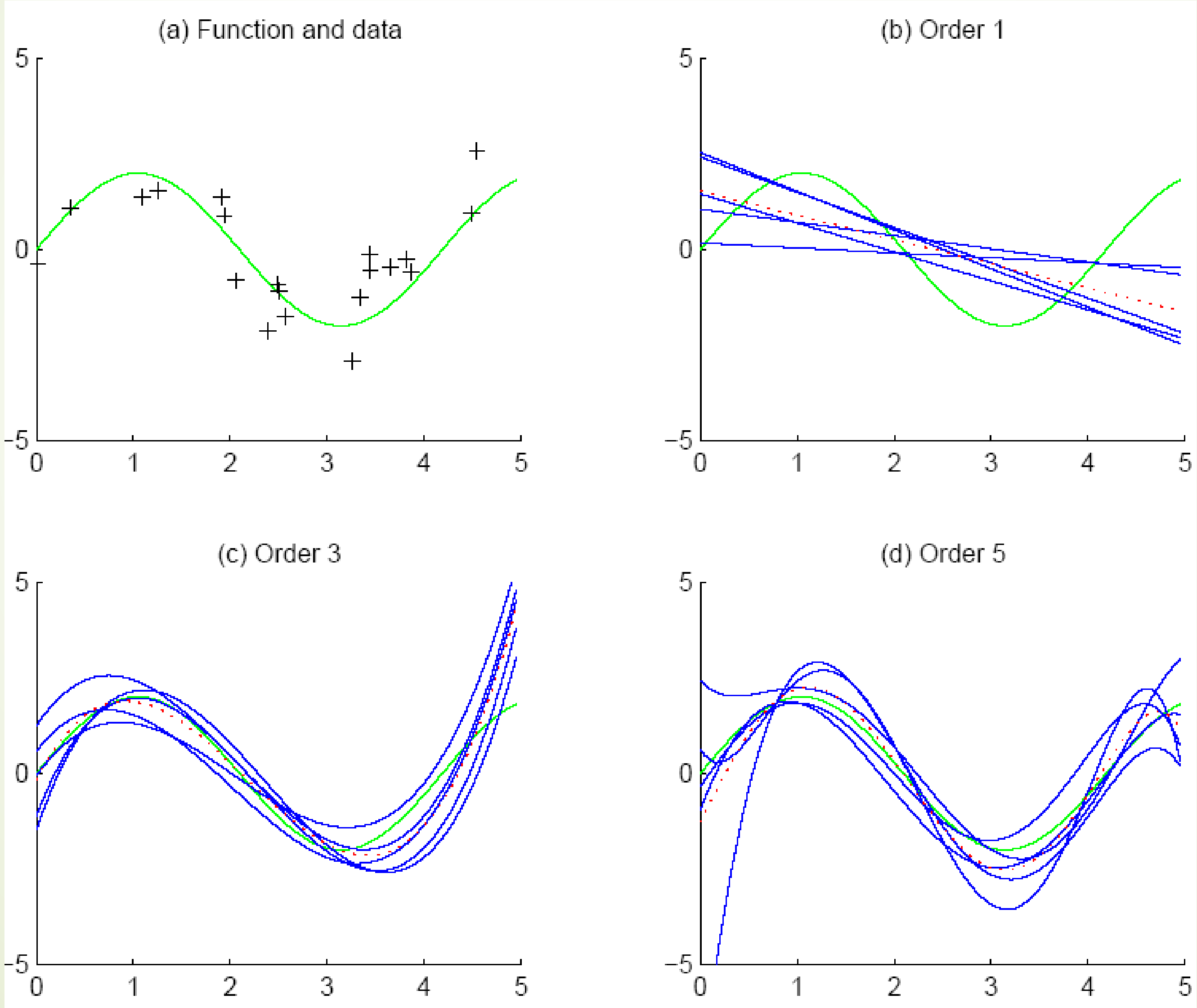


Bias/Variance Dilemma

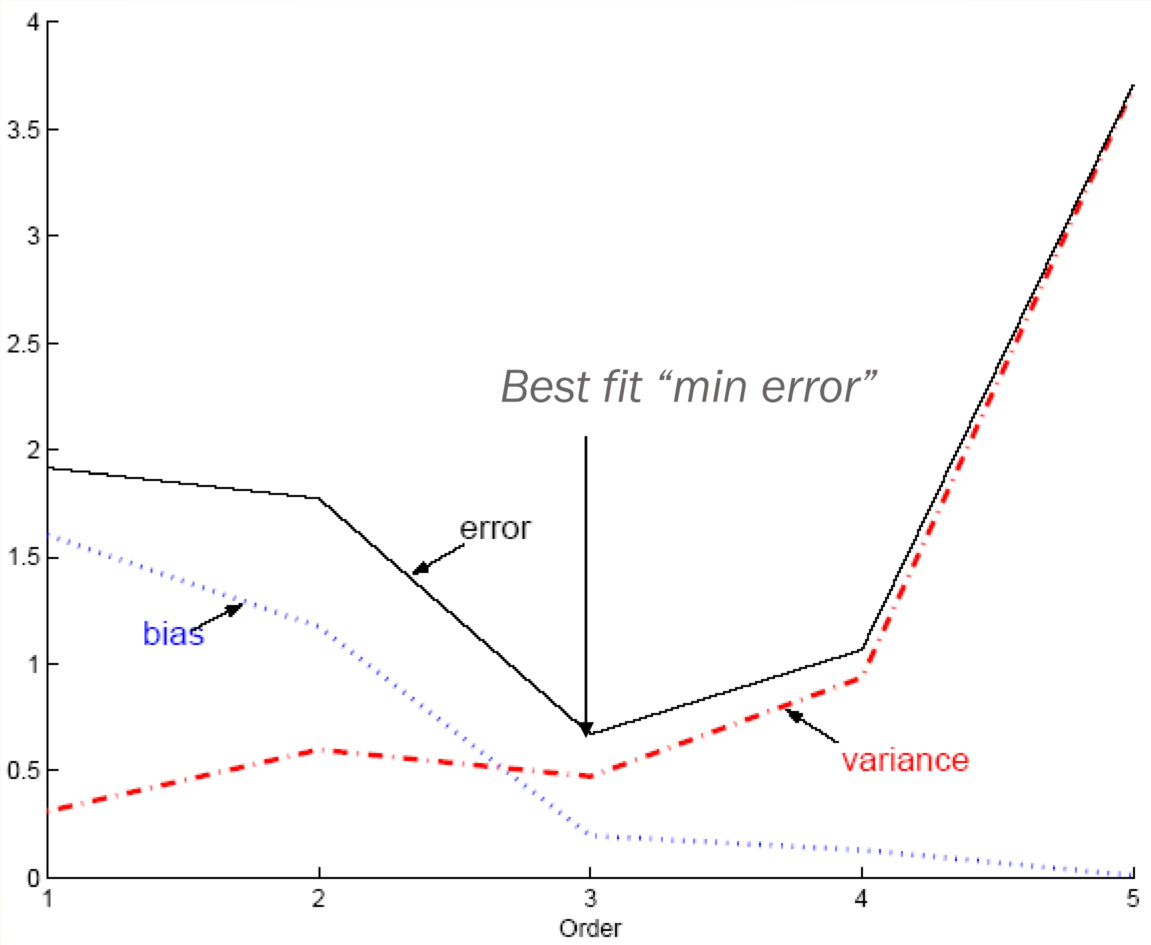
$$\mathbb{E}_D[(g(x,D)-f(x))^2]$$
$$=\underbrace{(\mathbb{E}_D[g(x,D)-f(x)])^2}_{\text{bias}^2}+\underbrace{\left(\mathbb{E}_D[(g(x,D)-\mathbb{E}_D[g(x,D)])^2]\right)}_{\text{variance}}$$

- Bias: the difference between the expected value and the true value
- Variance: variations of estimated values
- Given training set D, as we increase complexity,
  - bias decreases (a better fit to data) and variance increases (fit varies more with data)
- High bias means usually low variance, and vice versa
- Bias/Variance dilemma: (Geman et al., 1992)

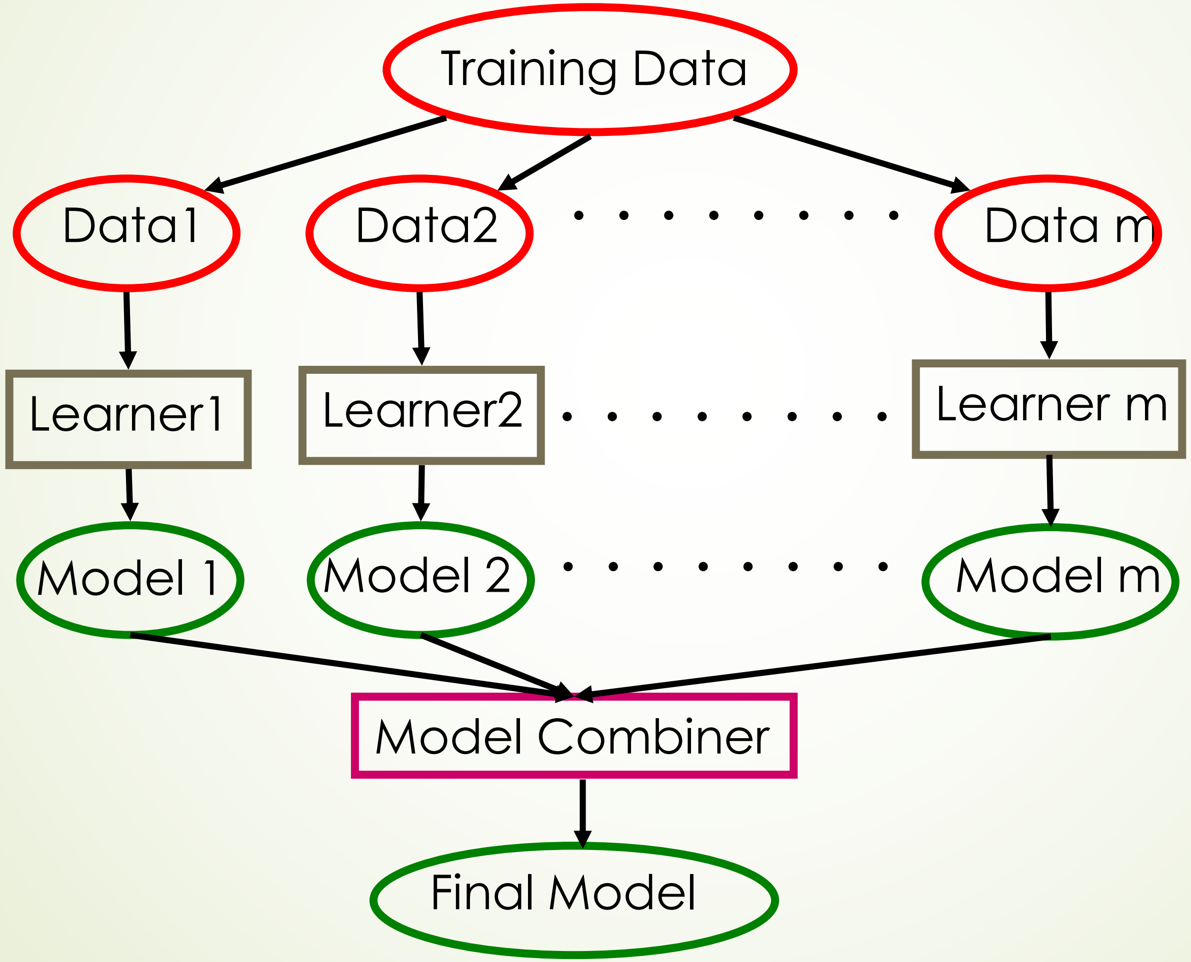
Bias/Variance Dilemma



Bias/Variance Dilemma



Ensemble Learning



Ensemble Learning: Intuition

- **Majority Vote**
- Suppose we have 5 completely independent classifiers each having 70% accuracy
  - $10 \cdot 0.7^3 \cdot 0.3^2 + 5 \cdot 0.7^4 \cdot 0.3^1 + 0.7^5 = 83.7\%$ majority accuracy
- 101 such classifiers
  - 99.9% majority vote accuracy

Ensemble Learning

- **Bagging** (Breiman 1994,…): Fit several classifiers to bootstrap resampled versions of the training data, and classify by majority vote.
- **Boosting** (Freund and Schapire 1995, Friedman et al. 1998,…): Fit many large or small classifiers to reweighted versions of the training data, then classify by weighted majority vote
- **Random forests** (Breiman 2001,…): Fancier version of bagging

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

Voting of Classifiers

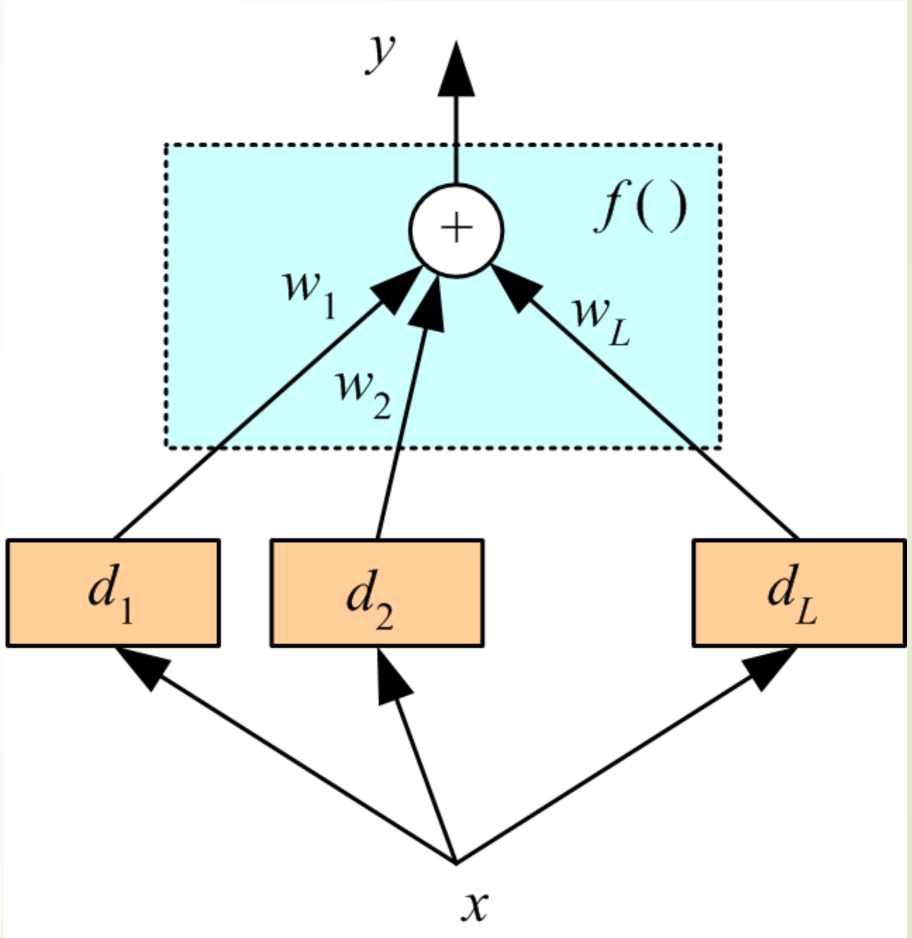
- Linear combination

$$y = \sum_{j=1}^{L} w_j d_j$$

$$w_j \geq 0 \ \text{and} \ \sum_{j=1}^{L} w_j = 1$$

- Classification

$$y_i = \sum_{j=1}^{L} w_j d_{ji}$$



Voting of Classifiers

Rule	Fusion function $f(\cdot)$
Sum	$y_i = \frac{1}{L}\sum_{j=1}^{L} d_{ji}$
Weighted sum	$y_i = \sum_j w_j d_{ji}, w_j \geq 0, \sum_j w_j = 1$
Median	$y_i = \text{median}_j d_{ji}$
Minimum	$y_i = \min_j d_{ji}$
Maximum	$y_i = \max_j d_{ji}$
Product	$y_i = \prod_j d_{ji}$

	C_1	C_2	C_3
d_1	0.2	0.5	0.3
d_2	0.0	0.6	0.4
d_3	0.4	0.4	0.2
Sum	0.2	**0.5**	0.3
Median	0.2	**0.5**	0.4
Minimum	0.0	**0.4**	0.2
Maximum	0.4	**0.6**	0.4
Product	0.0	**0.12**	0.032

Voting of Classifiers

- Bayesian perspective:

$$P(C_i|x) = \sum_{\text{all models } \mathrm{M}_j} P(C_i|x, \mathrm{M}_j)\ P(\mathrm{M}_j)$$

If d_j are iid

$$E[y] = E\left[\sum_j \frac{1}{L} d_j\right] = \frac{1}{L} L \cdot E[d_j] = E[d]$$

$$\mathrm{Var}(y) = \mathrm{Var}\left(\sum_j \frac{1}{L} d_j\right) = \frac{1}{L^2} \mathrm{Var}\left(\sum_j d_j\right) = \frac{1}{L^2} L \cdot \mathrm{Var}(d_j) = \frac{1}{L} \mathrm{Var}(d)$$

Bias does not change, variance decreases to $1/L$

- If dependent, error increases with positive correlation

$$\mathrm{Var}(y) = \frac{1}{L^2} \mathrm{Var}\left(\sum_j d_j\right) = \frac{1}{L^2}\left[\sum_j \mathrm{Var}(d_j) + 2 \sum_j \sum_{i<j} Cov(d_i, d_j)\right]$$

Resampling for Classifier Design: Jackknife

- Remove some point from the training set: $D_{(i)}$
- Calculate the leave-one-out statistics with the new training set

$$\mu_{(i)} = \frac{1}{n-1}\sum_{j \neq i} x_j$$

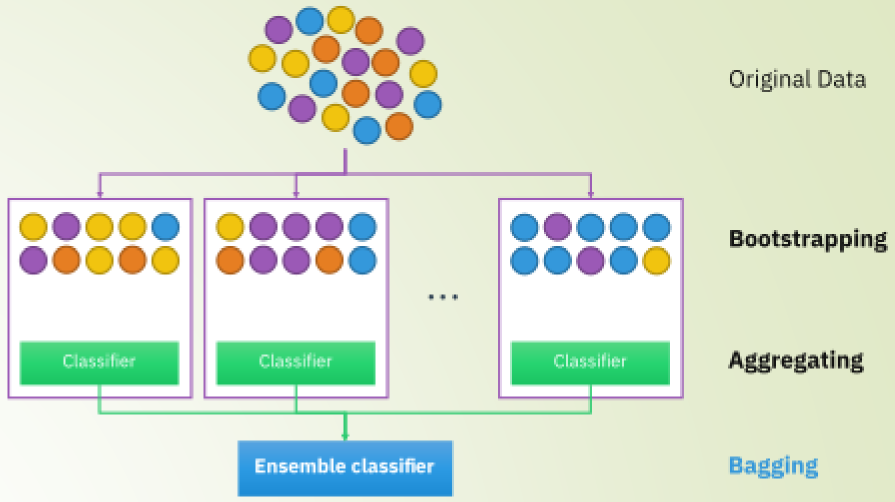
- Repeat for all points
- Calculate the jackknife (leave-one-out) statistics

$$\mu_{(\cdot)} = \frac{1}{n}\sum_{i=1}^{n} \mu_{(i)}$$

Resampling for Classifier Design

- Arcing
  - **A**daptive **r**eweighting and **c**ombining
  - Reuse or select data in order to improve classification
  - **Bootstrap** data set
    - Randomly select n points from the training set D with replacement

Bagging



- Bootstrap AGGregation
  - Use multiple versions of a training set by drawing $n < N$ samples from D with replacement.
  - Each of the bootstrap data sets is used to train a different component classifier
  - The final classification decision is based on the votes of the component classifiers

Bagging

- Instability
  - A classifier/learning algorithm is called “unstable” if “small” changes in the training data lead to “large” changes in accuracy
  - In general, bagging improves recognition for unstable classifiers since it effectively averages over such discontinuities

Boosting

- Overview of boosting
  - Improve the accuracy of any given learning algorithm
  - First create a (**weak**) classifier with accuracy on the training set greater than average
  - Add new component classifiers to form an ensemble whose joint decision rule has arbitrarily high accuracy on the training set
  - The classification performance has been “boosted”

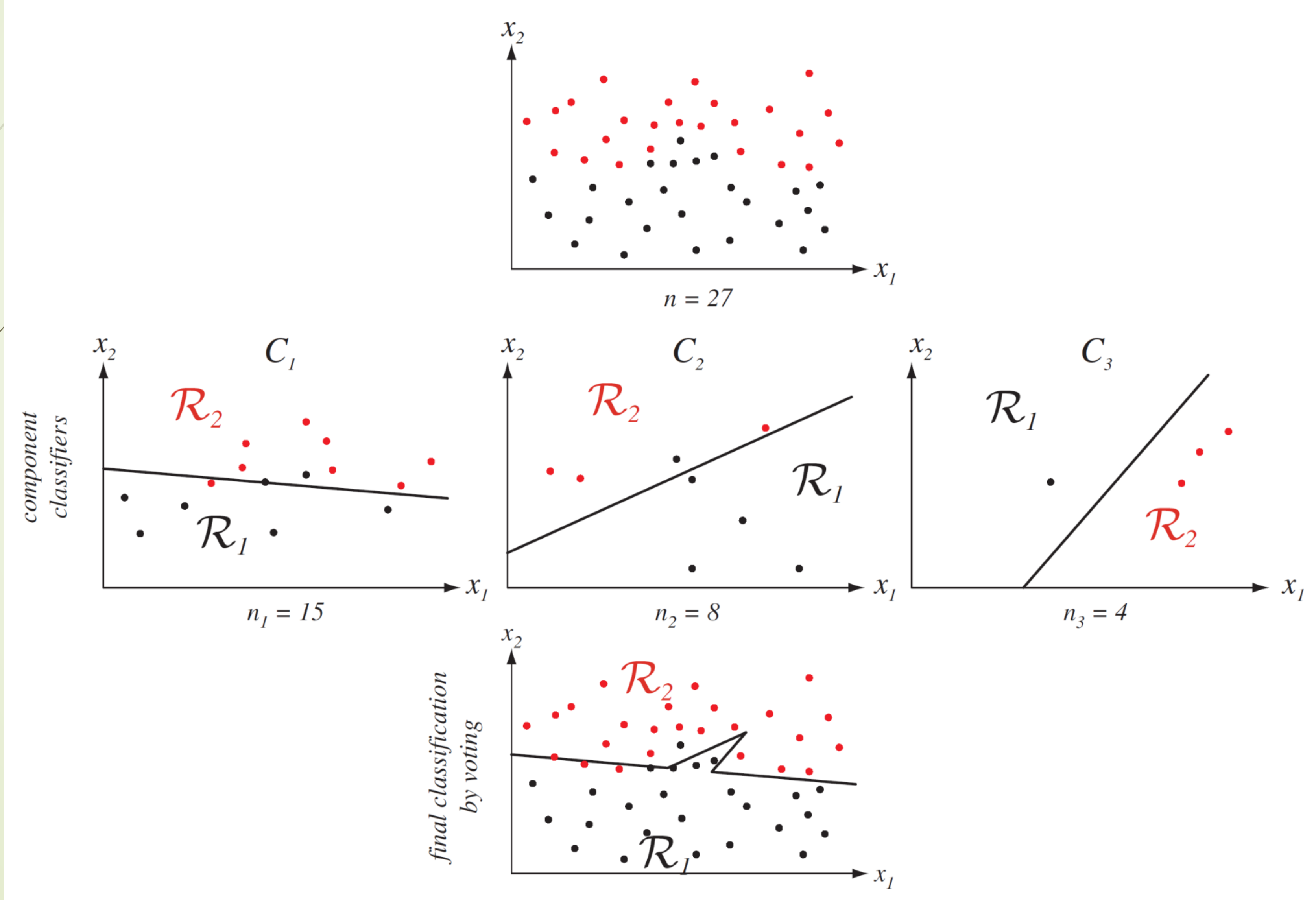
Boosting

- Boosting procedure
  - Randomly select a set of $n_1 < n$ patterns from the full training set D without replacement; call this set D_1
  - Train the first component classifier C_1 with D_1; C_1 is usually a weak learner
    - A weak leaner has accuracy only slightly better than chance.
  - Seek a second training set D_2, that is "most informative" given component classifier C_1
    - Half of the patterns in D_2 should be correctly classified by C_1
    - Half of the patterns in D_2 should be incorrectly classified by C_1 sampled from the remaining samples in D
  - Train the second component classifier C_2 with D_2

Boosting

- Boosting procedure
  - Then, the third component classifier C_3 with D_3
    - The samples in D_3 are selected from the remaining samples in D that are not well classified by voting by C_1 and C_2 (i.e., the classification results are not consistent)
  - The final classification decision is based on the votes of the component classifiers.

Boosting



x_2
x_1
$n = 27$
component classifiers
C_1
$\mathcal{R}_2$
$\mathcal{R}_1$
$n_1 = 15$
C_2
$\mathcal{R}_2$
$\mathcal{R}_1$
$n_2 = 8$
C_3
$\mathcal{R}_1$
$\mathcal{R}_2$
$n_3 = 4$
final classification by voting
$\mathcal{R}_2$
$\mathcal{R}_1$

Boosting

- How to decide n_1
  - We would like to train the ensemble classifier using all patterns in D.
  - A reasonable guess $n_1 \cong n_2 \cong n_3 = \frac{n}{3}$
  - In practice, we need to run the overall boosting procedure a few times, adjusting n_1 to use the full training set and get roughly equal partitions of the training set, if possible.

AdaBoosting

- Overview of AdaBoosting
  - A variant of boosting
  - Adaptive Boosting
  - Add weak learners until some desired low training error has been achieved.
  - Each training pattern receives a weight that determines its probability of being selected for a training set for an individual component classifier.
  - Adaboost “focuses on” the informative or “difficult” patterns.

AdaBoosting

Algorithm 1 (AdaBoost)

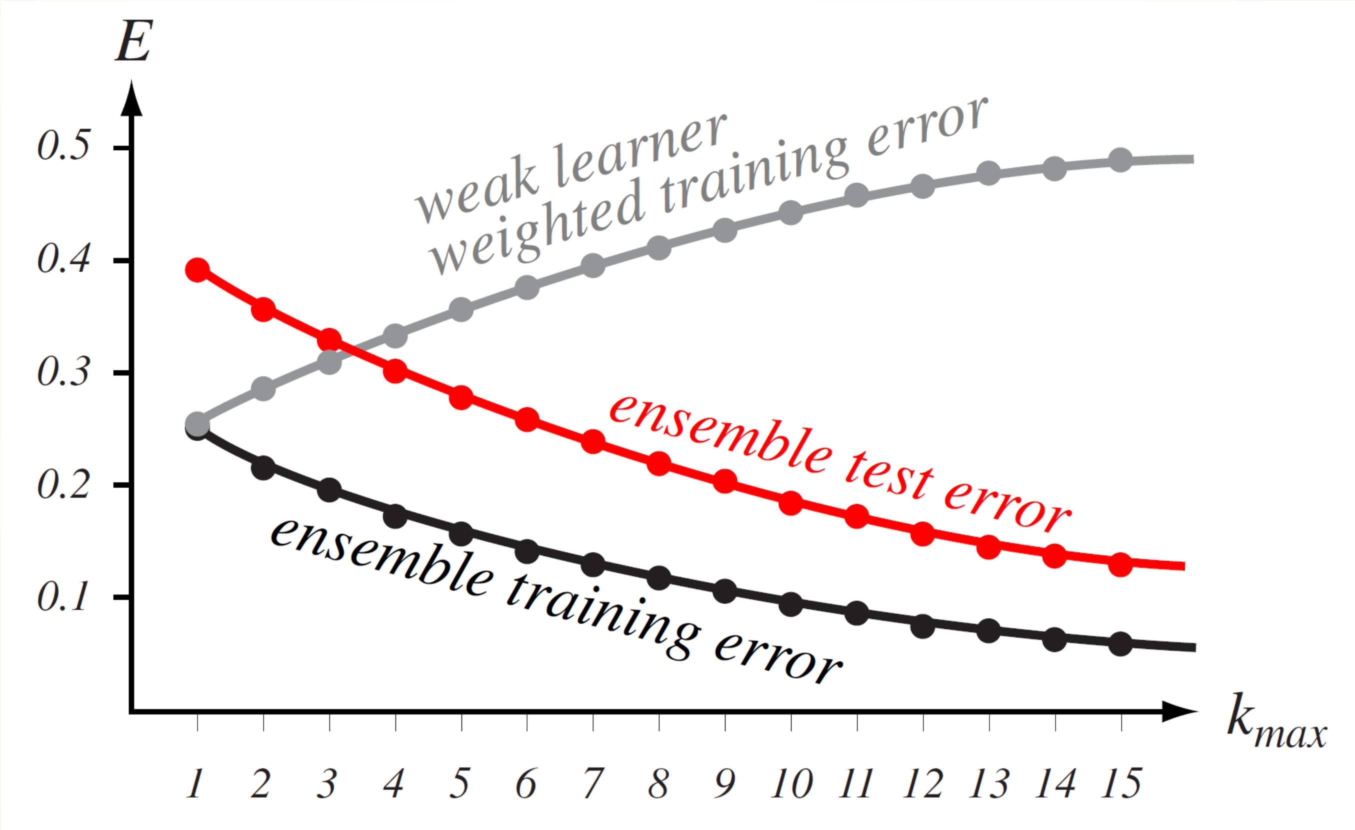
begin initialize $\mathcal{D} = \{\mathbf{x}^1, y_1, \mathbf{x}^2, y_2, \ldots, \mathbf{x}^n, y_n\}, k_{max}, W_1(i) = 1/n, i = 1, \ldots, n$
    $k \leftarrow 0$
    do $k \leftarrow k + 1$
        Train weak learner C_k using $\mathcal{D}$ sampled according to distribution $W_k(i)$
        $E_k \leftarrow$ Training error of C_k measured on $\mathcal{D}$ using $W_k(i)$
        $\alpha_k \leftarrow \frac{1}{2}\ln[(1 - E_k)/E_k]$
        $W_{k+1}(i) \leftarrow \frac{W_k(i)}{Z_k} \times \begin{cases} e^{-\alpha_k} & \text{if } h_k(\mathbf{x}^i) = y_i \text{ (correctly classified)} \\ e^{\alpha_k} & \text{if } h_k(\mathbf{x}^i) \neq y_i \text{ (incorrectly classified)} \end{cases}$
    until $k = k_{max}$
  return C_k and α_k for $k = 1$ to k_{max} (ensemble of classifiers with weights)
end

$$g(\mathbf{x}) = \sum_{k=1}^{k_{\max}} \alpha_k h_k(\mathbf{x}),$$

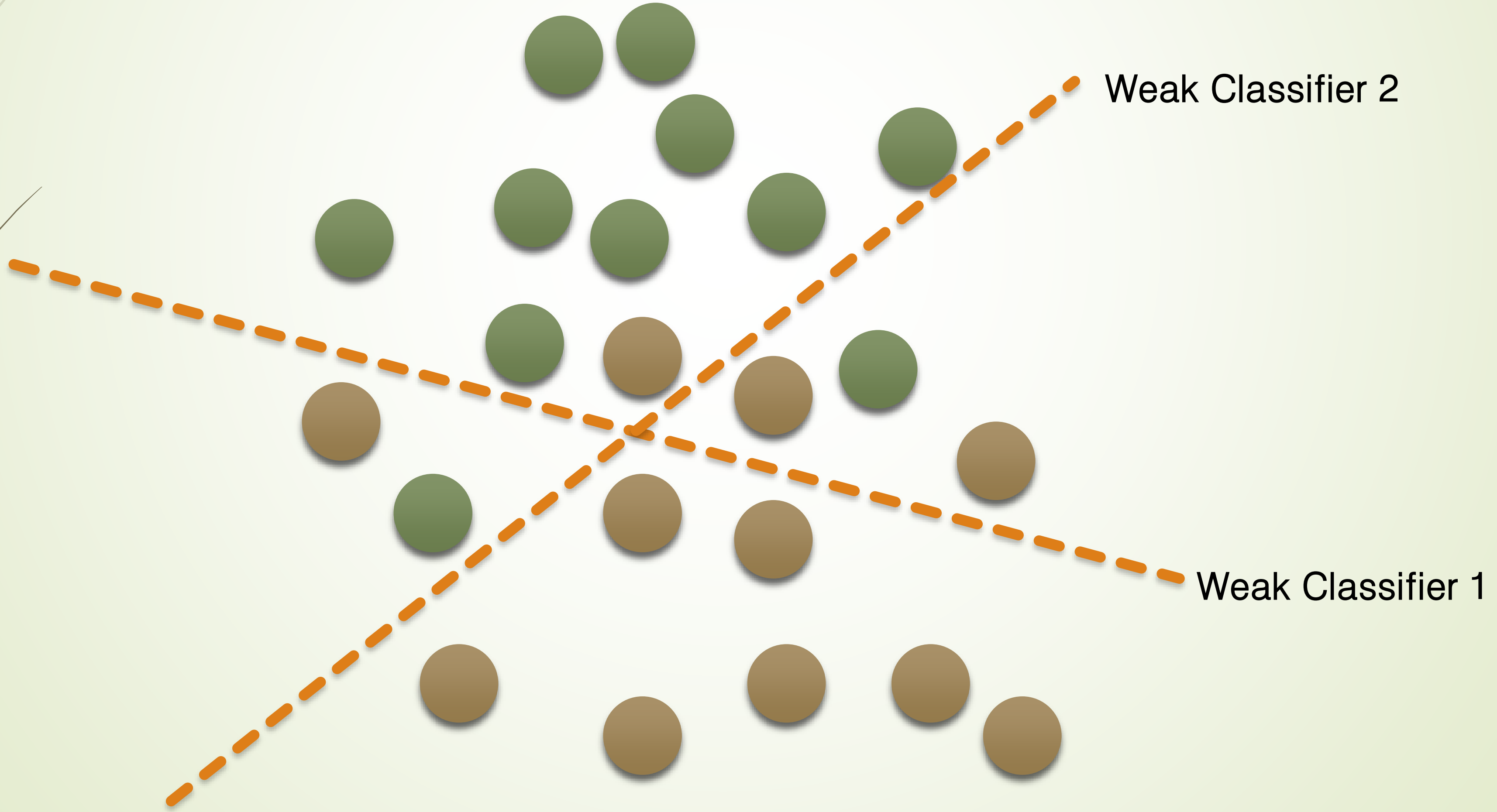
$h_k(\mathbf{x}) = +1 \text{ or } -1,$

the category label given to $\mathbf{x}$ by componet classifier C_k

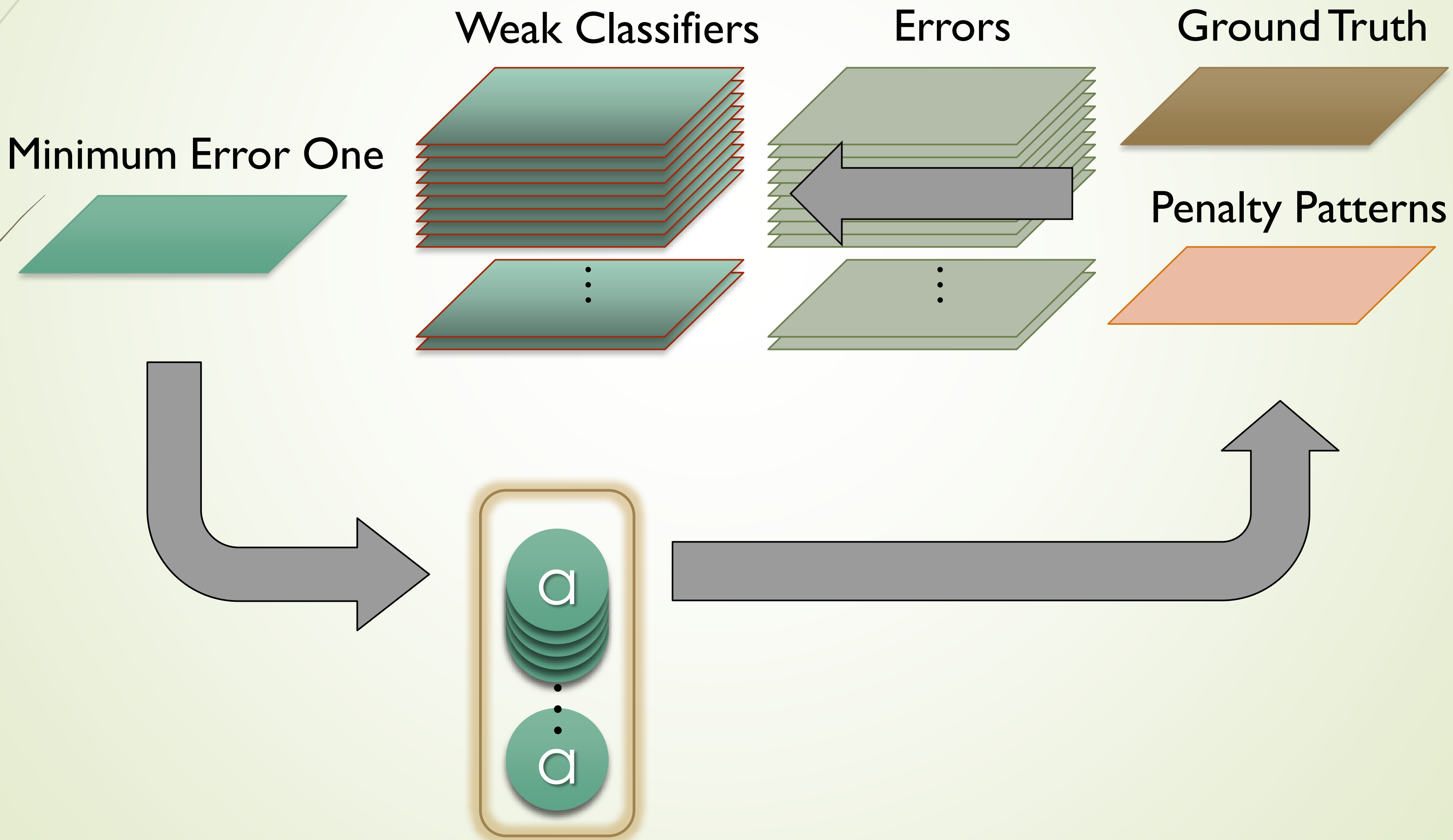
AdaBoosting



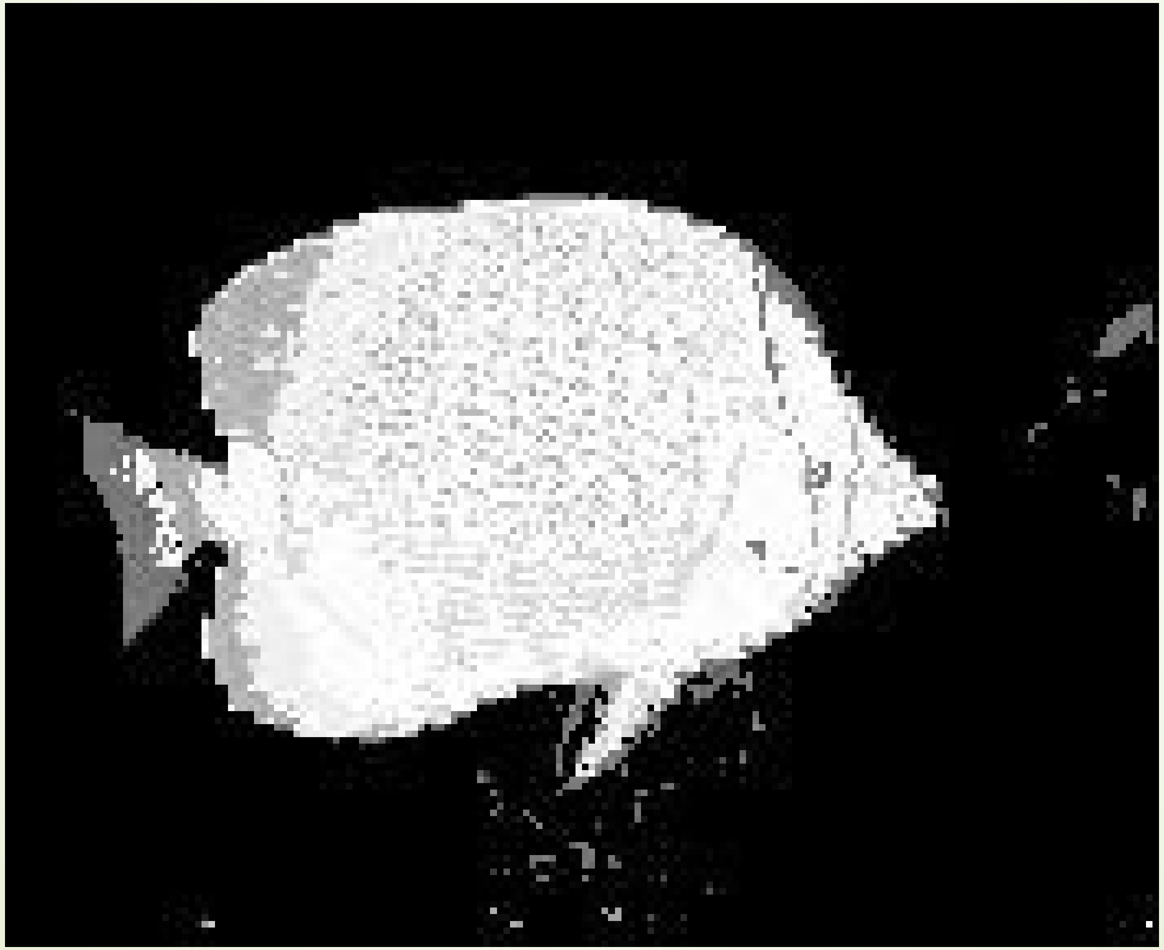
AdaBoosting for Classification



AdaBoosting for Classification



AdaBoosting for Classification



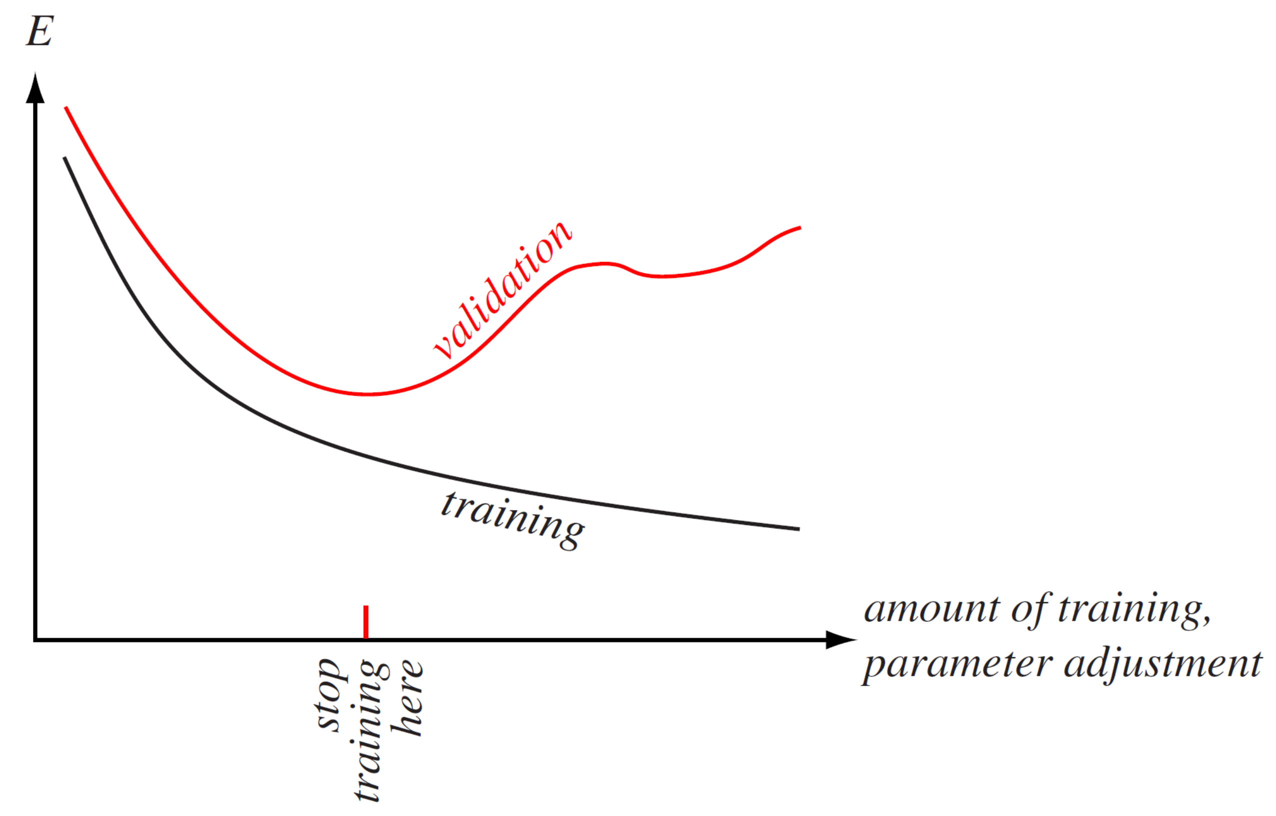
Random Forest

- Decision Tree Bagging
  - Given a training set $X = x_1, \cdots, x_n$ with labels $Y = y_1, \cdots, y_n$, bagging repeatedly (L times) selects a random sample with replacement of the training set and fits trees to these samples
  - For $i = 1, \ldots, L$:
    - Sample, with replacement, m training examples from X, Y; call these X_i, Y_i
    - Train a decision or regression tree f_i on X_i, Y_i
  - Prediction for an unseen x' can be made by averaging the predictions from all the individual trees on x'
- Random Forest use a random subset of the features in tree learning, called "feature bagging"

Cross-Validation

- Simple validation
  - Split the set of labeled training samples D into two parts
    - Traditional training set
      - Train the classifier
    - Validation set
      - Estimate the generalization error

Cross-Validation



Cross-Validation

- m-fold cross-validation
  - Split the training set D into m disjoint sets of equal size n/m
  - The classifier is trained m times, each time with a different set held out as a validation set
  - The estimated performance is the mean of the m errors
  - Leave-one-out approach $(m = n)$

Cross-Validation

- 5-fold cross-validation -> split the training data into 5 equal folds
- 4 of them for training and 1 for validation