Ensemble Classifiers

- Srijith P K

आई आई टी हैदराबाद
IIT Hyderabad

Product purchase

Best laptop in the market with all latest features.
Reviewed in India on 21 October 2019
Size name: without Accessory | Style name: Core i7-9750H 9th Gen

i am a graphic designer and video editor Man super duper laptop . Best laptop till now available in the market. If price is around 1.60 - 1.70 than I buy. Till wait .

Best
Reviewed in India on 28 October 2019
Size name: without Accessory | Style name: Core i9-9980HK 9th Gen

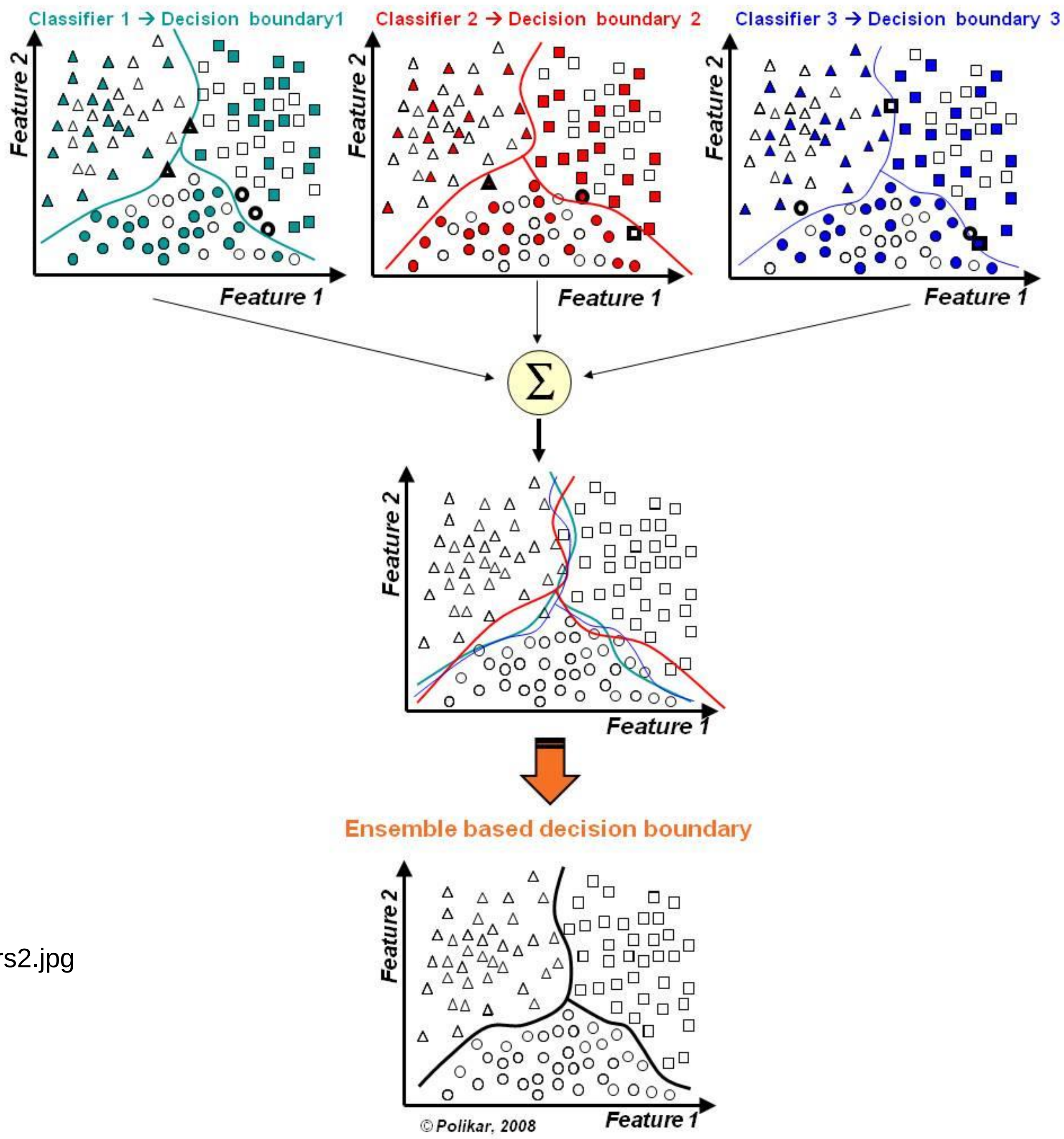
I received it within 36 hours and it is worth its price.
System boots up within 1 second and games are running butterly smooth.

Asus Zenbook Pro Duo Laptop Overview It has 2 Screens!

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

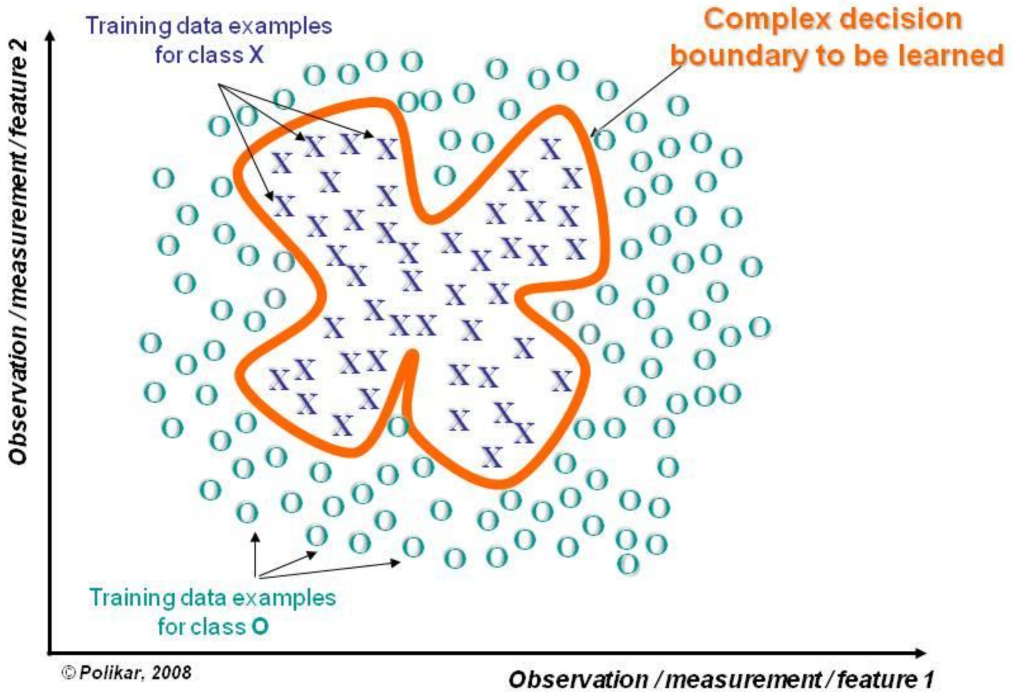


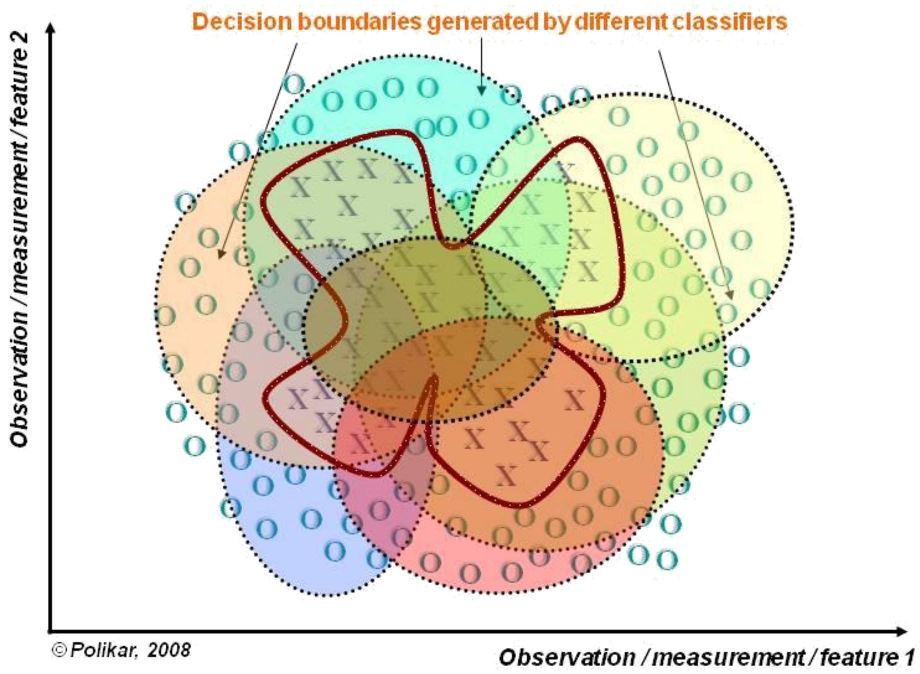
http://www.scholarpedia.org/wiki/images/8/82/Combining_classifiers2.jpg

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

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 $S = \{(x_i, y_i), i = 1 \ldots N\}$ 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 $\omega_i \Omega = \{\omega_1,...,\omega_C\}$ 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, ..., 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, $\mathcal{E}$.

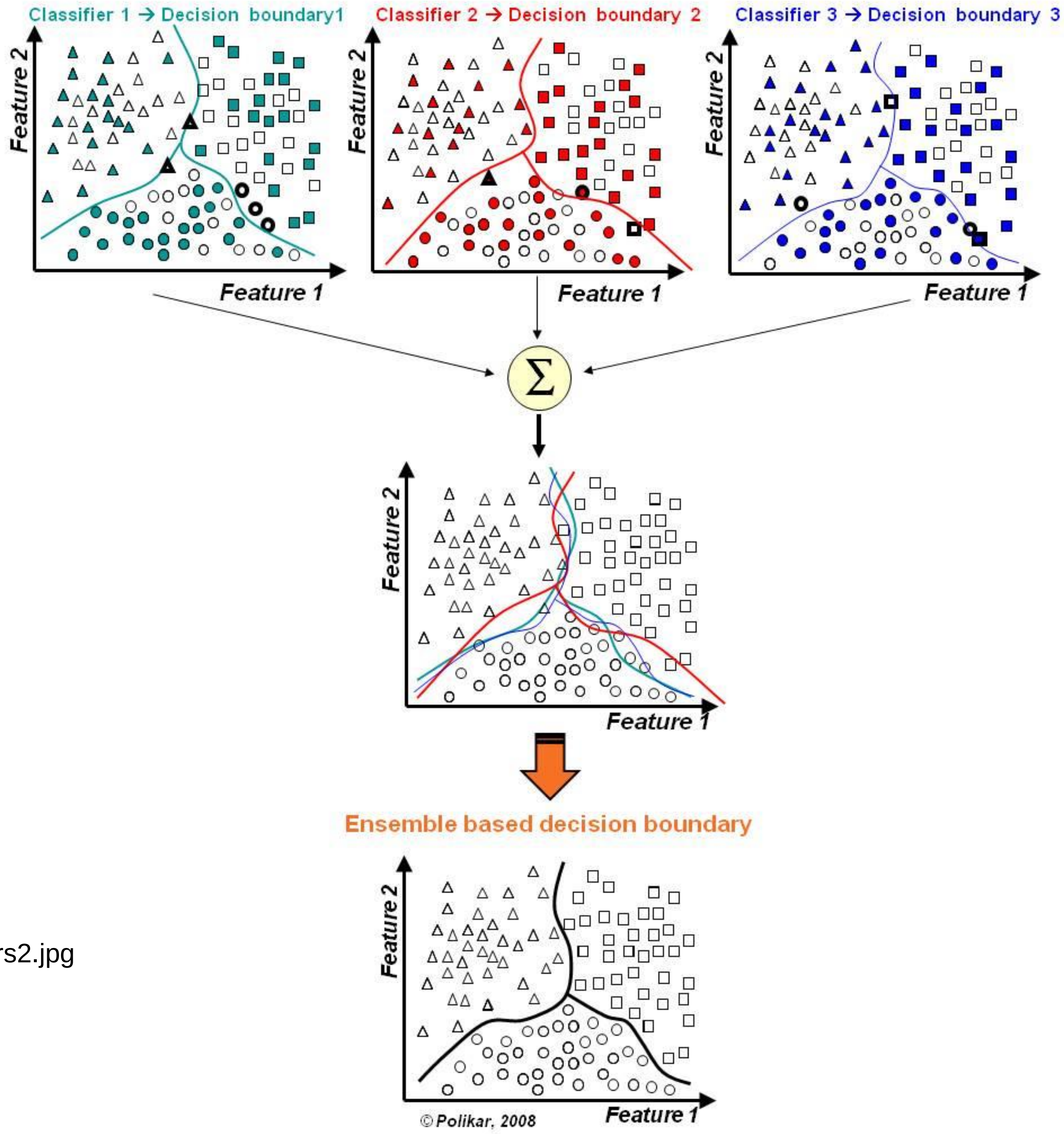
End

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

1. Evaluate the ensemble $\mathcal{E} = \{h_1,...,h_T\}$ 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 ω_j by classifier h_t. (1)
3. Obtain total vote received by each class, $V_j = \sum_{t=1}^{T} v_{t,j}$ $j = 1,...,C$. (2)
4. Choose the class that receives the highest total vote as the final classification.

Polikar, © 2008

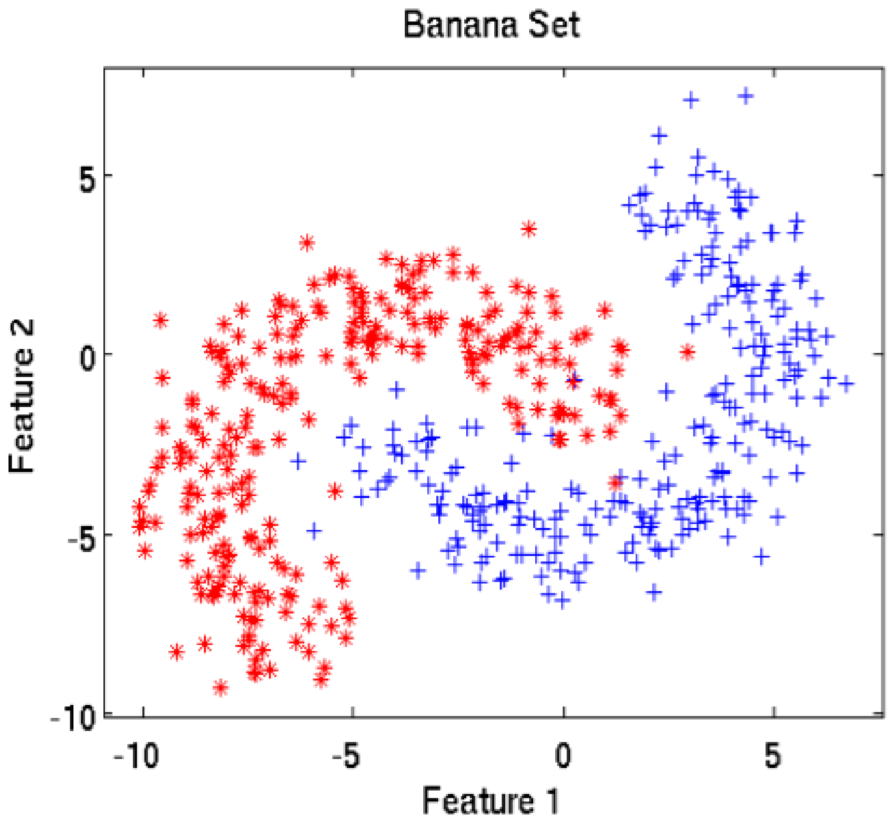
Ensemble Learning



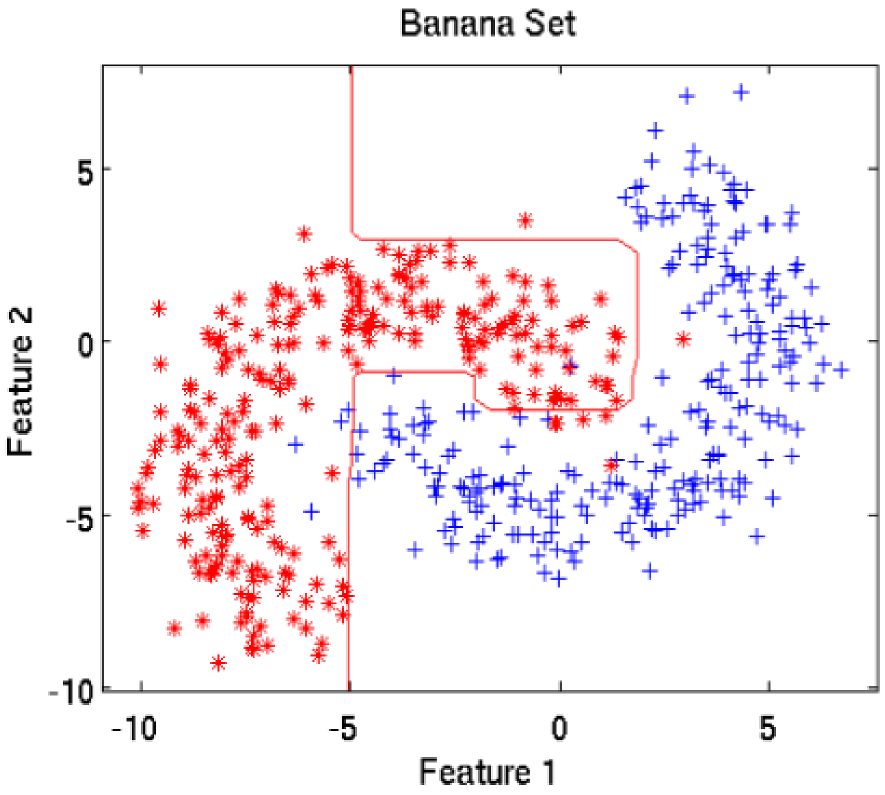
http://www.scholarpedia.org/wiki/images/8/82/Combining_classifiers2.jpg

आई आई टी हैदराबाद
IIT Hyderabad

Bagging: Illustration

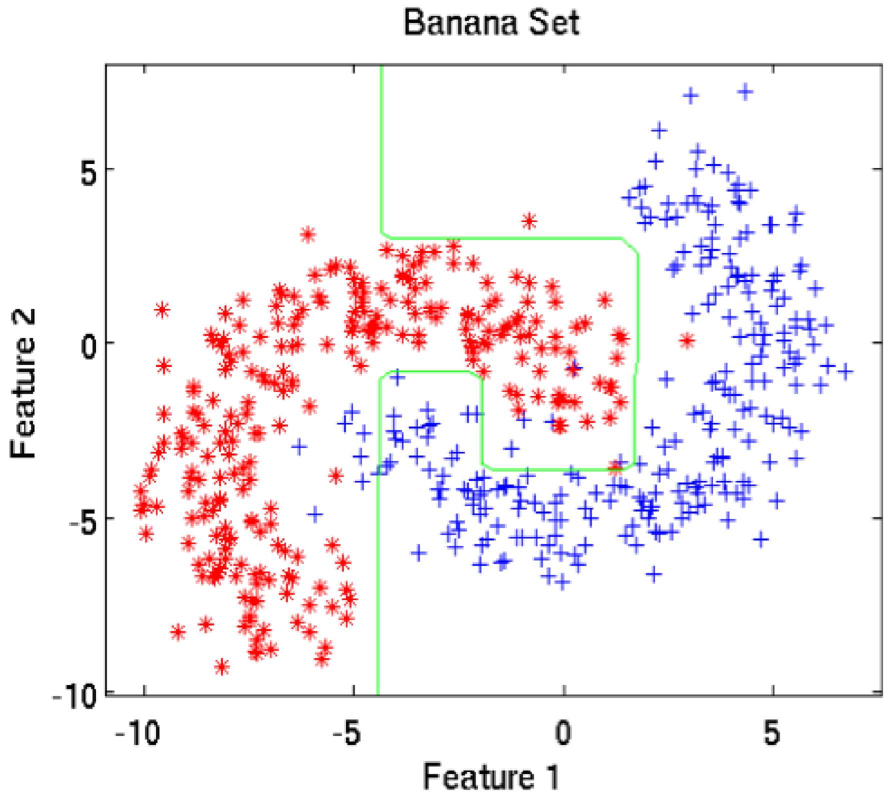


Training data

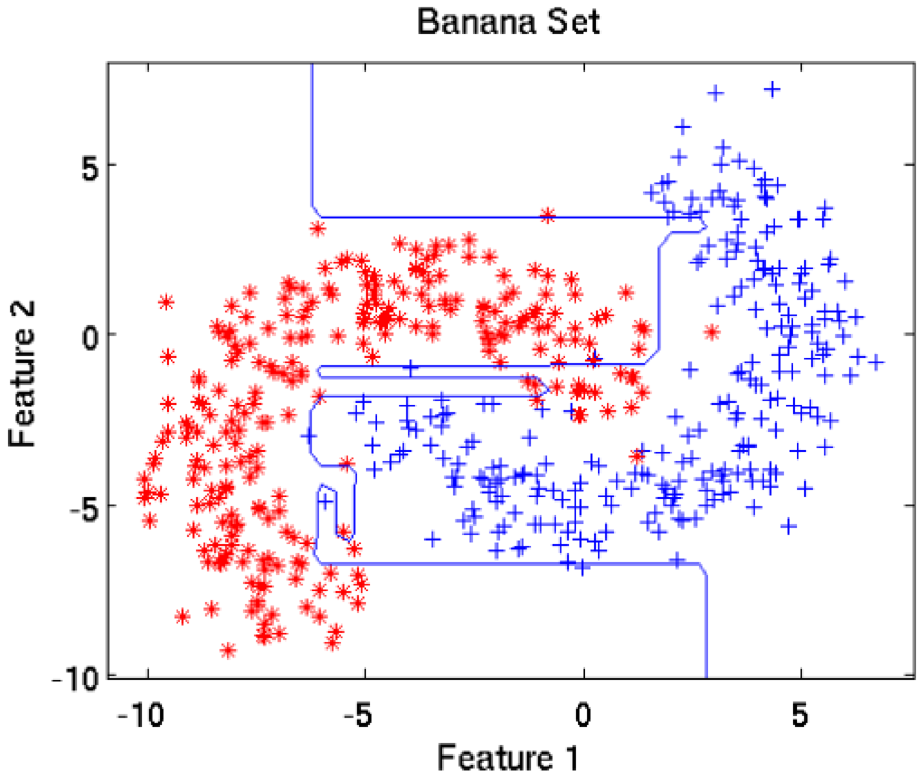


Decision boundary produced by one tree

Bagging: Illustration

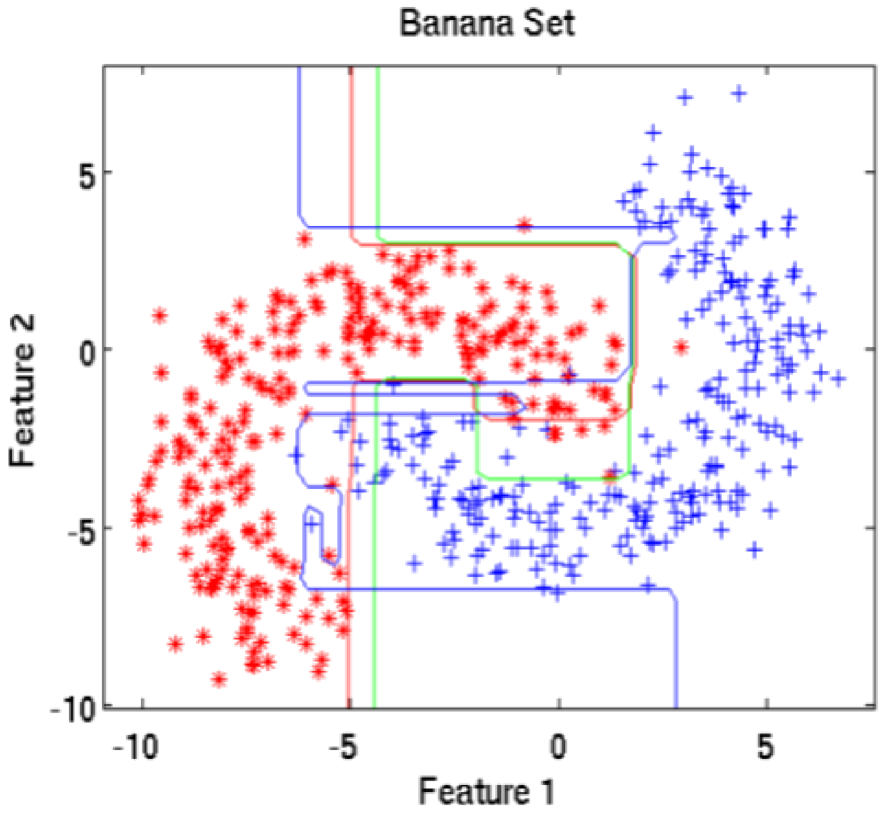


Decision boundary produced by a second tree

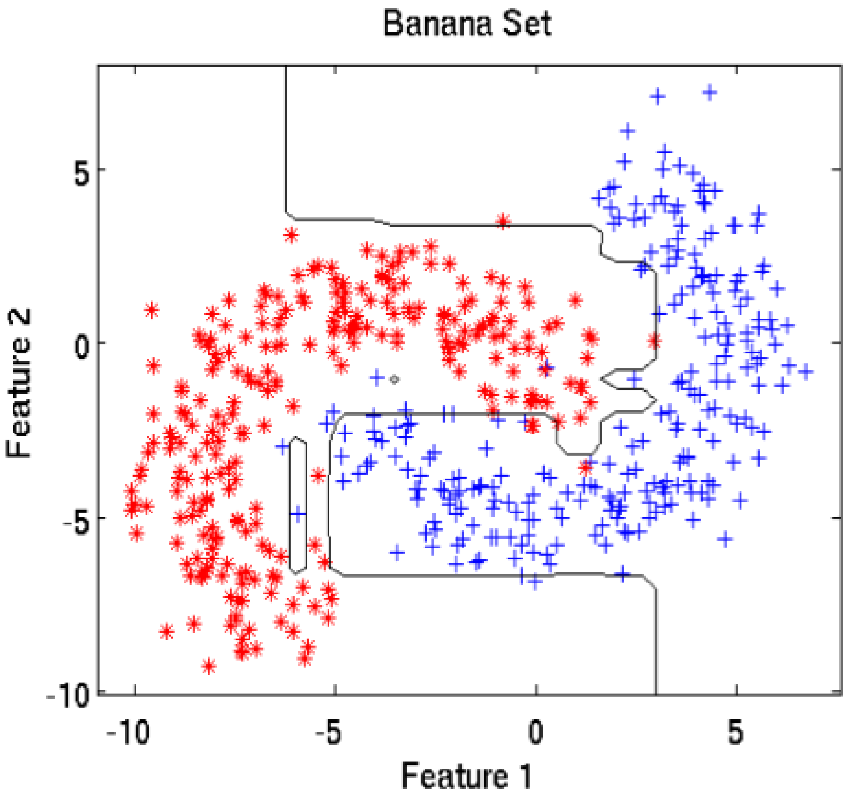


Decision boundary produced by a third tree

Bagging: Illustration

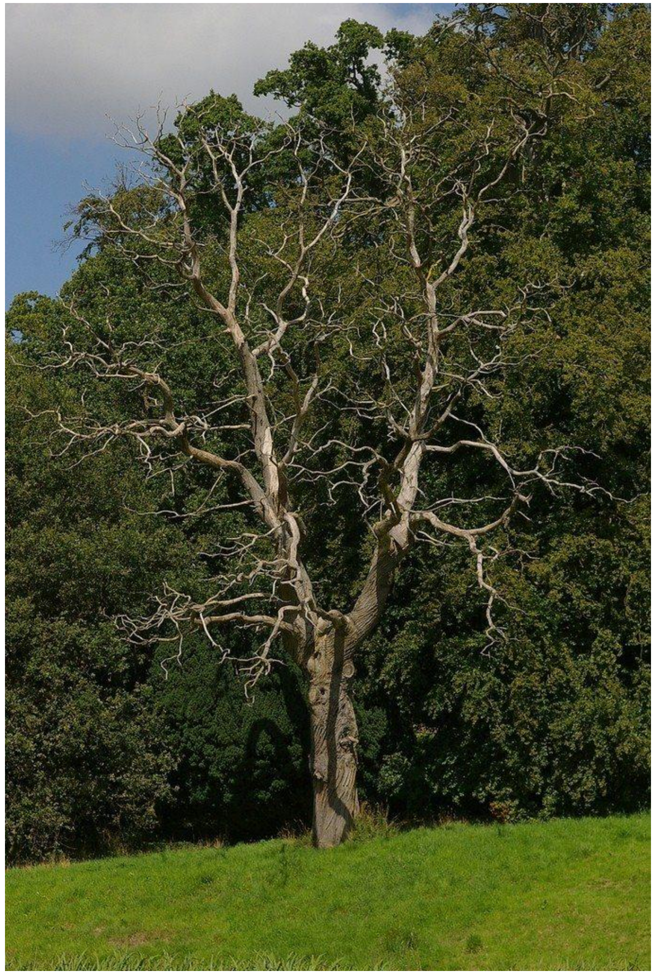


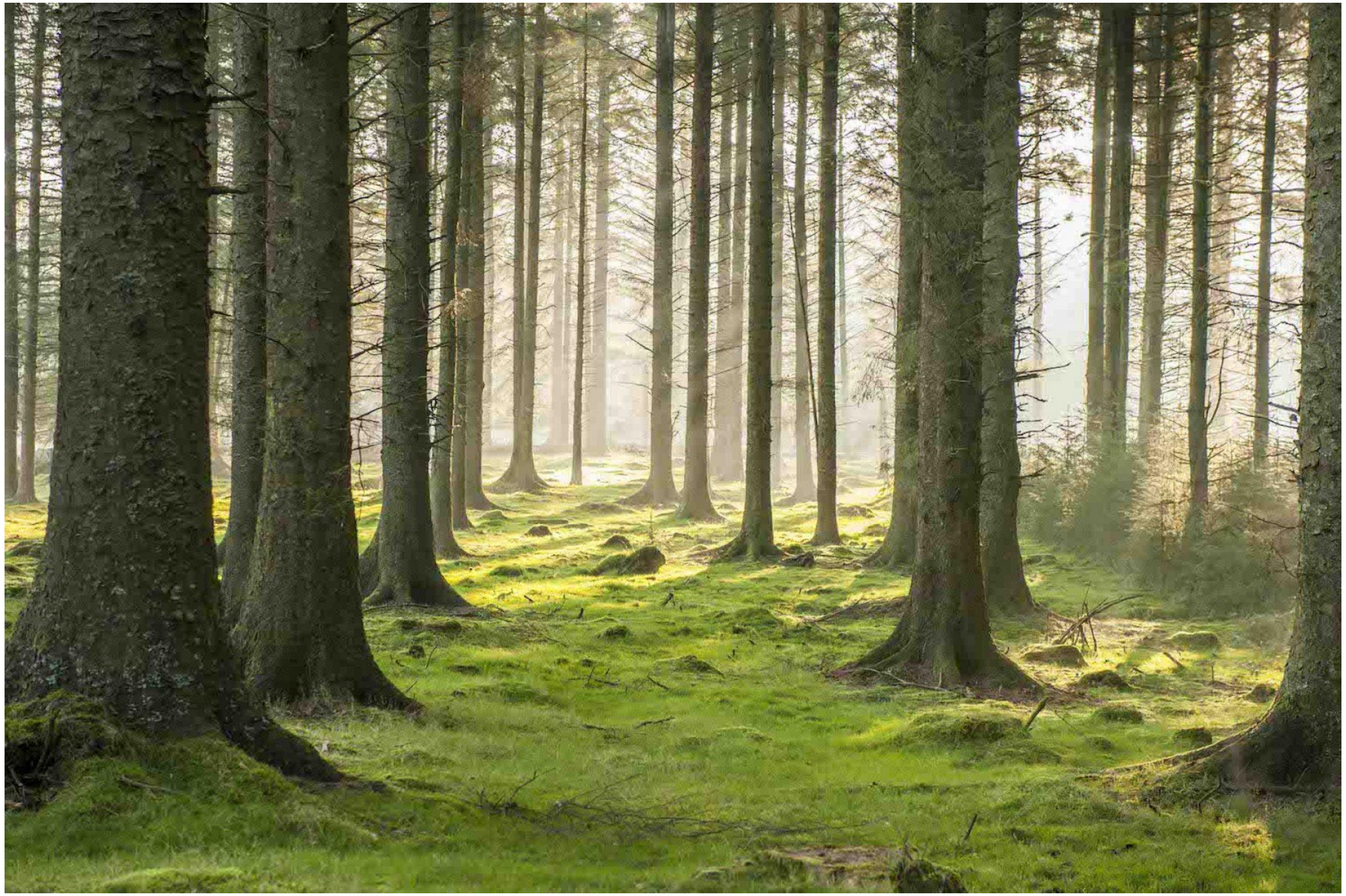
Three trees and final boundary overlaid



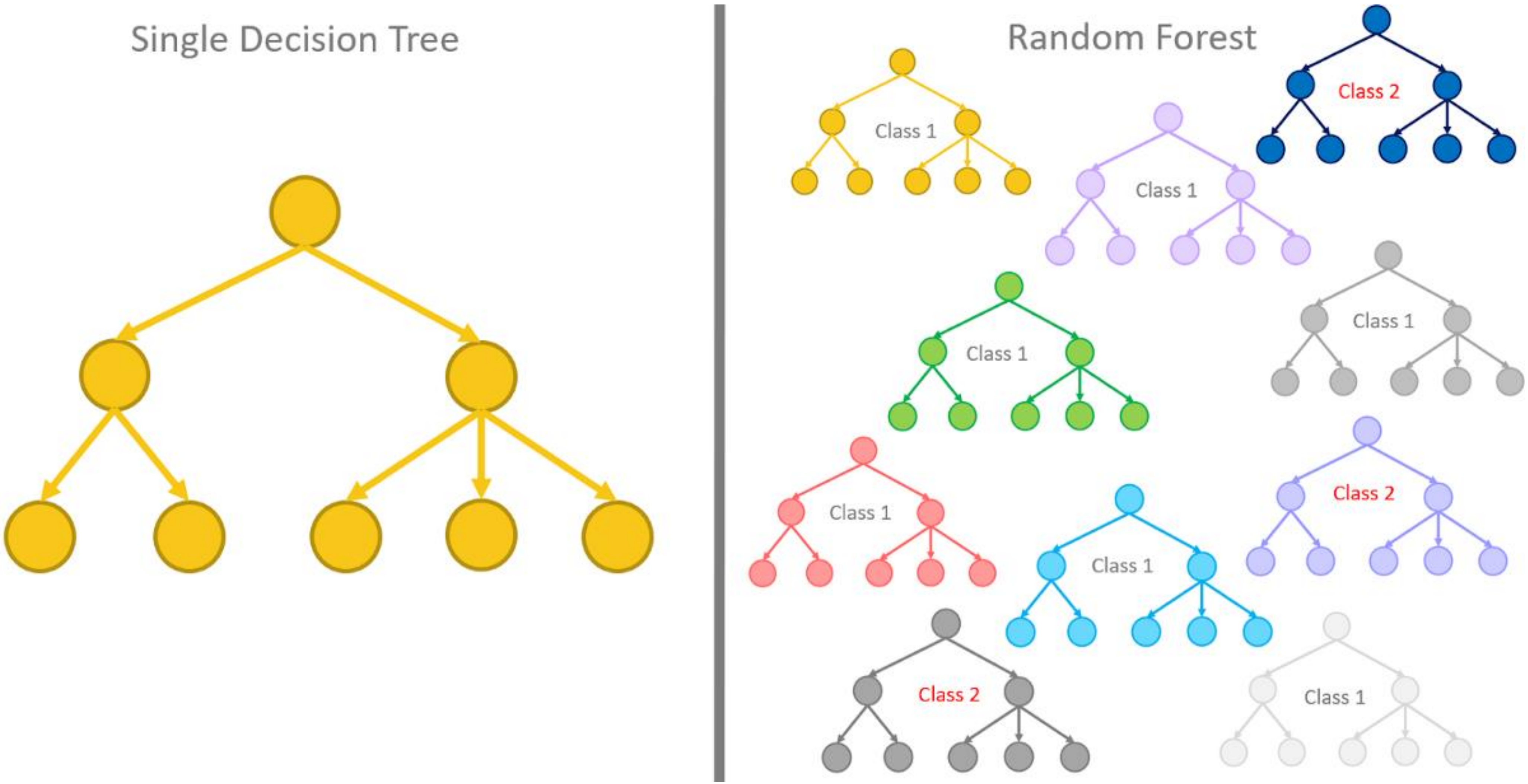
Final result from bagging all trees.

Random Forest



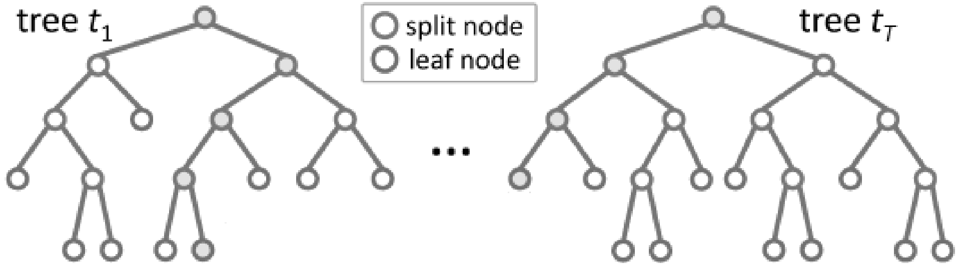


Random Forest



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.

आई आई टी हैदराबाद
IIT Hyderabad

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

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

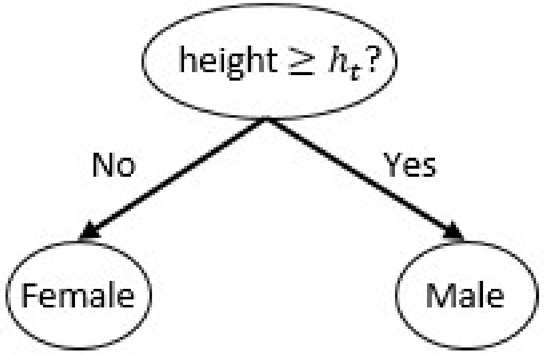
- Weak classifiers are learnt sequentially

- 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



IIT Hyderabad

Algorithm: Boosting

Input:

- Training data S of size N with correct labels ω_i, $\Omega = \{\omega_1, \omega_2\}$;
- Weak learning algorithm **WeakLearn**.

Training

1. Select $N_1 < N$ patterns without replacement from S to create data subset S_1.
2. Call **WeakLearn** and train with S_1 to create classifier C_1.
3. Create dataset S_2 as the most informative dataset, given C_1, such that half of S_2 is correctly classified by C_1, and the other half is misclassified.:
   a. Flip a fair coin. If Head, select samples from S, and present them to C_1 until the first instance is misclassified. Add this instance to S_2.
   b. If Tail, select samples from S, and present them to C_1 until the first one is correctly classified. Add this instance to S_2.
   c. Continue flipping coins until no more patterns can be added to S_2.
4. Train the second classifier C_2 with S_2.
5. Create S_3 by selecting those instances for which C_1 and C_2 disagree. Train the third classifier C_3 with S_3.

Test – Given a test instance **x**

1. Classify **x** by C_1 and C_2. If they agree on the class, this class is the final classification.
2. If they disagree, choose the class predicted by C_3 as the final classification.

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

- Arguably the best known of all ensemble-based algorithms, **AdaBoost** (Adaptive Boosting) extends boosting to multi-class and regression problems.

- Bootstrap training data samples are drawn from a distribution D that is iteratively updated such that subsequent classifiers focus on increasingly difficult instances.

Adaboost Algorithm (won the prestigious Gödel Prize.)

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

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

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

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

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

  $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j$, where $\epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$
- Prerequisite: $\varepsilon_t < 0.5$, otherwise stop.
- Choose $\alpha_t \in \mathbf{R}$, typically $\alpha_t = \frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}$ where ε_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:

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

Reweighting

Effect on the training set

Reweighting formula:

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

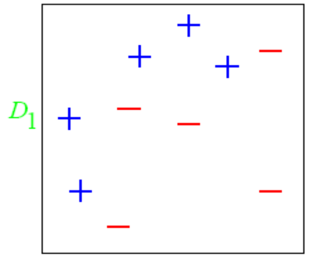
$$exp(-\alpha_t y_i h_t(x_i)) \begin{cases} < 1, & y_i = h_t(x_i) \\ > 1, & y_i \neq h_t(x_i) \end{cases}$$

y * h(x) = 1

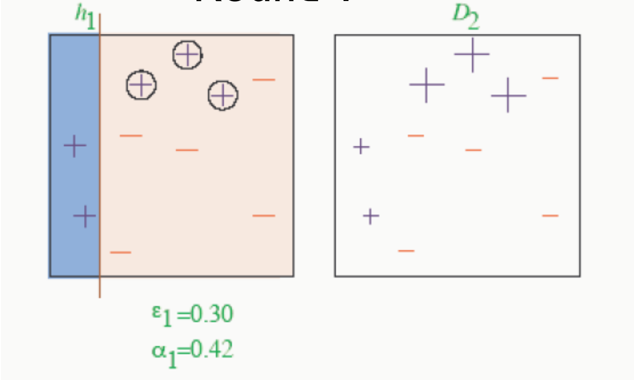
y * h(x) = -1

$\Rightarrow$ Increase (decrease) weight of wrongly (correctly) classified examples

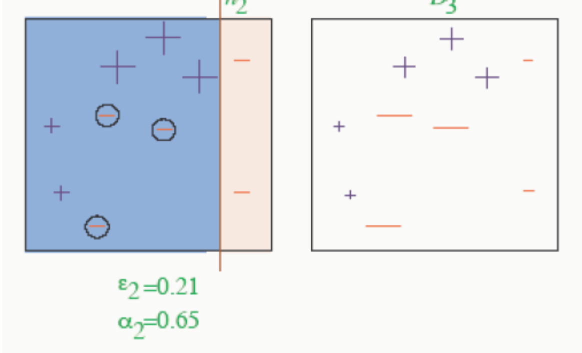
Simple Example



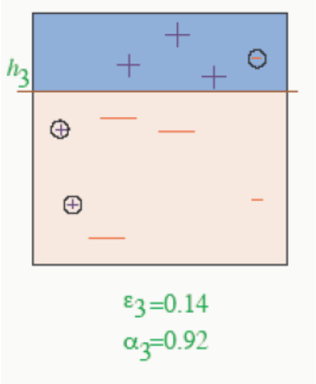
Round 1



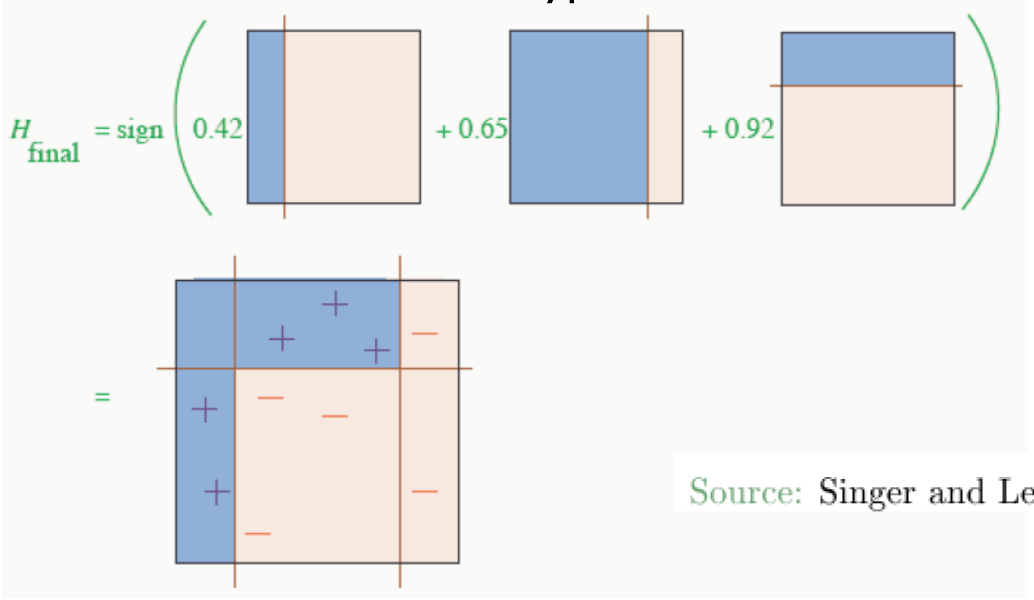
Round 2



Round 3



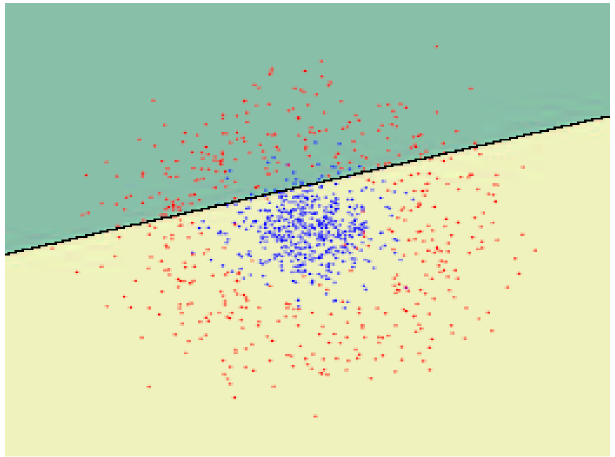
Final Hypothesis

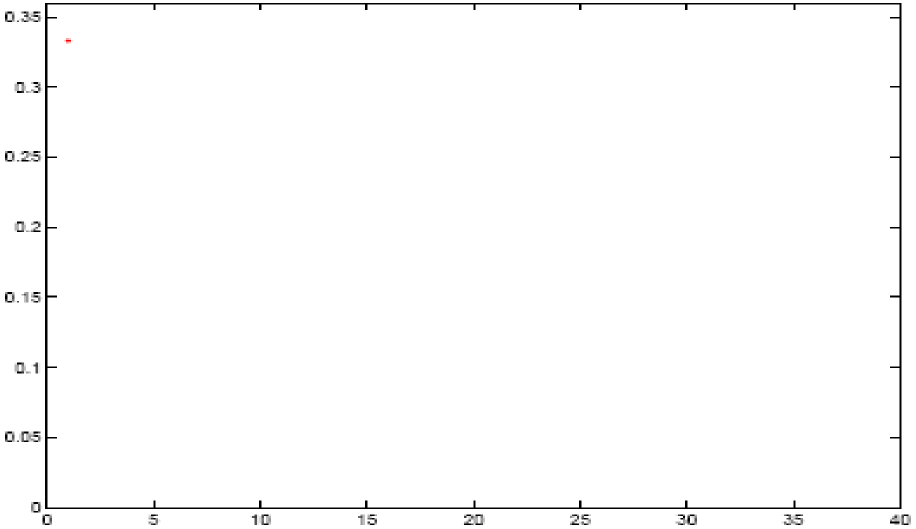


Source: Singer and Lewis Tutorial

Illustration

$t = 1$

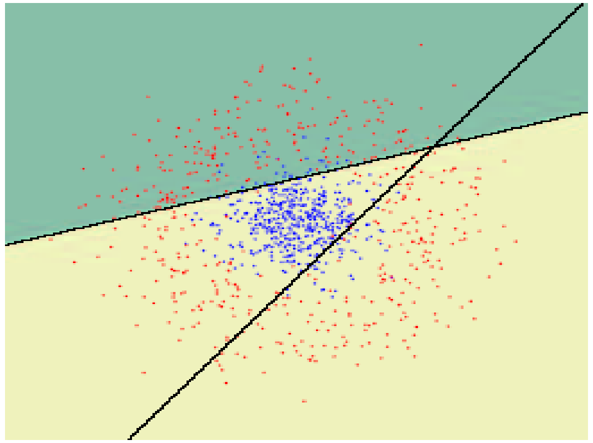


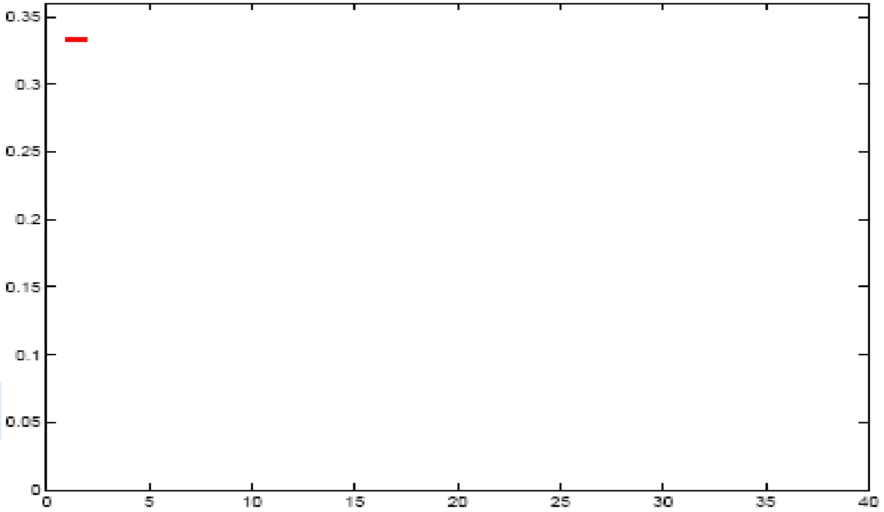


Sample plot of validation error

Illustration

$t = 2$

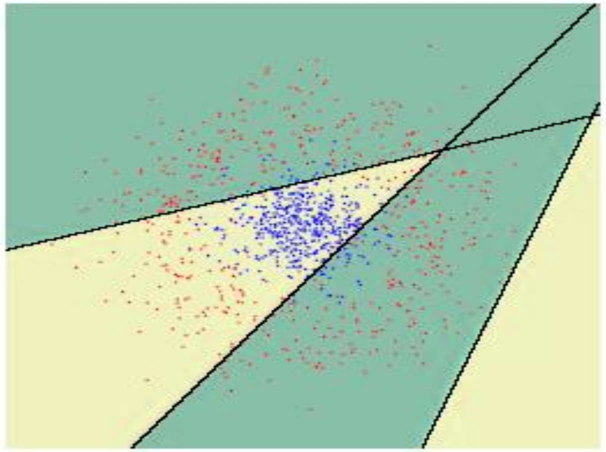


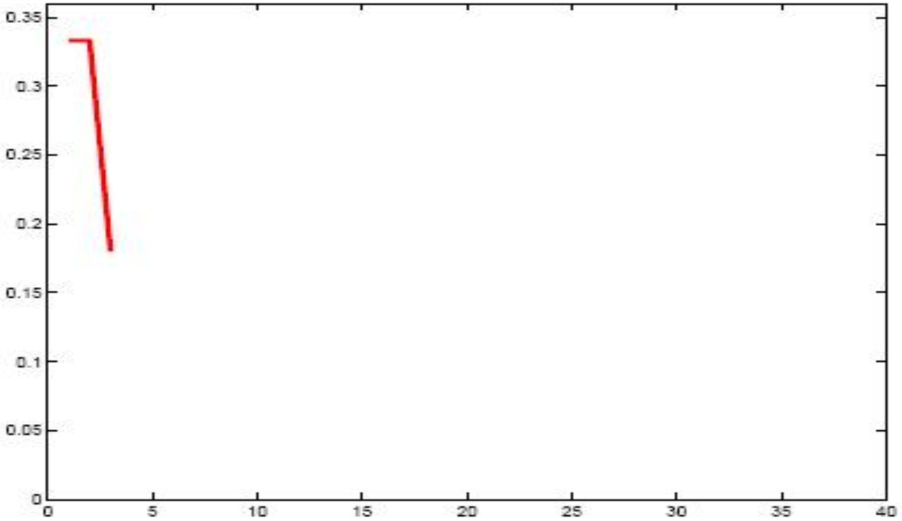


Sample plot of validation error

Illustration

$t = 3$

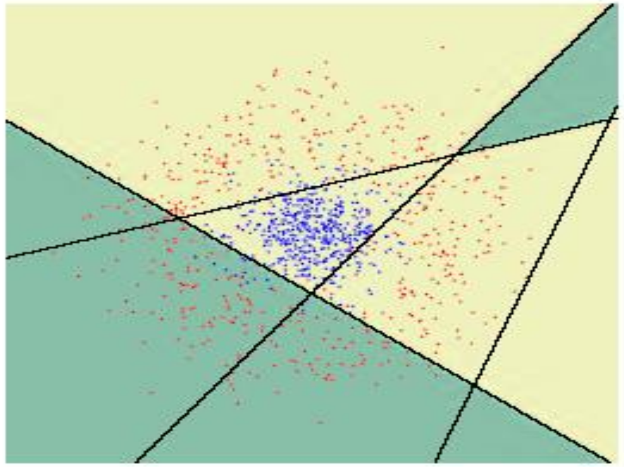




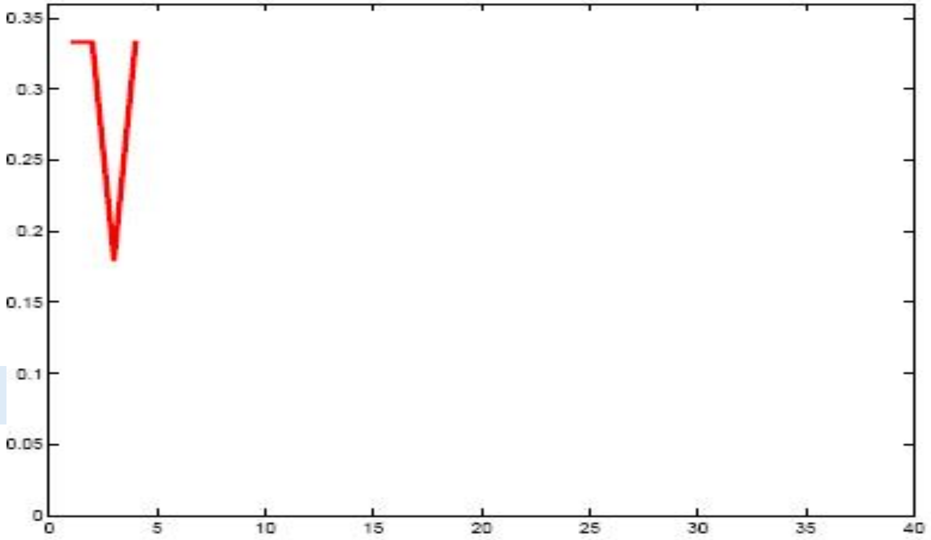
Sample plot of validation error

Illustration

$t = 4$

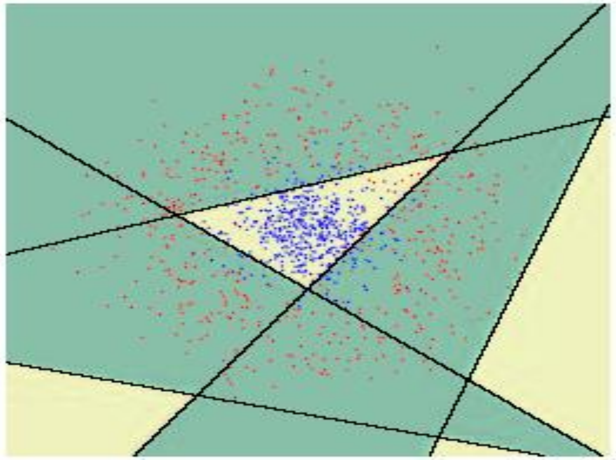


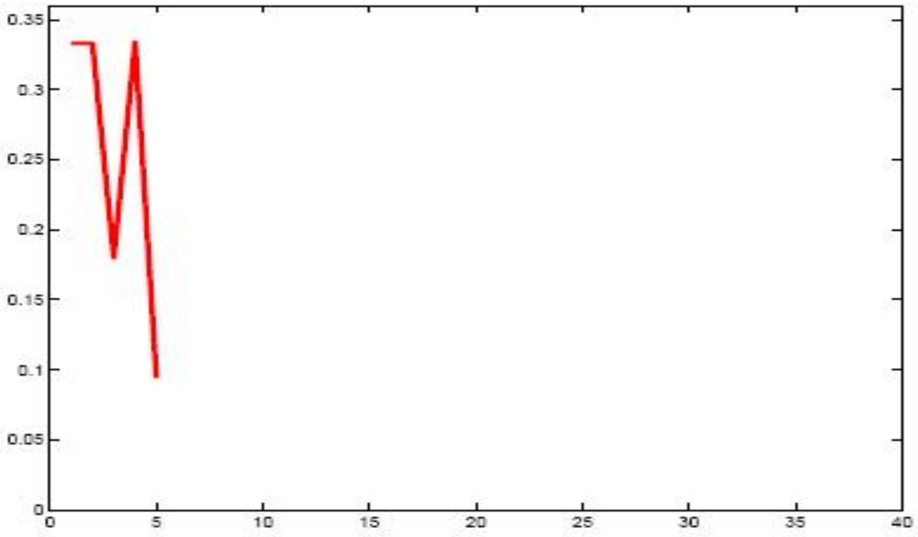
Sample plot of validation error



Illustration

$t = 5$

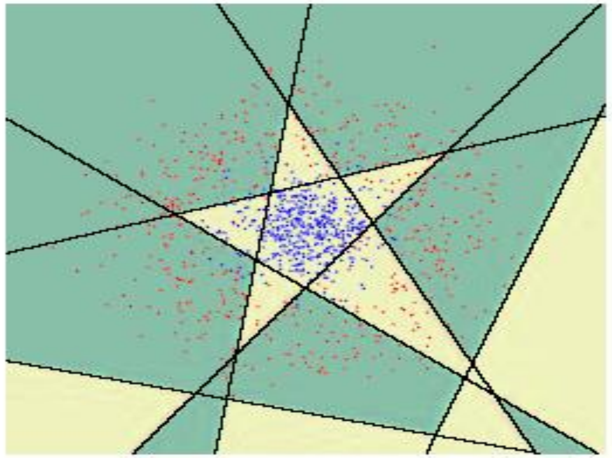




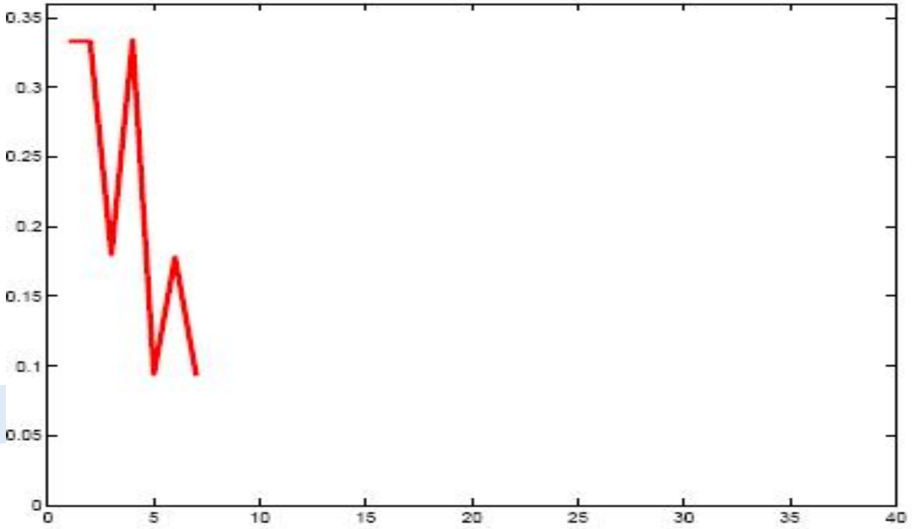
Sample plot of validation error

Illustration

$t = 7$

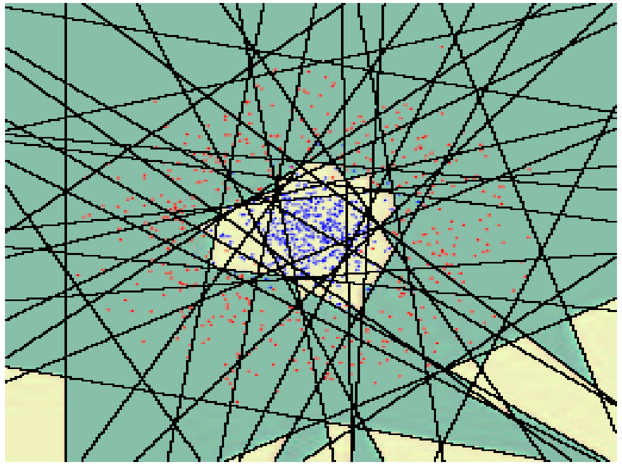


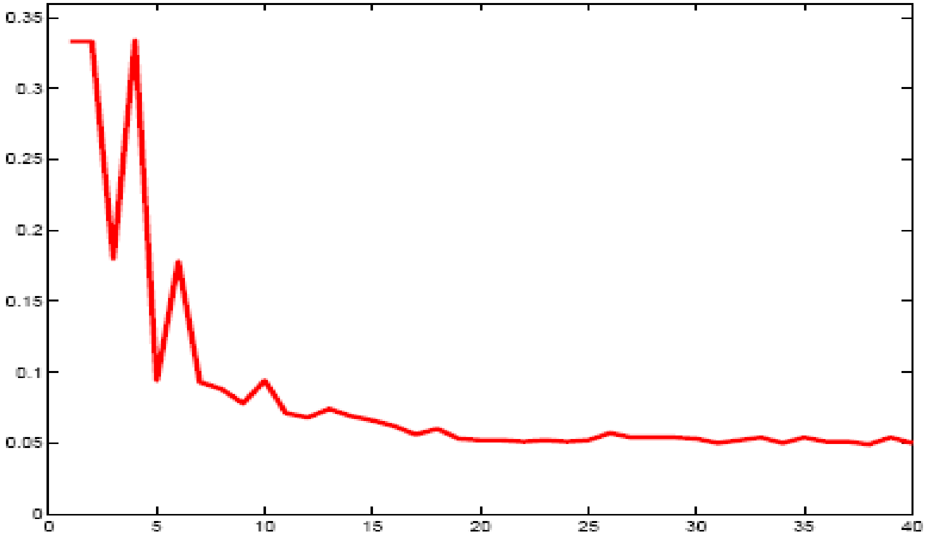
Sample plot of validation error



Illustration

$t = 40$



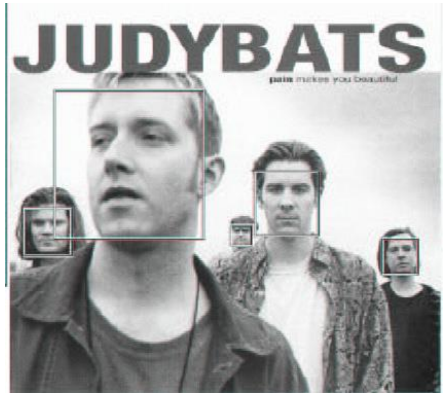


Sample plot of validation error

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

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

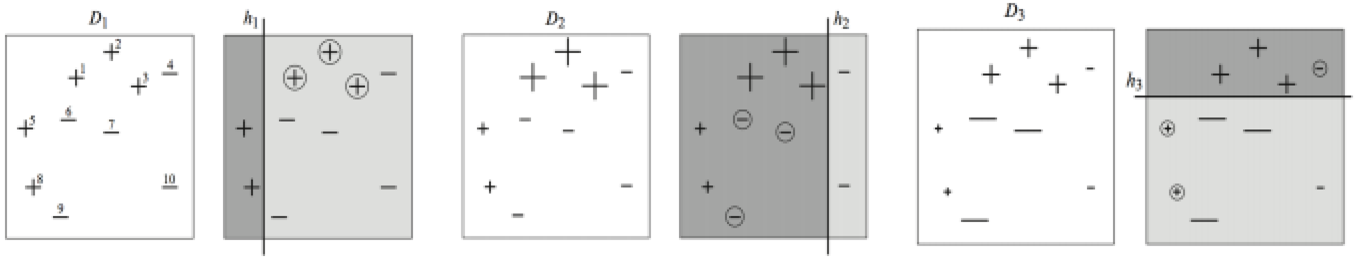
Data set	Adaboost	Forest-RI
Glass	1.6	.4
Breast cancer	43.2	1.8
Diabetes	6.8	1.7
Sonar	15.1	−6.6
Ionosphere	27.7	3.8
Soybean	26.9	3.2
Ecoli	7.5	7.9
Votes	48.9	6.3
Liver	10.3	−.2

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**
- boosting algorithms as iterative*gradient descent* - optimize a cost function over function space by iteratively choosing a function that points in the negative gradient direction
- In a regression setting, gradient boosting fits hm to the residual y-Fm.

$$F_{m+1}(x) = F_m(x) + h_m(x) = y$$

$$h_m(x) = y - F_m(x).$$

- hm proportional to the negative gradient of the MSE loss

$$L_{\mathrm{MSE}} = \frac{1}{2}(y - F(x))^2$$

$$h_m(x) = -\frac{\partial L_{\mathrm{MSE}}}{\partial F} = y - F(x).$$

Gradient Boosting

- **Gradient Boosting = Gradient Descent + Boosting**
- boosting algorithms as iterative*gradient descent* - optimize a cost function over function space by iteratively choosing a function that points in the negative gradient direction
- Generalization to arbitrary loss function
- Starts with a constant function, and incrementally expands it in a greedy fashion

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

$$F_m(x) = F_{m-1}(x) + \left(\arg\min_{h_m \in \mathcal{H}} \left[\sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h_m(x_i)) \right] \right)(x)$$

$$F_m(x) = F_{m-1}(x) - \gamma \sum_{i=1}^{n} \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i))$$

Gradient Boosting Algorithm

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:

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

2. For m = 1 to M:

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

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

    2. Fit a base learner (or weak learner, e.g. tree) closed under scaling $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^n$.
    3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

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

    4. Update the model:

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

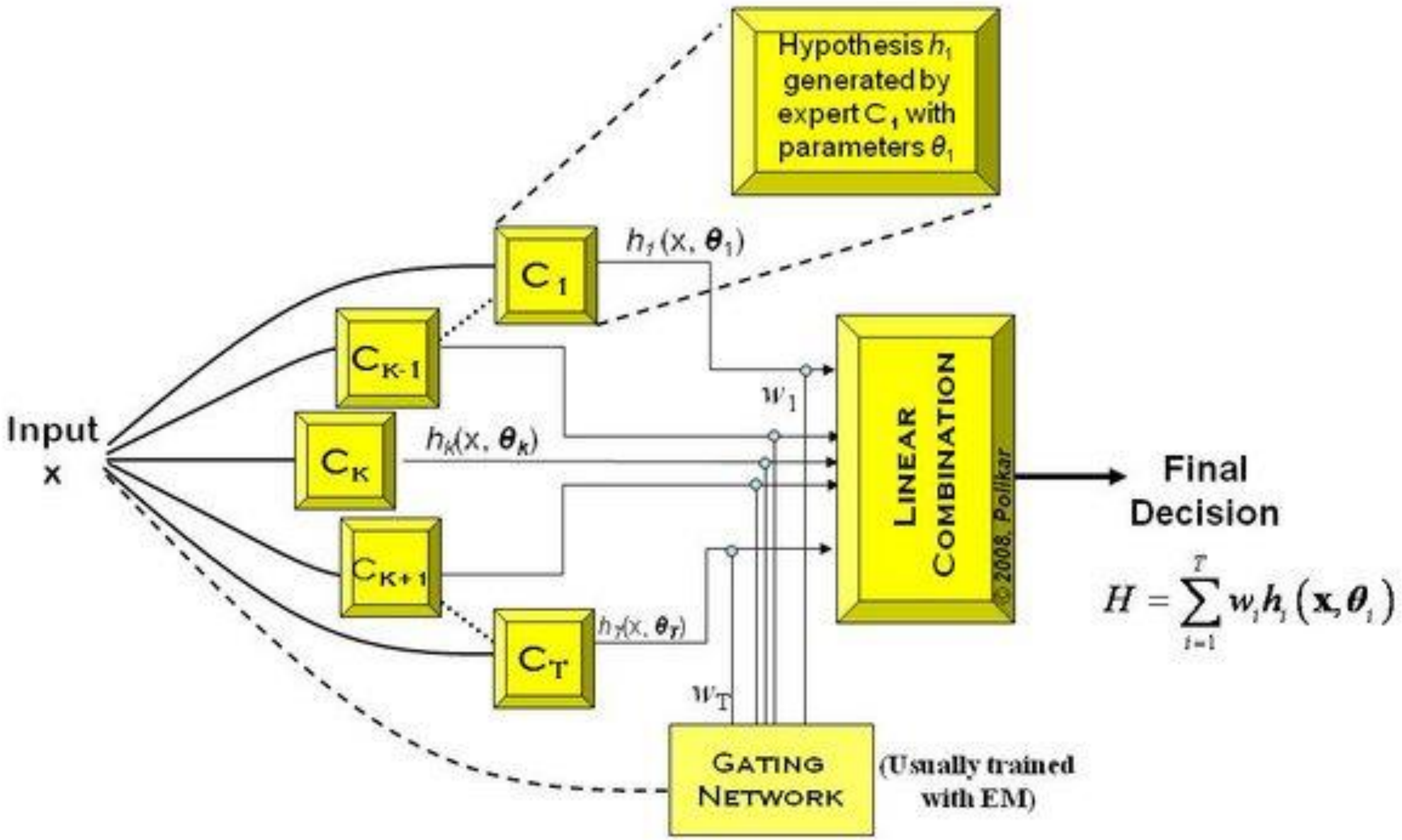
3. Output $F_M(x)$.

Mixture of Experts

outputs are combined through a (generalized) linear rule.

weights of this combination are determined by a gating network

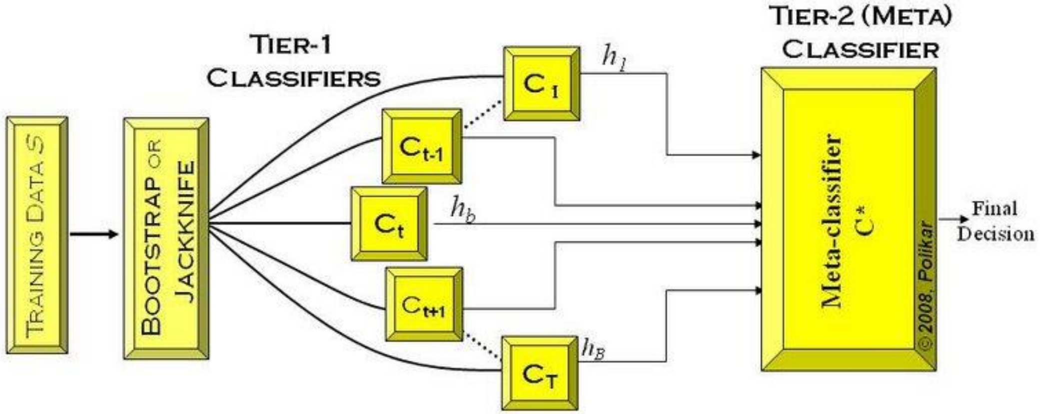
gating network requires the input instances for training.



Stacked Ensembles

- An ensemble of classifiers is first trained using bootstrapped samples of the training data, creating *Tier-1 classifiers*, whose outputs are then used to train a *Tier-2classifier (meta-classifier)*

- Tier -1 classifier is first trained on (a different set of) *T*-1 blocks of the training data. Each classifier is then evaluated on the *T*th (pseudo-test) block, not seen during training. The outputs of these classifiers on their pseudo-training blocks, along with the actual correct labels for those blocks constitute the training dataset for the Tier 2 classifier



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
- XGBoost and Gradient Boosting : *Mason, L.; Baxter, J.; Bartlett, P. L.; Frean, Marcus (1999). "Boosting Algorithms as Gradient Descent" (PDF). NeurIPS.*
- https://github.com/dmlc/xgboost