

# Ensemble Classifiers

Srijith P K

Slide Credits : Vineeth N Balasubramanian



आई आई टी हैदराबाद  
IIT Hyderabad

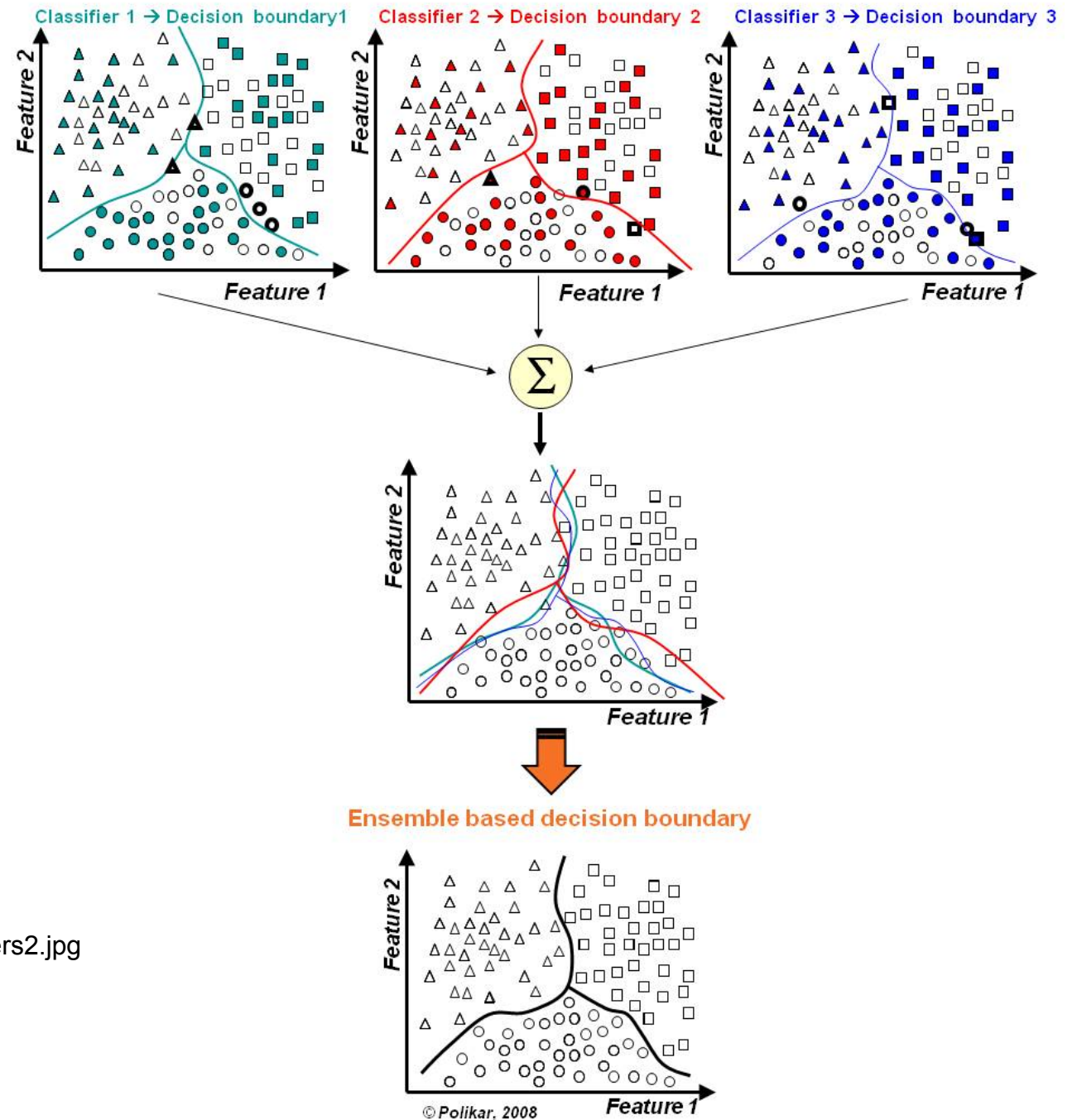
# Ensemble Learning



# Ensemble Learning

- Ensemble classification combines multiple classifiers to improve accuracy
- Use multiple methods to obtain better performance than any of the individual method
- Can combine outputs

# Ensemble Learning

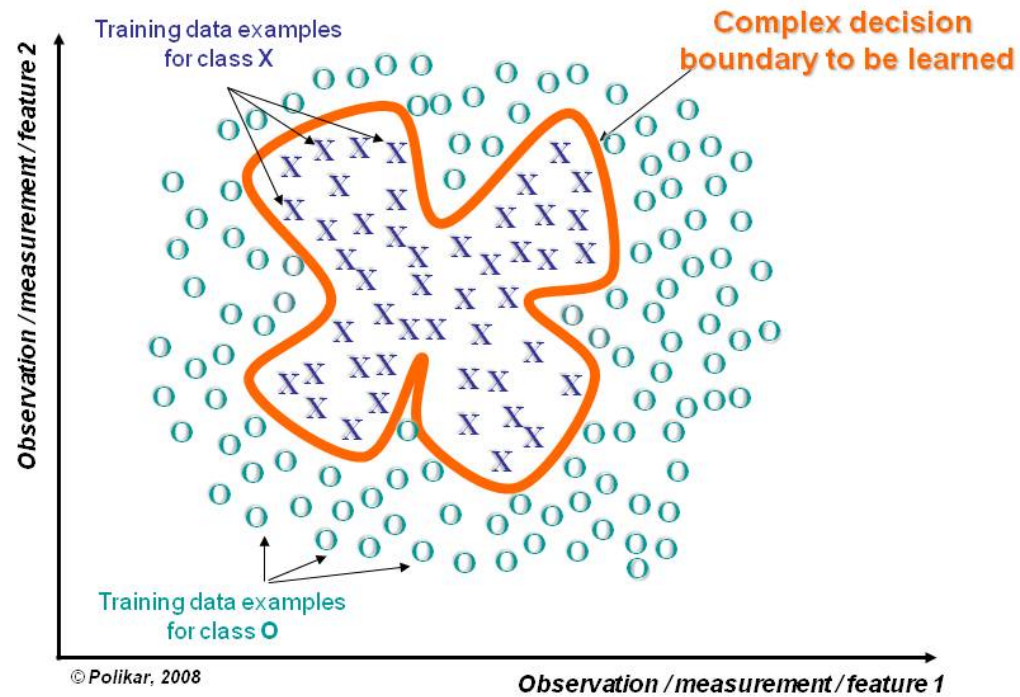


[http://www.scholarpedia.org/wiki/images/8/82/Combining\\_classifiers2.jpg](http://www.scholarpedia.org/wiki/images/8/82/Combining_classifiers2.jpg)

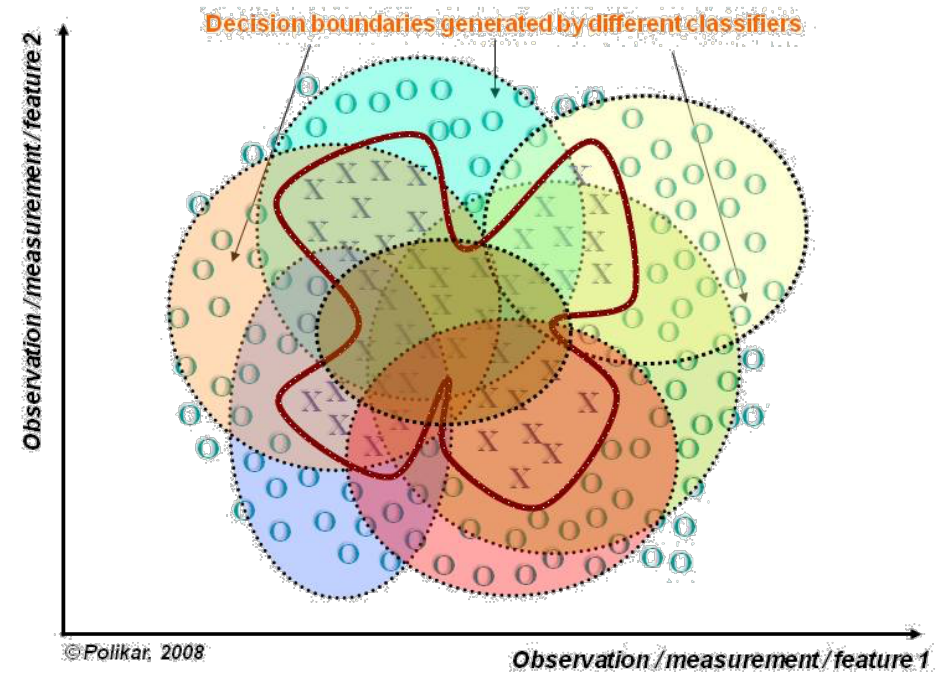
# Advantages

- Large datasets
  - May be too large for training single classifier
  - Can use different subsets of data to train multiple classifiers
- Small datasets
  - Can handle them with bootstrapping (random sampling)
- Some problems too complicated to solve with a single classifier
  - Eg. Complex separation between classes

# Divide and Conquer



<http://www.scholarpedia.org/article/Image:Figure1a.jpg>



<http://www.scholarpedia.org/wiki/images/a/ab/Figure1b.jpg>

# Advantages

- Model Selection
  - multilayer perceptron (MLP), support vector machines (SVM), decision trees, naive Bayes classifier ?
- Data Fusion
  - diagnosis of a neurological disorder, a neurologist may use the electroencephalogram (one-dimensional time series data), magnetic resonance imaging MRI, functional MRI, or positron emission tomography PET scan images (two-dimensional spatial data)
- Confidence Estimation
  - confidence to the decision made by such a system.

# Forming Ensembles : Diversity

- If all classifiers provided the same output, correcting a possible mistake would not be possible.
- If each classifier makes different errors, then a strategic combination of these classifiers can reduce the total error
- Ensemble system needs classifiers whose decision boundaries are adequately different from those of others.
  - Different training datasets to train individual classifiers.
  - Different training parameters for different classifiers.
  - Different type of classifiers, such MLPs, decision trees,..
  - different features, or different subsets of existing features.



# Types of Ensemble Classifiers

- Bagging (bootstrap aggregating)
  - Train several models using bootstrapped datasets
  - The majority classification is selected
- Boosting
  - Use several weak classifiers to create a strong classifier
  - Resample previously misclassified points
- Stacking (stacked generalization)
  - Train multiple tiers of classifiers
  - Higher tiers can correct lower tiers

# Bagging

- **Problem:** we only have one dataset.
- **Solution:** generate new ones of size  $n$  by **bootstrapping**, i.e. sampling it with replacement
- Bagging works because it reduces variance by voting/averaging
- Usually, the more classifiers the better
- Some candidates:
  - Decision tree, decision stump, SVMs
  - Can do this with regression too: Regression tree, linear regression

# Bagging

- **Problem:** we only have one dataset.
- **Solution:** generate new ones of size  $n$  by **bootstrapping**, i.e. sampling it with replacement
- Bagging works because it reduces variance by voting/averaging
- Usually, the more classifiers the better
- Some candidates:
  - Decision tree, decision stump, SVMs
  - Can do this with regression too: Regression tree, linear regression

# Bagging: Why does it work?

- Let  $S = \{(x_i, y_i), i = 1 \dots N\}$  be the training dataset
- Let  $\{S_k\}$  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

$$\varphi(x, P) = E_S(\varphi(x, S_k))$$

### Algorithm: Bagging

#### Input:

- Training data  $S$  with correct labels  $\omega_i, \Omega = \{\omega_1, \dots, \omega_C\}$  representing  $C$  classes
- Weak learning algorithm **WeakLearn**,
- Integer  $T$  specifying number of iterations.
- Percent (or fraction)  $F$  to create bootstrapped training data

Do  $t=1, \dots, T$

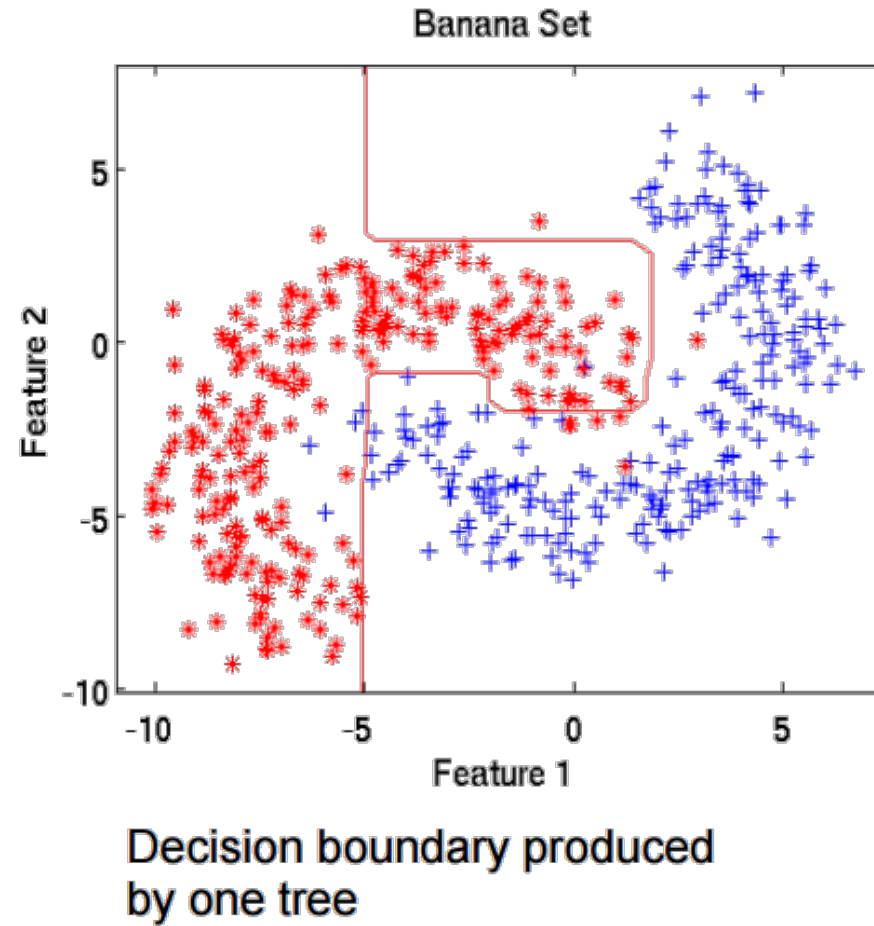
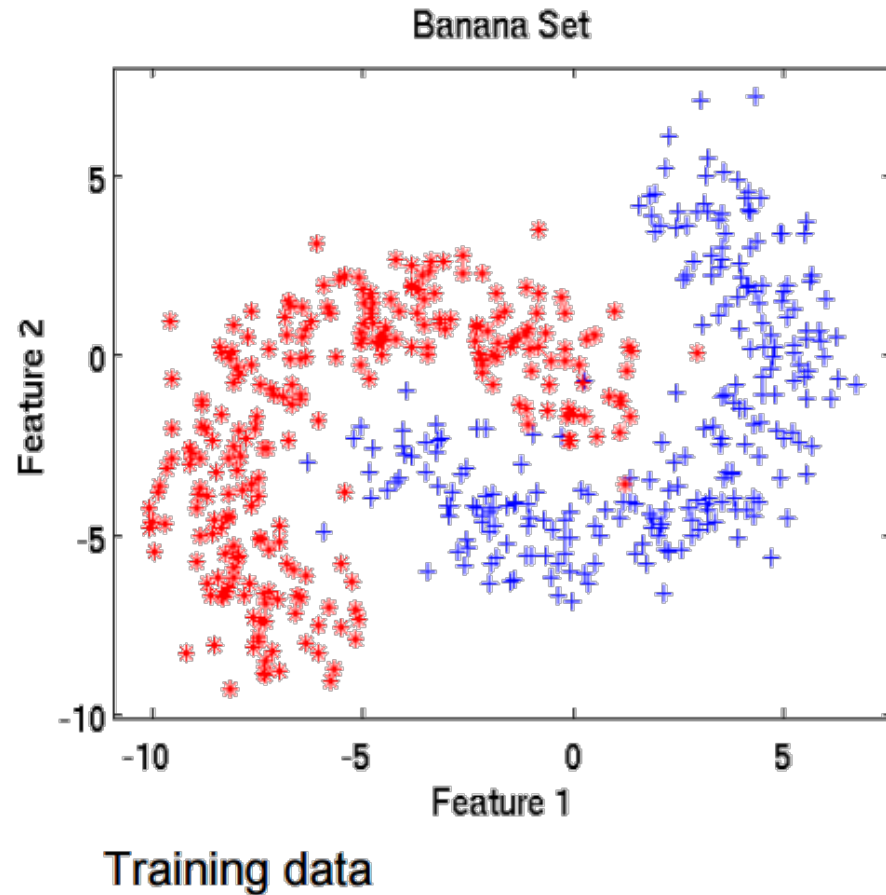
1. Take a bootstrapped replica  $S_t$  by randomly drawing  $F$  percent of  $S$ .
2. Call **WeakLearn** with  $S_t$  and receive the hypothesis (classifier)  $h_t$ .
3. Add  $h_t$  to the ensemble,  $\mathcal{E}$ .

End

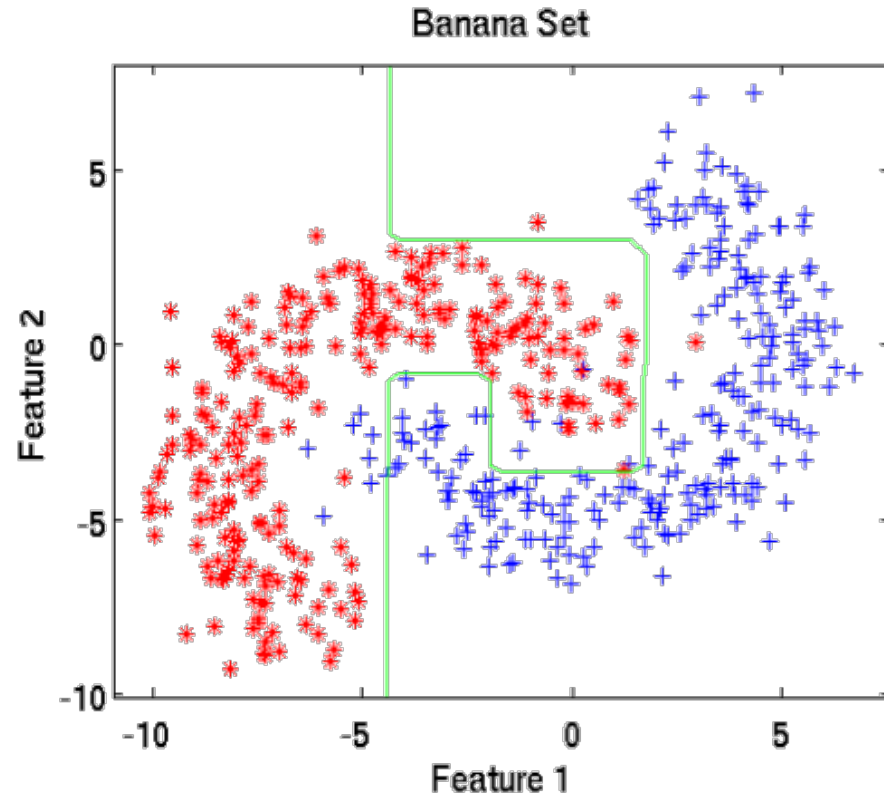
**Test: Simple Majority Voting** – Given unlabeled instance  $x$

1. Evaluate the ensemble  $\mathcal{E} = \{h_1, \dots, h_T\}$  on  $x$ .
2. Let 
$$v_{t,j} = \begin{cases} 1, & \text{if } h_t \text{ picks class } \omega_j \\ 0, & \text{otherwise} \end{cases}$$
 be the vote given to class  $\omega_j$  by classifier  $h_t$ . (1)
3. Obtain total vote received by each class, 
$$V_j = \sum_{t=1}^T v_{t,j} \quad j = 1, \dots, C. \quad (2)$$
4. Choose the class that receives the highest total vote as the final classification.

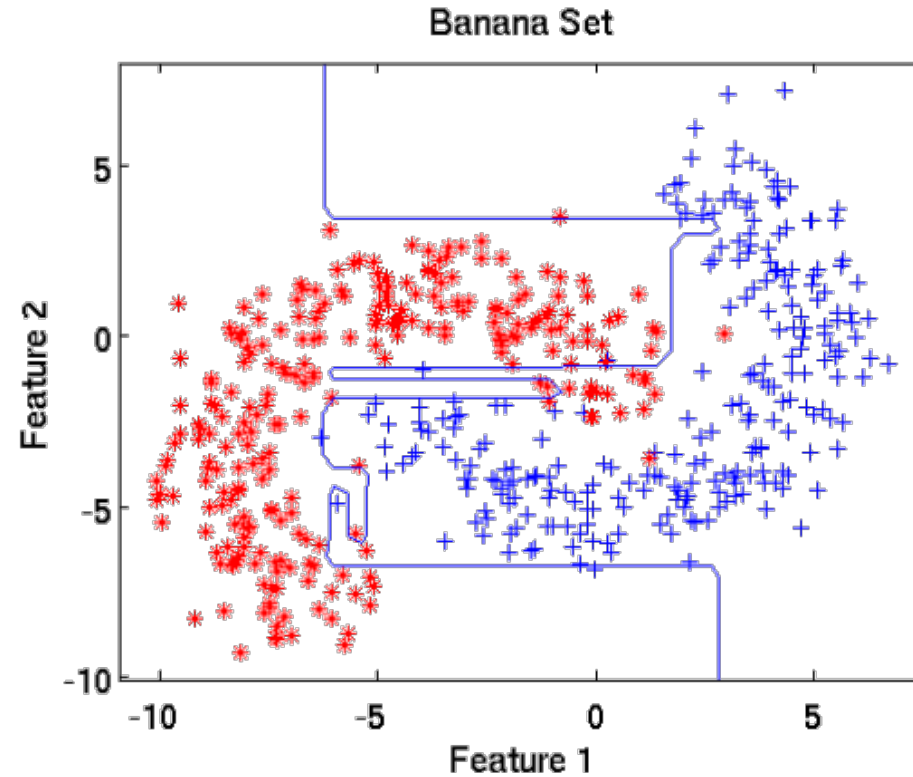
# Bagging: Illustration



# Bagging: Illustration

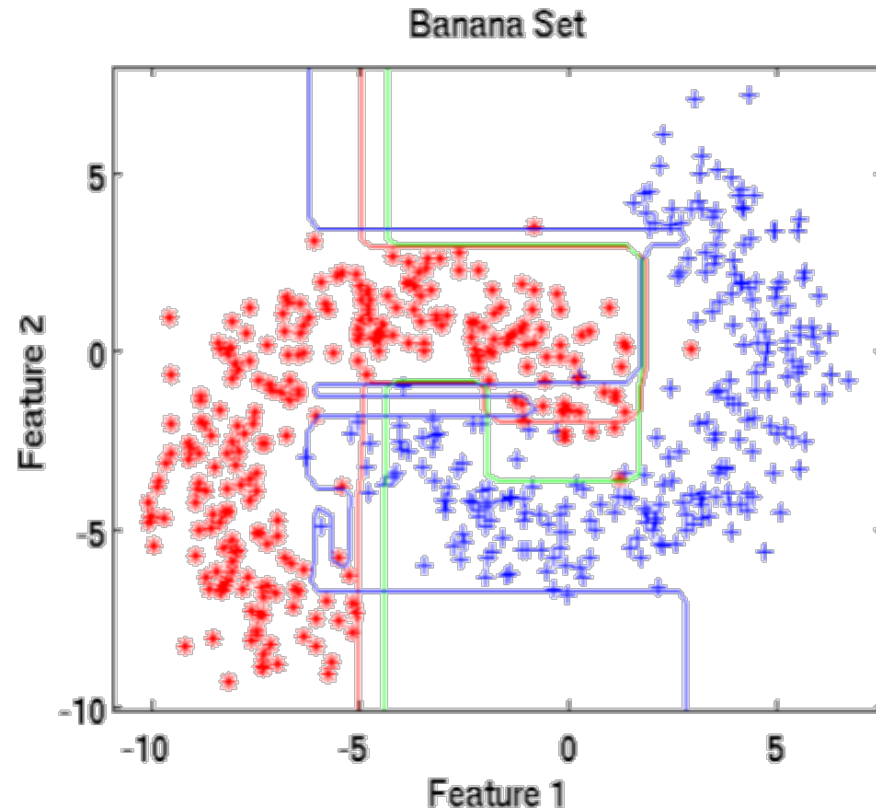


Decision boundary produced by a second tree

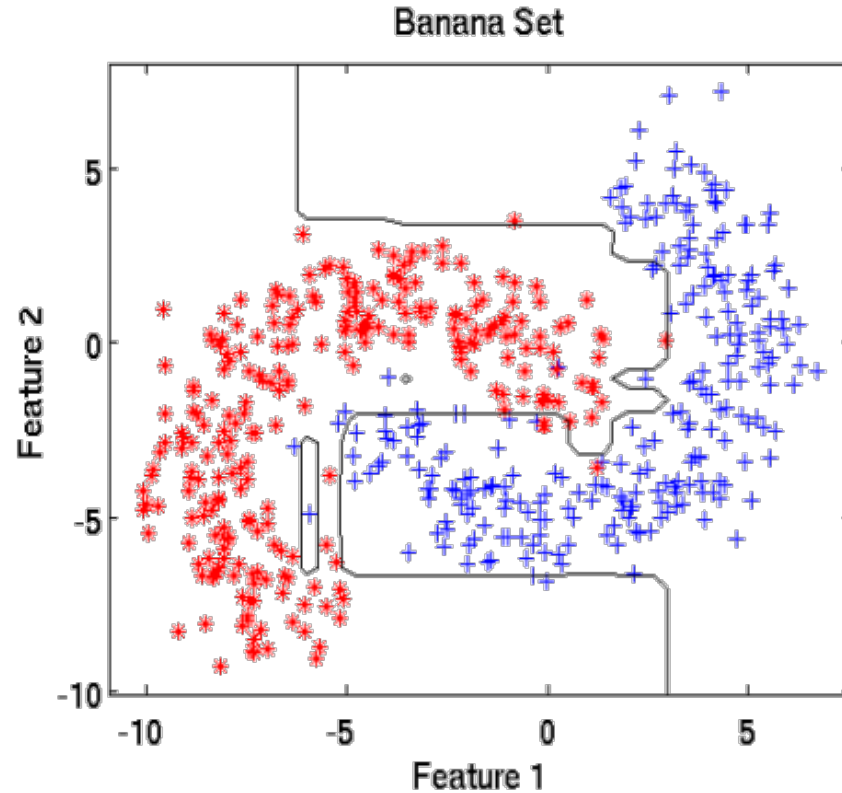


Decision boundary produced by a third tree

# Bagging: Illustration



Three trees and final boundary overlaid



Final result from bagging all trees.

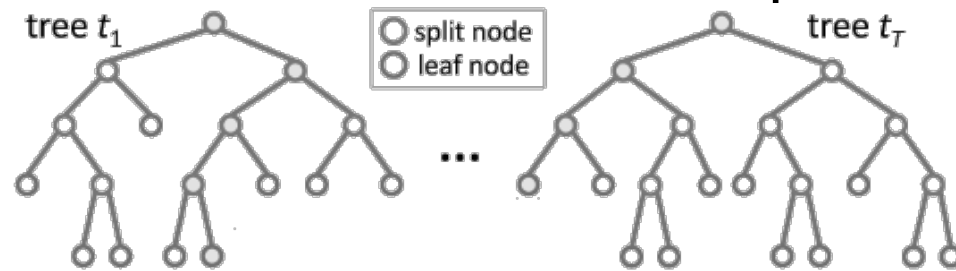


# Random Forest



# Random Forests

- Bagging, the decision trees can have a lot of structural similarities and in turn have high correlation in their predictions.
- Random forests (RF) are a combination of tree predictors
  - Variant of bagging, proposed by Breiman in 2001
- Extremely successful, especially on Kaggle challenges
- The generalization error depends on the strength of the individual trees and the correlation between them
- learning algorithm is limited to a random sample of features of which to search.



# Random Forests: Algorithm

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

# Random Forests: Algorithm

- If there are  $M$  input variables, a number  $m$  is specified such that at each node,  $m$  variables are selected at random out of the  $M$  and the best split on these  $m$  is used to split the node. The value of  $m$  is held constant during the forest growing.
  - Depending upon the value of  $m$ , there are three slightly different systems:
    - Random splitter selection:  $m = 1$
    - Breiman's bagger:  $m = \text{total number of predictor variables}$
    - Random forest:  $m \ll \text{number of predictor variables}$ . Breiman suggests three possible values for  $m$ :  $\frac{1}{2}\sqrt{M}$ ,  $\sqrt{M}$ , and  $2\sqrt{M}$
- Each tree is grown to the largest extent possible. There is no pruning.

# Bagging: When does it work?

- Can help if data is noisy
- If learning algorithm is unstable, i.e. if small changes to the training set cause large changes in the learned classifier

# Features of Random Forests

- One of the best in the business today
- It runs efficiently on large data bases
- It can handle thousands of input variables without variable deletion/reduction
- It gives estimates of what variables are important in the classification
- Does not overfit by design
- The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them

# Types of Ensemble Classifiers

- Bagging (bootstrap aggregating)
  - Train several models using bootstrapped datasets
  - The majority classification is selected
- Boosting
  - Use several weak classifiers to create a strong classifier
  - Resample previously misclassified points
- Stacking (stacked generalization)
  - Train multiple tiers of classifiers
  - Higher tiers can correct lower tiers



# Boosting

- **Strong Learner** → Objective of machine learning
  - Take labeled data for training
  - Produce a classifier which can be *arbitrarily well-correlated with the true classification.*
- **Weak Learner**
  - Take labeled data for training
  - Produce a classifier which is only slightly correlated with the true classification (**more accurate than random guessing**)

Can a set of **weak learners** create a single **strong learner** ?



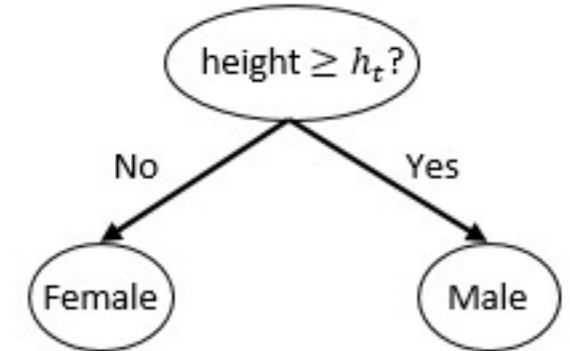
# Boosting: Key Idea

- Weak classifiers are learnt sequentially
- An algorithm for constructing a “strong” classifier as linear combination of “simple” “weak” classifier

$$f(x) = \sum_{t=1}^T \alpha_t h_t(x)$$

- Final classification based on weighted vote of weak classifiers

Decision stump



# Boosting: Key Idea

- Resampling is strategically geared to provide the most informative training data for each consecutive classifier.
- Associates a weight to each data point
- After a weak learner is added, the data weights are readjusted, known as "**re-weighting**". Misclassified input data gain a higher weight and examples that are classified correctly lose weight
- Future weak learners focus more on the examples that previous weak learners misclassified

# Adaboost Algorithm (won the prestigious Gödel Prize.)

Given:  $(x_1, y_1), \dots, (x_m, y_m)$  where  $x_i \in X$ ,  $y_i \in Y = \{-1, +1\}$

Initialise  $D_1(i) = \frac{1}{m}$ .

For  $t = 1, \dots, T$ :

- Find the classifier  $h_t : X \rightarrow \{-1, +1\}$  that minimizes the error with respect to the distribution  $D_t$ :

$$h_t = \arg \min_{h_j \in \mathcal{H}} \epsilon_j, \text{ where } \epsilon_j = \sum_{i=1}^m D_t(i) [y_i \neq h_j(x_i)]$$

- Prerequisite:  $\epsilon_t < 0.5$ , otherwise stop.
- Choose  $\alpha_t \in \mathbf{R}$ , typically  $\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}$  where  $\epsilon_t$  is the weighted error rate of classifier  $h_t$ .
- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where  $Z_t$  is a normalisation factor (chosen so that  $D_{t+1}$  will be a distribution).

Output the final classifier:

$$H(x) = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(x) \right)$$

Each training sample has a weight, which determines the probability of being selected for training the component classifier

# Reweighting

## Effect on the training set

Reweighting formula:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} = \frac{\exp(-y_i \sum_{q=1}^t \alpha_q h_q(x_i))}{m \prod_{q=1}^t Z_q}$$

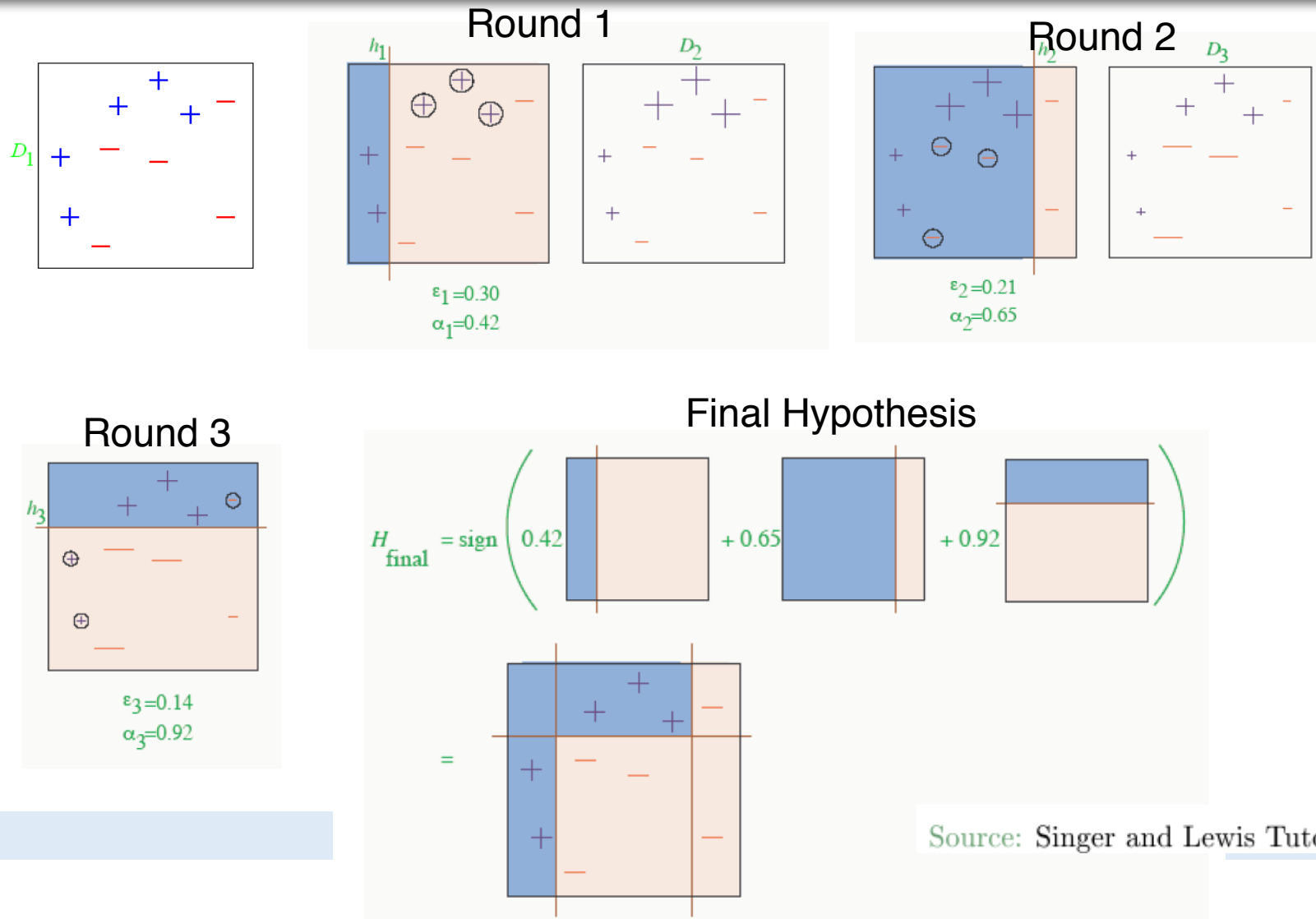
$$\exp(-\alpha_t y_i h_t(x_i)) \begin{cases} < 1, & y_i = h_t(x_i) \\ > 1, & y_i \neq h_t(x_i) \end{cases}$$

$$y * h(x) = 1$$

$$y * h(x) = -1$$

⇒ Increase (decrease) weight of wrongly (correctly) classified examples

# Simple Example



Source: Singer and Lewis Tutorial

# Real-world Use of Boosting: Face Detection

Viola–Jones face detection algorithm employs AdaBoost with decision stumps as weak learners

A landmark paper in vision!



Viola and Jones  
2001

More details: [https://en.wikipedia.org/wiki/Viola%E2%80%93Jones\\_object\\_detection\\_framework](https://en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework)

# Adaboost vs Random Forests

- Dietterich (1998) showed that
  - when a fraction of the output labels in the training set are randomly altered, the accuracy of Adaboost degenerates, while bagging is more immune to the noise.

Increases in error rates due to noise

Data set	Adaboost	Forest-RI
Glass	1.6	.4
Breast cancer	43.2	1.8
Diabetes	6.8	1.7
Sonar	15.1	−6.6
Ionosphere	27.7	3.8
Soybean	26.9	3.2
Ecoli	7.5	7.9
Votes	48.9	6.3
Liver	10.3	−.2

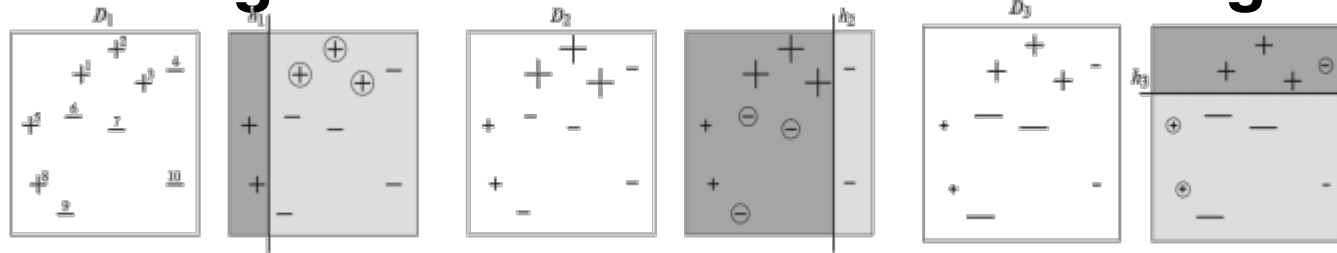
# What is a good weak learner?

- The set of weak rules (features) should be **flexible enough to be (weakly) correlated** with most conceivable relations between feature vector and label.
- **Small enough to allow exhaustive search** for the minimal weighted training error.
- **Small enough to avoid over-fitting.**
- Should be able to **calculate predicted label very efficiently.**



# Gradient Boosting

- **Gradient Boosting = Gradient Descent + Boosting**



- Fit an additive model (ensemble) in a forward stage-wise manner.
- In each stage, introduce a weak learner to compensate the shortcomings of existing weak learners.
- In Gradient Boosting, “shortcomings” are identified by gradients.
- Recall that, in Adaboost, “shortcomings” are identified by high-weight data points.
- Both high-weight data points and gradients tell us how to improve our model.

# Gradient Boosting

- **Gradient Boosting = Gradient Descent + Boosting**
- In Gradient Boosting, “shortcomings” are identified by gradients.
- At some stage  $m$ , assume there is a imperfect model  $F_m$
- Gradient boosting algorithm improves on  $F_m$  by constructing a new model that adds an estimator  $h$ ,  $F_{m+1}(x) = F_m(x) + h(x) = y$
- Perfect  $h(x)$  such that  $h(x) = y - F_m(x)$  **Residual**
- **gradient boosting will fit  $h(x)$  to the residual**
- $y - F(x)$  is negative gradient with respect to of the squared error loss function  $\frac{1}{2} (y - F(x))^2$

# Gradient Boosting

Input: training set  $\{(x_i, y_i)\}_{i=1}^n$ , a differentiable loss function  $L(y, F(x))$ , number of iterations  $M$ .

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, \gamma).$$

2. For  $m = 1$  to  $M$ :

1. Compute so-called *pseudo-residuals*:

$$r_{im} = - \left[ \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \dots, n.$$

2. Fit a base learner (e.g. tree)  $h_m(x)$  to pseudo-residuals, i.e. train it using the training set  $\{(x_i, r_{im})\}_{i=1}^n$

3. Compute multiplier  $\gamma_m$  by solving the following **one-dimensional optimization** problem:

$$\gamma_m = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)).$$

4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

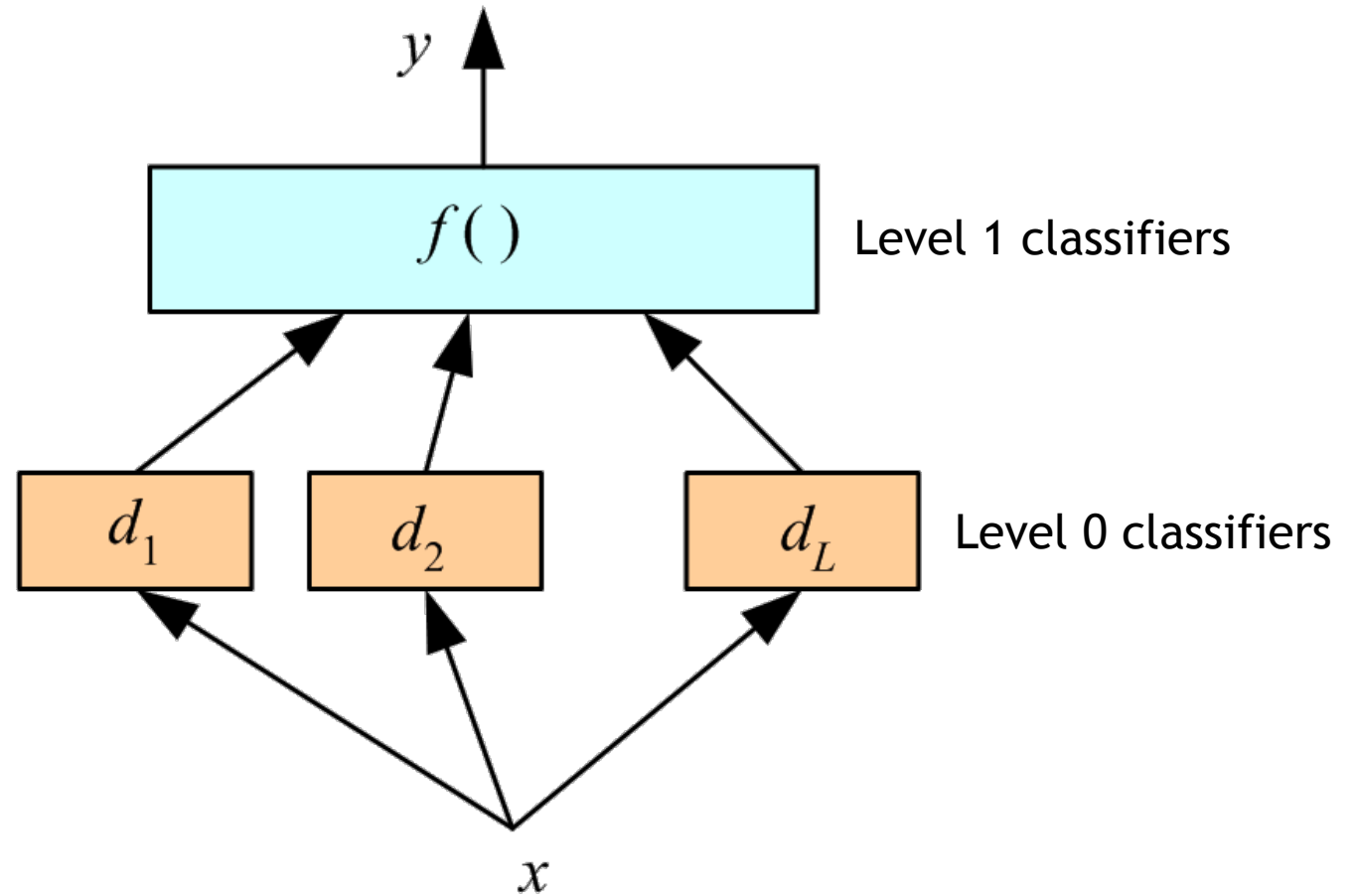
3. Output  $F_M(x)$ .

# Types of Ensemble Classifiers

- Bagging (bootstrap aggregating)
  - Train several models using bootstrapped datasets
  - The majority classification is selected
- Boosting
  - Use several weak classifiers to create a strong classifier
  - Resample previously misclassified points
- Stacking (stacked generalization)
  - Train multiple tiers of classifiers
  - Higher tiers can correct lower tiers

# Stacked Ensembles

- Combiner  $f()$  is another learner (Wolpert, 1992)
- In practice, often equivalent to voting

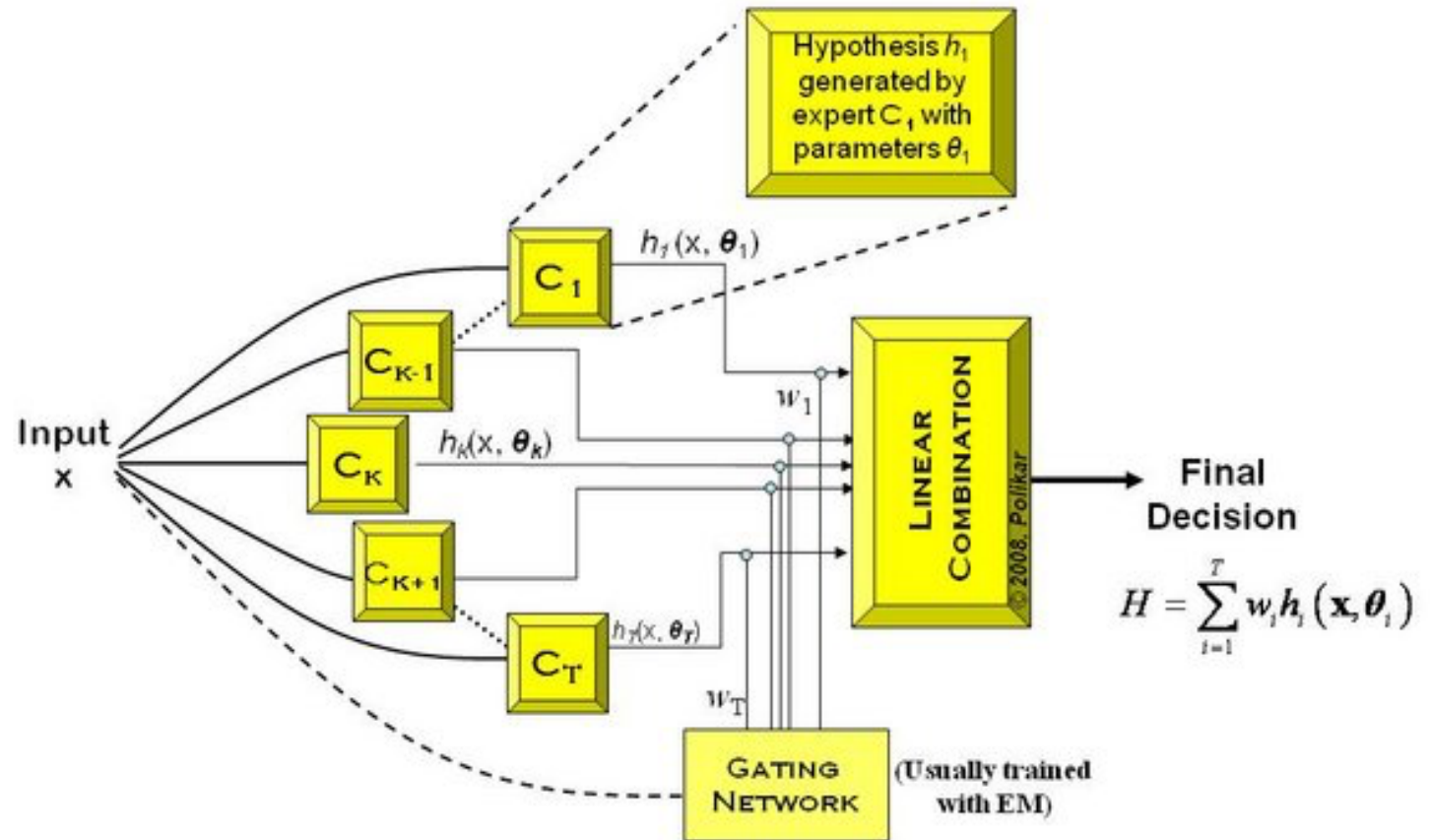


# Mixture of Experts

outputs are combined through a (generalized) linear rule.

weights of this combination are determined by a gating network

gating network requires the input instances for training.



# Readings

- [“Introduction to Machine Learning” by Ethem Alpaydin](#), Chapter 17
- Bishop, PRML (2006 edn), Sections 14.2-14.4
- Boosting Algorithms: A Review of Methods, Theory, and Applications, Ferreira and Figueiredo