Bagging and Boosting

Amit Srinet
Dave Snyder

Outline

- Bagging
  - Definition
  - Variants
  - Examples
- Boosting
  - Definition
  - Hedge(β)
  - AdaBoost
  - Examples
- Comparison

Bagging

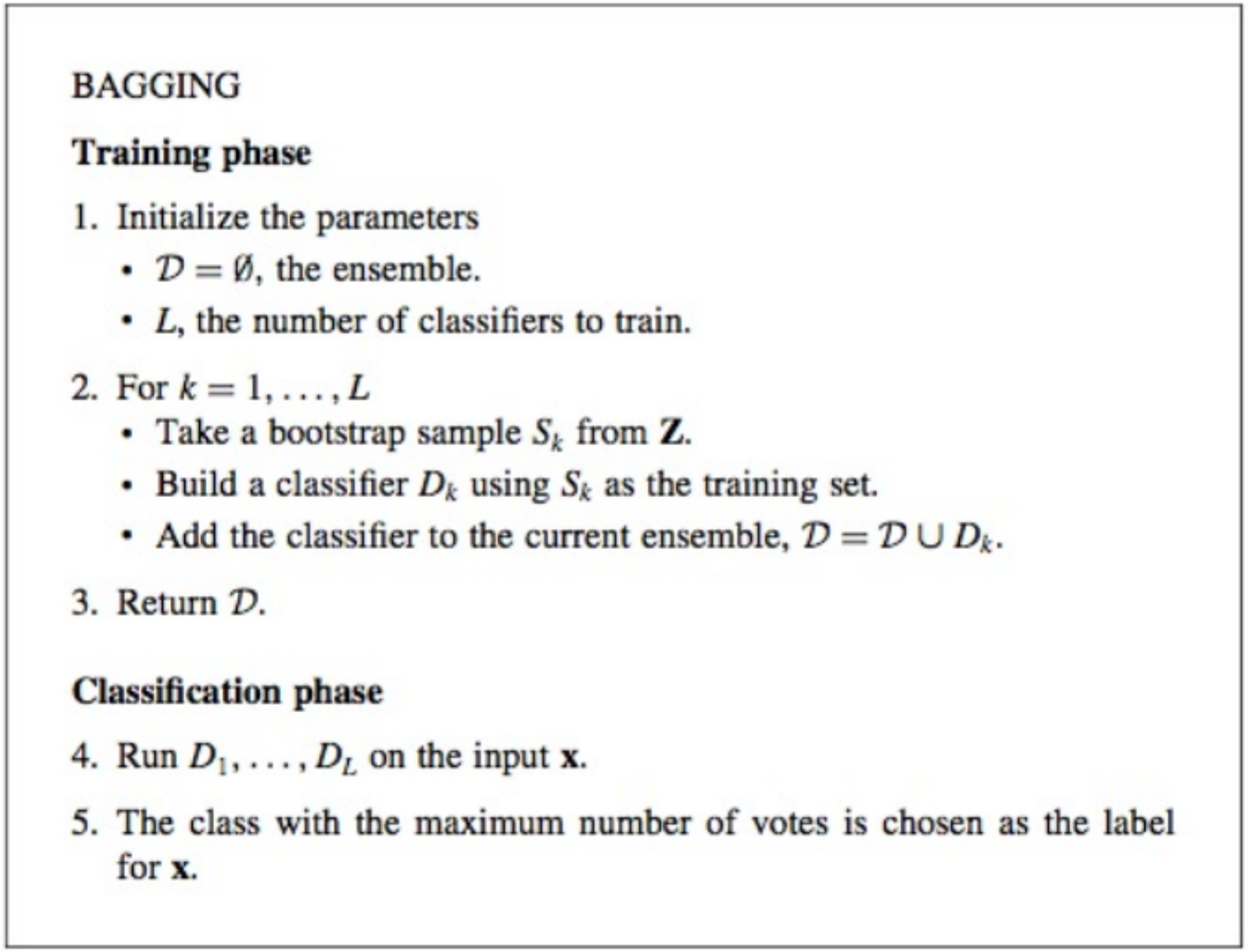
- Bootstrap Model
- Randomly generate L set of cardinality N from the original set Z with replacement.
- Corrects the optimistic bias of R-Method

- "Bootstrap Aggregation"
- Create Bootstrap samples of a training set using sampling with replacement.
- Each bootstrap sample is used to train a different component of base classifier
- Classification is done by plurality voting

Bagging

- Regression is done by averaging
- Works for unstable classifiers
    - Neural Networks
    - Decision Trees

Bagging



```
BAGGING

Training phase

1. Initialize the parameters
    • $\mathcal{D} = \emptyset$, the ensemble.
    • $L$, the number of classifiers to train.

2. For $k = 1, \ldots, L$
    • Take a bootstrap sample $S_k$ from $\mathbf{Z}$.
    • Build a classifier $D_k$ using $S_k$ as the training set.
    • Add the classifier to the current ensemble, $\mathcal{D} = \mathcal{D} \cup D_k$.

3. Return $\mathcal{D}$.

Classification phase

4. Run $D_1, \ldots, D_L$ on the input $\mathbf{x}$.

5. The class with the maximum number of votes is chosen as the label
   for $\mathbf{x}$.
```

Fig. 7.1 *The bagging algorithm.*

Kuncheva

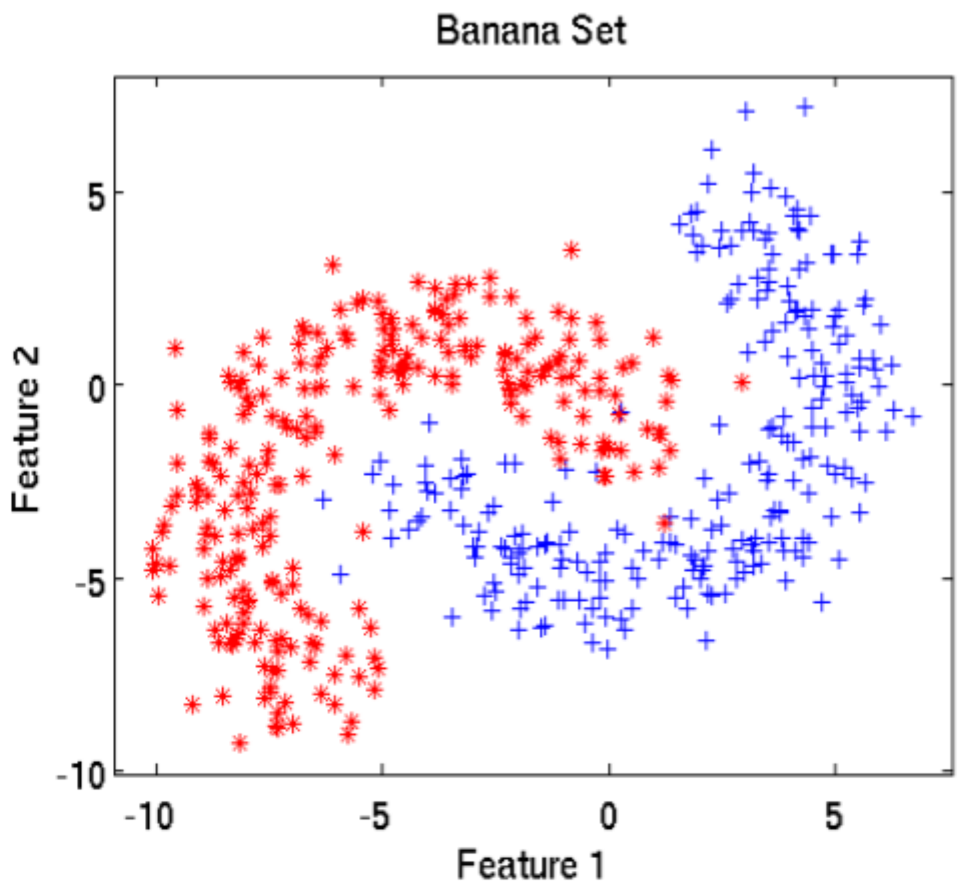
Example

- PR Tools:

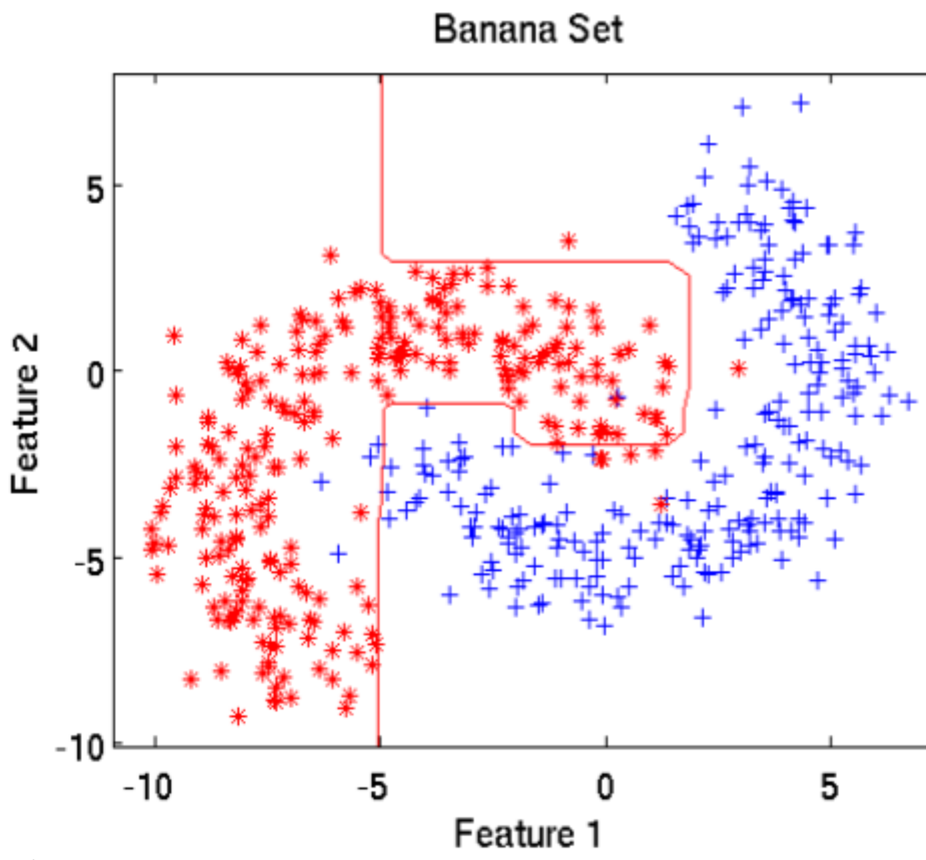
```
>> A = gendatb(500,1);
>> scatterd(A)
>> W1 = baggingc(A,treec,100,[],[]);
>> plotc(W1(:,1:2),'r')
>> W2 = baggingc(A,treec,100,treec,[]);
>> plotc(W2)
```

- Generates 100 trees with default settings - stop based on purity metric, zero pruning

Example



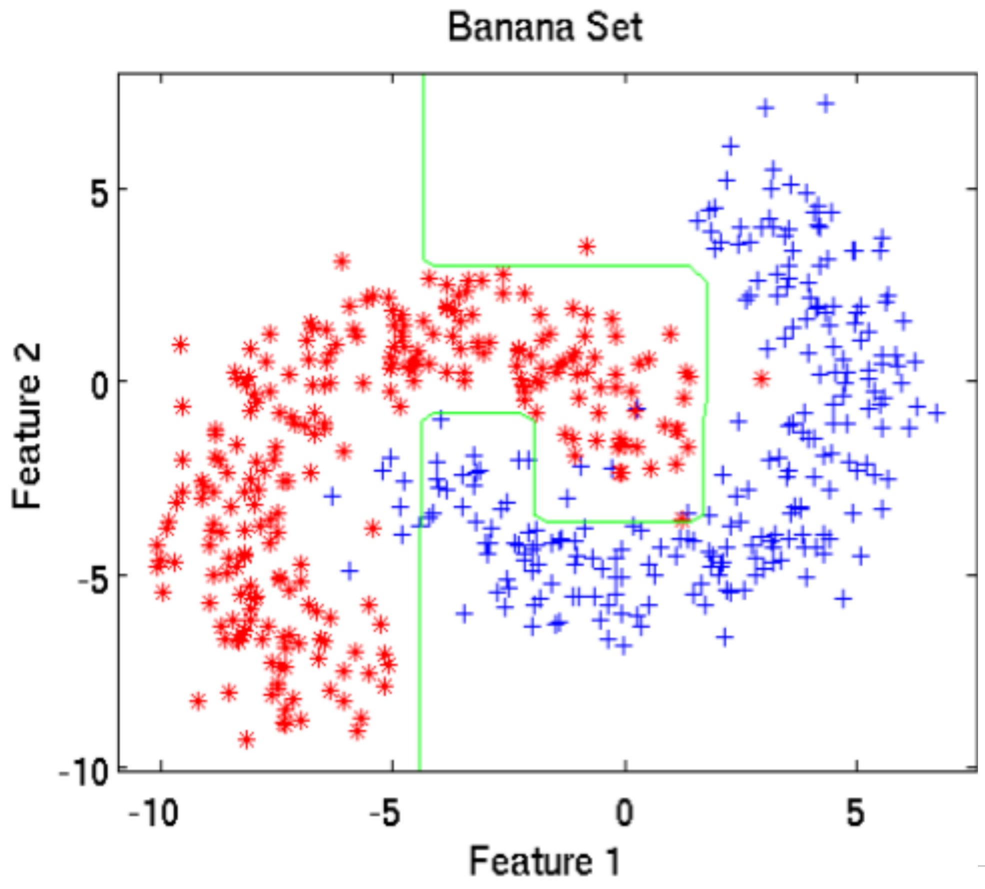
Training data



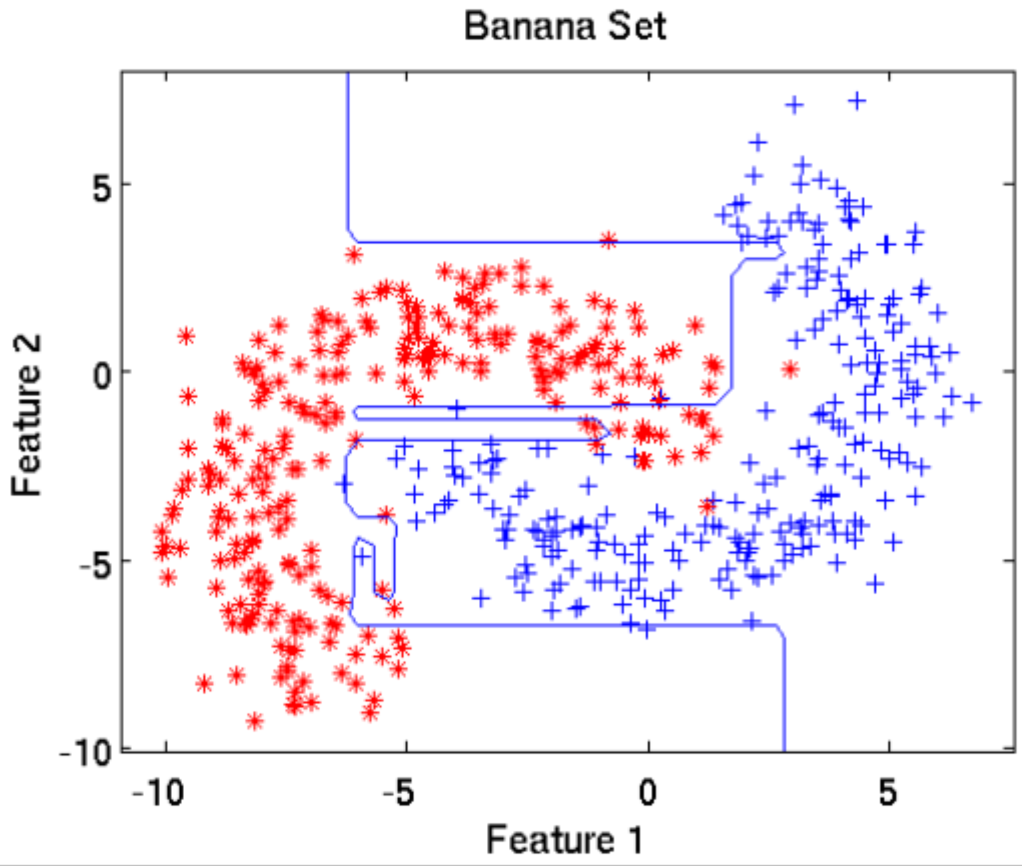
Decision boundary produced by one tree

Bagging: Decision Tree

Example



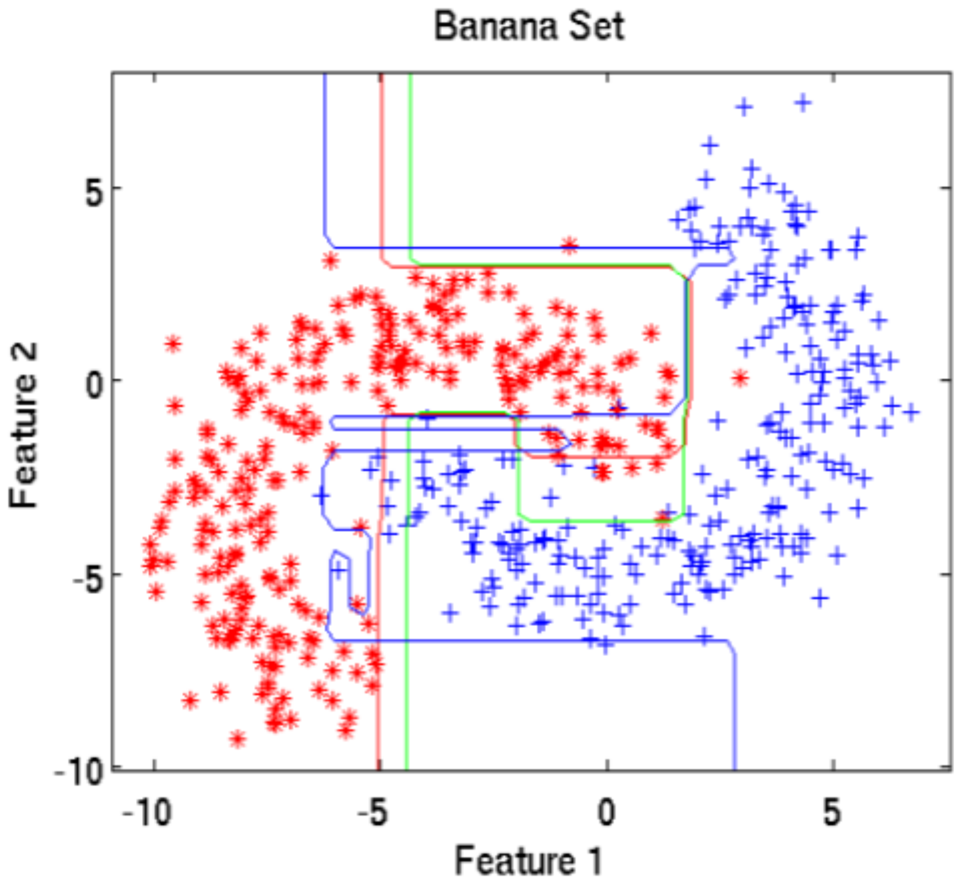
Decision boundary produced by a second tree



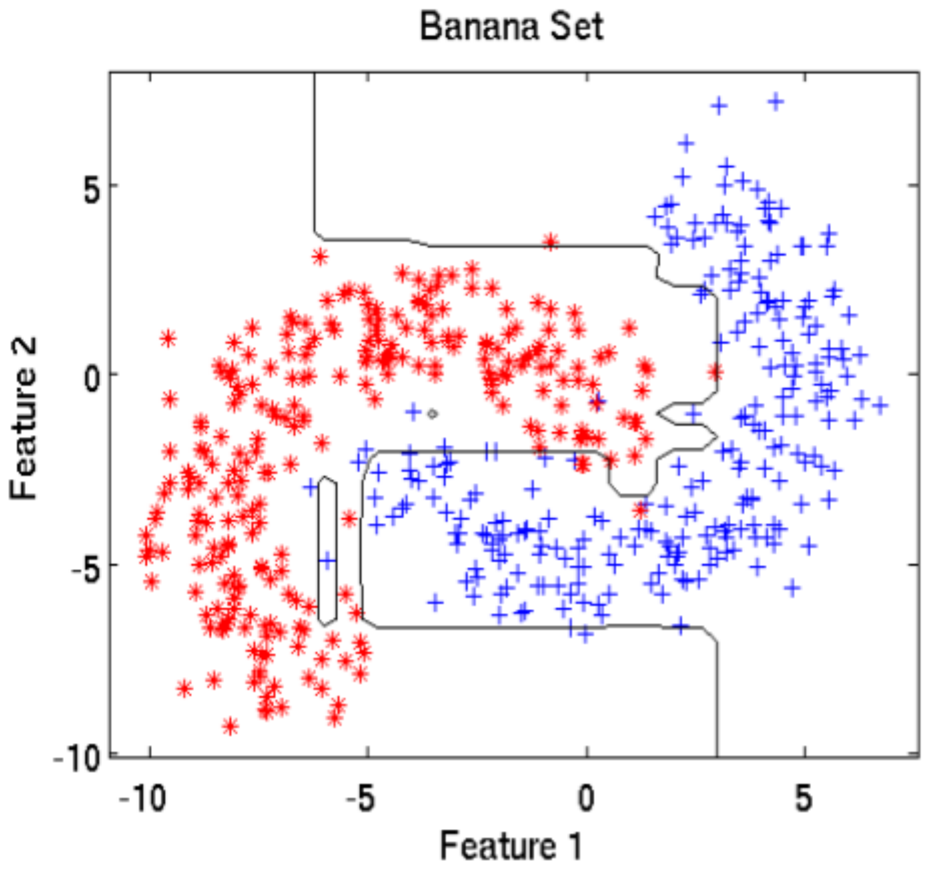
Decision boundary produced by a third tree

Bagging: Decision Tree

Example



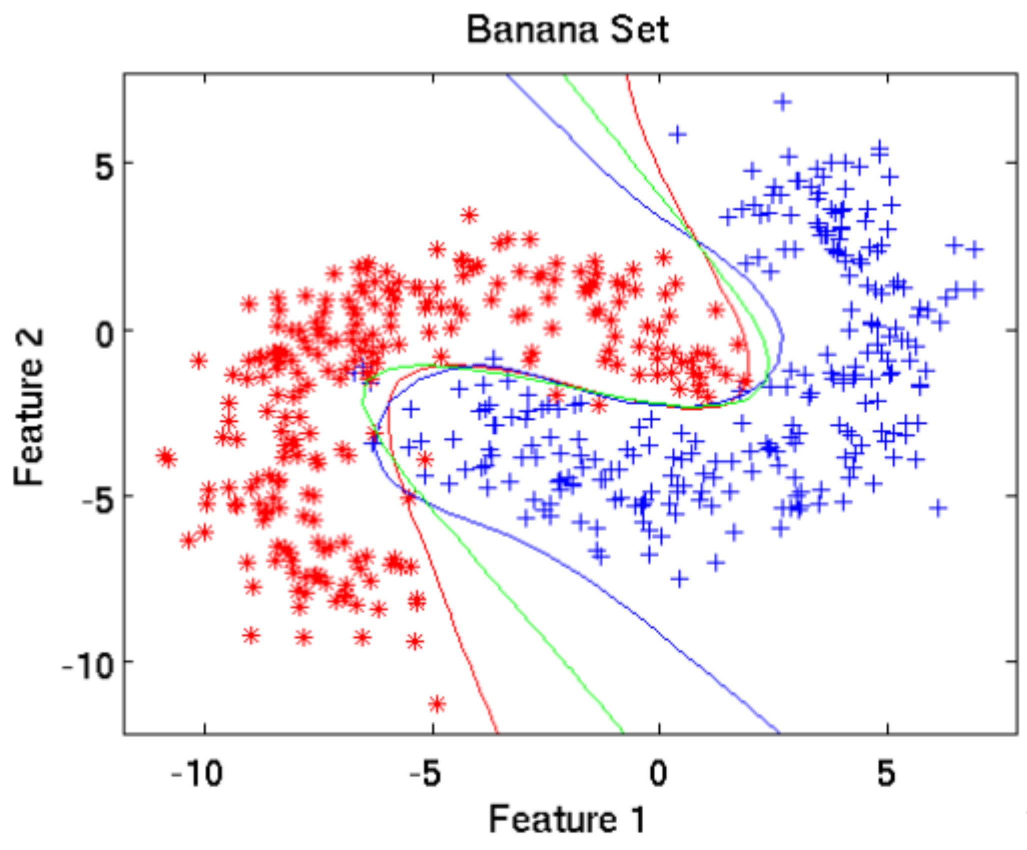
Three trees and final boundary overlaid



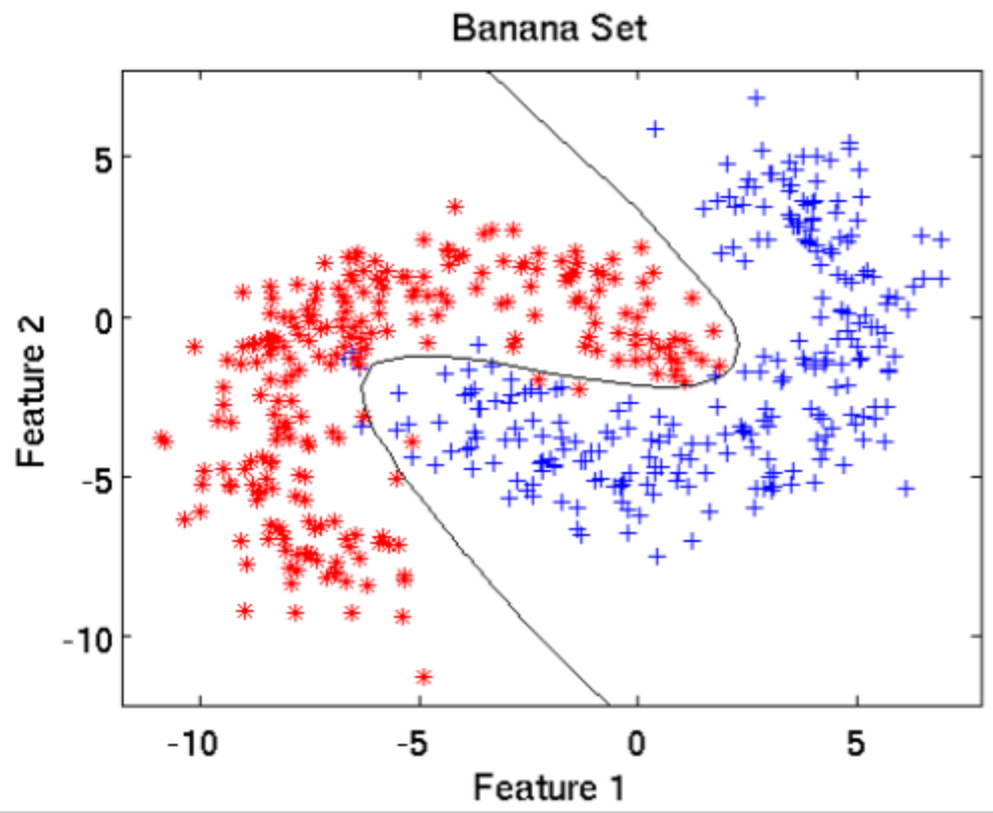
Final result from bagging all trees.

Bagging: Decision Tree

Example



Three neural nets generated with default settings [bpxnc]



Final output from bagging 10 neural nets

Bagging: Neural Net

Why does bagging work ?

- Main reason for error in learning is due to noise ,bias and variance.
- Noise is error by the target function
- Bias is where the algorithm can not learn the target.
- Variance comes from the sampling, and how it affects the learning algorithm
- Does bagging minimizes these errors ?
- Yes
- Averaging over bootstrap samples can reduce error from variance especially in case of unstable classifiers

Bagging

- In fact Ensemble reduces variance
- Let f(x) be the target value of x and h1 to hn be the set of base hypotheses and h-average be the prediction of base hypotheses
- E(h,x) = (f(x) – h(x))^2 Squared Error

Ensemble Reduces variance

- Let f(x) be the target value for x.
- Let h1, . . . , hn be the base hypotheses.
- Let h-avg be the average prediction of h1, . . . , hn.
- Let E(h, x) = (f(x) − h(x))2
- Is there any relation between h-avg and variance?
    - yes

- E(h-avg,x) = ∑(i = 1 to n)E(hi ,x)/n
- ∑(i = 1 to n) (hi(x) – h-avg(x))^2/n

That is squared error of the average prediction equals the average squared error of the base hypotheses minus the variance of the base hypotheses.

Reference – 1-End of the slideshow.

Bagging - Variants

- Random Forests
  - A variant of bagging proposed by Breiman
  - It’s a general class of ensemble building methods using a decision tree as base classifier.
- Classifier consisting of a collection of tree-structure classifiers.
- Each tree grown with a random vector Vk where k = 1,…L are independent and statistically distributed.
- Each tree cast a unit vote for the most popular class at input x.

Boosting

- □A technique for combining multiple base classifiers whose combined performance is significantly better than that of any of the base classifiers.
- Sequential training of weak learners
    - Each base classifier is trained on data that is weighted based on the performance of the previous classifier
- Each classifier votes to obtain a final outcome

Boosting

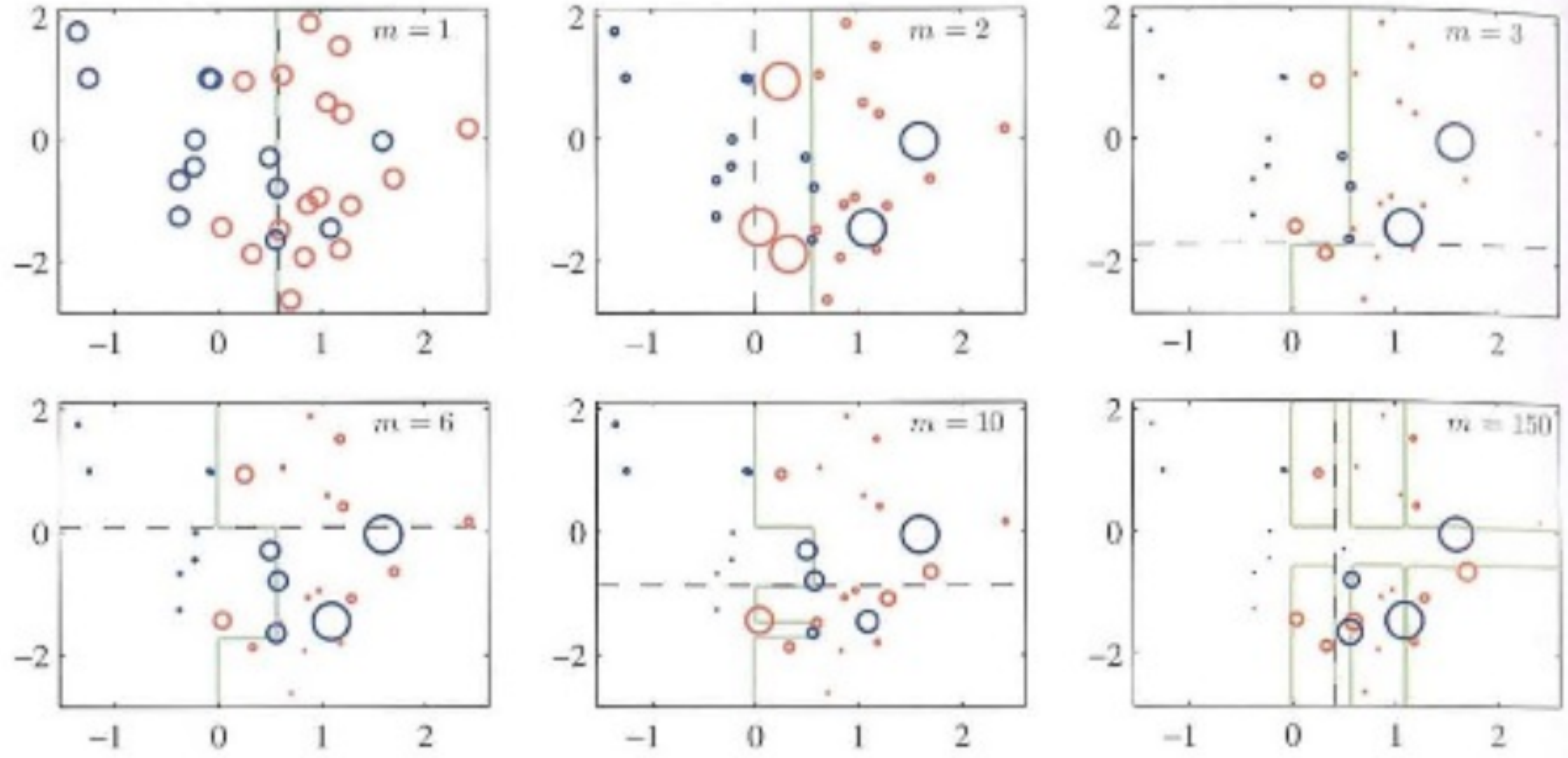


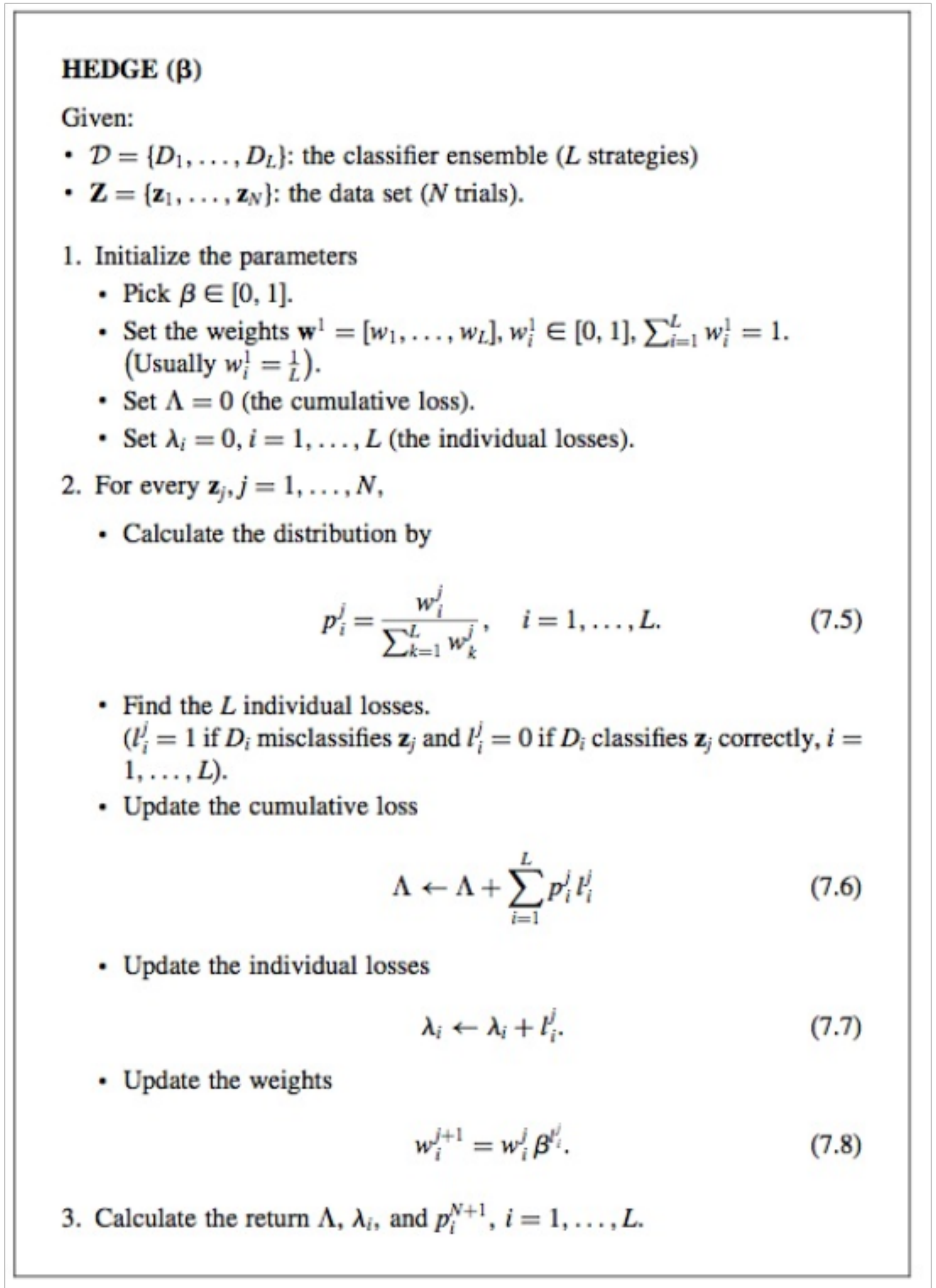
Figure 14.2 Illustration of boosting in which the base learners consist of simple thresholds applied to one or other of the axes. Each figure shows the number m of base learners trained so far, along with the decision boundary of the most recent base learner (dashed black line) and the combined decision boundary of the ensemble (solid green line). Each data point is depicted by a circle whose radius indicates the weight assigned to that data point when training the most recently added base learner. Thus, for instance, we see that points that are misclassified by the $m = 1$ base learner are given greater weight when training the $m = 2$ base learner.

Duda, Hart, and Stork

Boosting - Hedge(β)

- Boosting follows the model of online algorithm.
- Algorithm allocates weights to a set of strategies and used to predict the outcome of the certain event
- After each prediction the weights are redistributed.
- Correct strategies receive more weights while the weights of the incorrect strategies are reduced further.
- Relation with Boosting algorithm.
- Strategies corresponds to classifiers in the ensemble and the event will correspond to assigning a label to sample drawn randomly from the input.

Boosting



HEDGE (β)

Given:

- $\mathcal{D} = \{D_1, \dots, D_L\}$: the classifier ensemble (L strategies)
- $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_N\}$: the data set ($N$ trials).

1. Initialize the parameters
   - Pick $\beta \in [0, 1]$.
   - Set the weights $\mathbf{w}^1 = [w_1, \dots, w_L]$, $w_i^1 \in [0, 1]$, $\sum_{i=1}^{L} w_i^1 = 1$. $\left(\text{Usually } w_i^1 = \frac{1}{L}\right)$.
   - Set $\Lambda = 0$ (the cumulative loss).
   - Set $\lambda_i = 0$, $i = 1, \dots, L$ (the individual losses).
2. For every $\mathbf{z}_j, j = 1, \dots, N$,
   - Calculate the distribution by

$$p_i^j = \frac{w_i^j}{\sum_{k=1}^{L} w_k^j}, \quad i = 1, \dots, L. \tag{7.5}$$

   - Find the L individual losses.
     ($l_i^j = 1$ if D_i misclassifies $\mathbf{z}_j$ and $l_i^j = 0$ if D_i classifies $\mathbf{z}_j$ correctly, $i = 1, \dots, L$).
   - Update the cumulative loss

$$\Lambda \leftarrow \Lambda + \sum_{i=1}^{L} p_i^j l_i^j \tag{7.6}$$

   - Update the individual losses

$$\lambda_i \leftarrow \lambda_i + l_i^j. \tag{7.7}$$

   - Update the weights

$$w_i^{j+1} = w_i^j \beta^{l_i^j}. \tag{7.8}$$

3. Calculate the return Λ, λ_i, and p_i^{N+1}, $i = 1, \dots, L$.

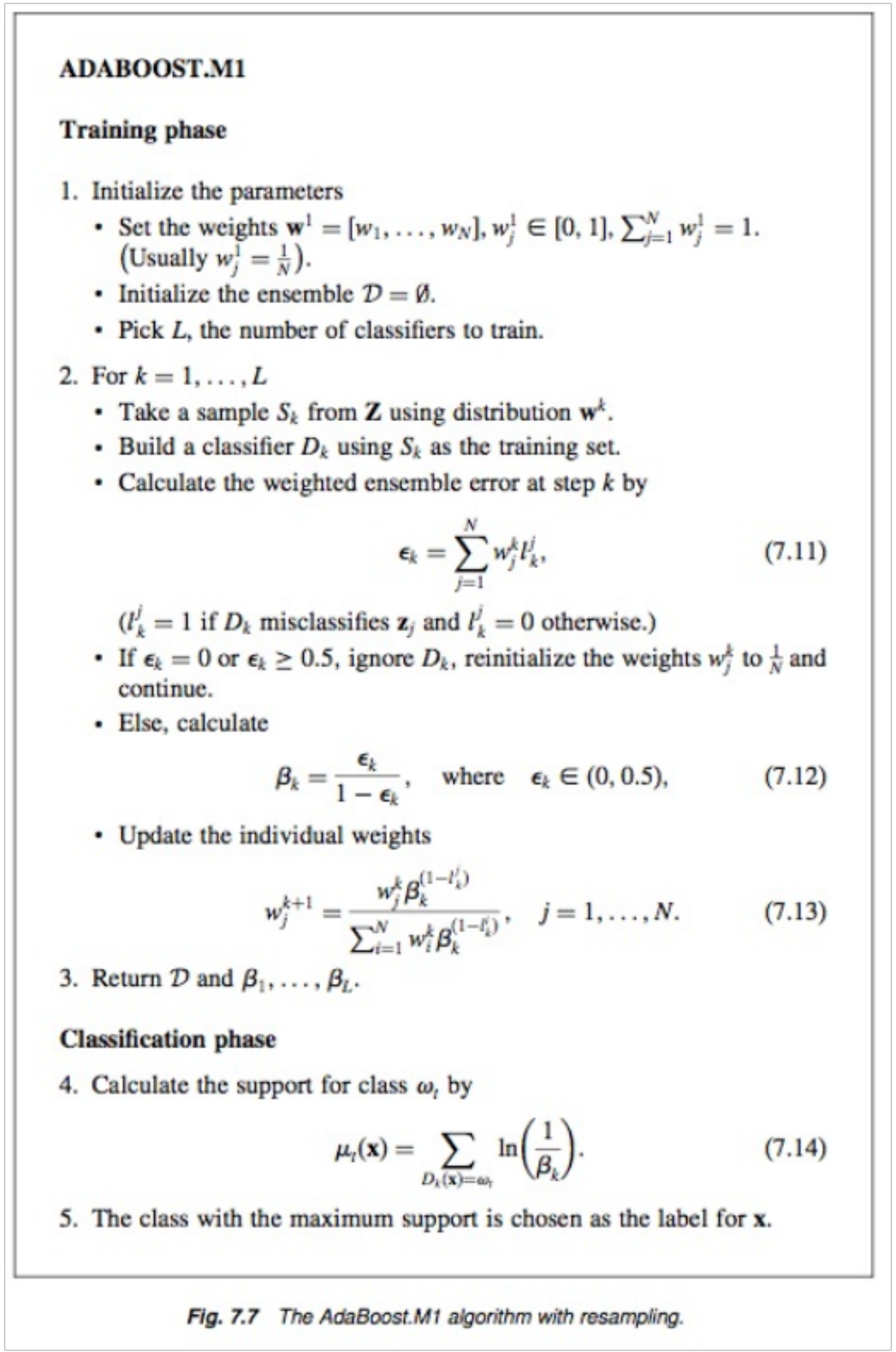
Fig. 7.5 *Algorithm Hedge(β).*

Kuncheva

Boosting - AdaBoost

- Start with equally weighted data, apply first classifier
- Increase weights on misclassified data, apply second classifier
- Continue emphasizing misclassified data to subsequent classifiers until all classifiers have been trained

Boosting



ADABOOST.M1

Training phase

1. Initialize the parameters
   - Set the weights $\mathbf{w}^1 = [w_1, \dots, w_N]$, $w_j^1 \in [0, 1]$, $\sum_{j=1}^{N} w_j^1 = 1$. $\left(\text{Usually } w_j^1 = \frac{1}{N}\right)$.
   - Initialize the ensemble $\mathcal{D} = \emptyset$.
   - Pick L, the number of classifiers to train.
2. For $k = 1, \dots, L$
   - Take a sample S_k from $\mathbf{Z}$ using distribution $\mathbf{w}^k$.
   - Build a classifier D_k using S_k as the training set.
   - Calculate the weighted ensemble error at step k by

     $$\epsilon_k = \sum_{j=1}^{N} w_j^k l_k^j, \tag{7.11}$$

     ($l_k^j = 1$ if D_k misclassifies $\mathbf{z}_j$ and $l_k^j = 0$ otherwise.)
   - If $\epsilon_k = 0$ or $\epsilon_k \geq 0.5$, ignore D_k, reinitialize the weights w_j^k to $\frac{1}{N}$ and continue.
   - Else, calculate

     $$\beta_k = \frac{\epsilon_k}{1 - \epsilon_k}, \quad \text{where} \quad \epsilon_k \in (0, 0.5), \tag{7.12}$$

   - Update the individual weights

     $$w_j^{k+1} = \frac{w_j^k \beta_k^{(1-l_k^j)}}{\sum_{i=1}^{N} w_i^k \beta_k^{(1-l_k^i)}}, \quad j = 1, \dots, N. \tag{7.13}$$

3. Return $\mathcal{D}$ and $\beta_1, \dots, \beta_L$.

Classification phase

4. Calculate the support for class ω_t by

   $$\mu_t(\mathbf{x}) = \sum_{D_k(\mathbf{x}) = \omega_t} \ln\left(\frac{1}{\beta_k}\right). \tag{7.14}$$

5. The class with the maximum support is chosen as the label for $\mathbf{x}$.

Fig. 7.7 The AdaBoost.M1 algorithm with resampling.

Kuncheva

Boosting - AdaBoost

- Training error: Kuncheva 7.2.4
- In practice overfitting rarely occurs (Bishop)

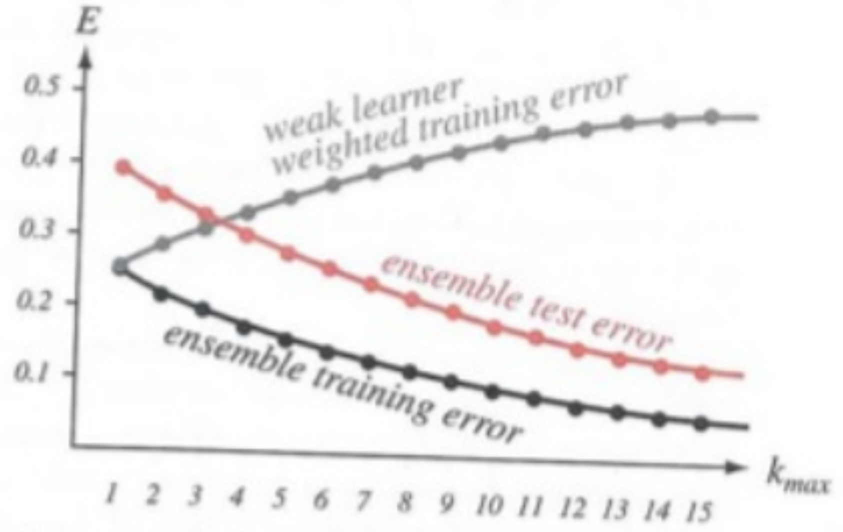


FIGURE 9.7. AdaBoost applied to a weak learning system can reduce the training error E exponentially as the number of component classifiers, k_{max}, is increased. Because AdaBoost "focuses on" *difficult* training patterns, the training error of each successive component classifier (measured on its own weighted training set) is generally larger than that of any previous component classifier (shown in gray). Nevertheless, so long as the component classifiers perform better than chance (e.g., have error less than 0.5 on a two-category problem), the weighted ensemble decision of Eq. 36 ensures that the training error will decrease, as given by Eq. 37. It is often found that the test error decreases in boosted systems as well, as shown in red.

Bishop

Margin Theory

- Testing error continues to decrease
- Ada-boost brought forward margin theory
- Margin for an object is related to certainty of its classification.
- Positive and large margin – correct classification
- Negative margin - Incorrect Classification
- Very small margin – Uncertainty in classification

- Similar classifier can give different label to an input.
- Margin of object x is calculated using the degree of support.

$$m(x) = \mu_k(x) - \max_{j \neq k}\{\mu_j(x)\}$$

Where $$\sum_{j=1}^{c} \mu_j(x) = 1$$

- Freund and schapire proved upper bounds on the testing error that depend on the margin
- Let *H a finite space of base classifiers.For delta > 0 and theta > 0* with probability at least 1 –delta over the random choice of the training set Z, any classifier ensemble *D {D1, . . . ,DL} ≤ H combined by the weighted average satisfies*

$$P(\text{error}) \leq P(\text{training margin}) \leq \theta + O\left(\frac{1}{\sqrt{N}}\left(\frac{\log N \log |H|}{\theta^2}\right) + \log(1/\delta)^{1/2}\right)$$

P(error) = probability that the ensemble will make an error in labeling x drawn randomly from the distribution of the problem

P(training margin < theta) is the probabilty that the margin for a randomly drawn data point from a randomly drawn training set does not exceed theta

- Thus the main idea for boosting is to approximate the target by approximating the weight of the function.
- These weights can be seen as the min-max strategy of the game.
- Thus we can apply the notion of game theory for ada-boost.
- This idea has been discussed in the paper of freund and schpaire.

Experiment

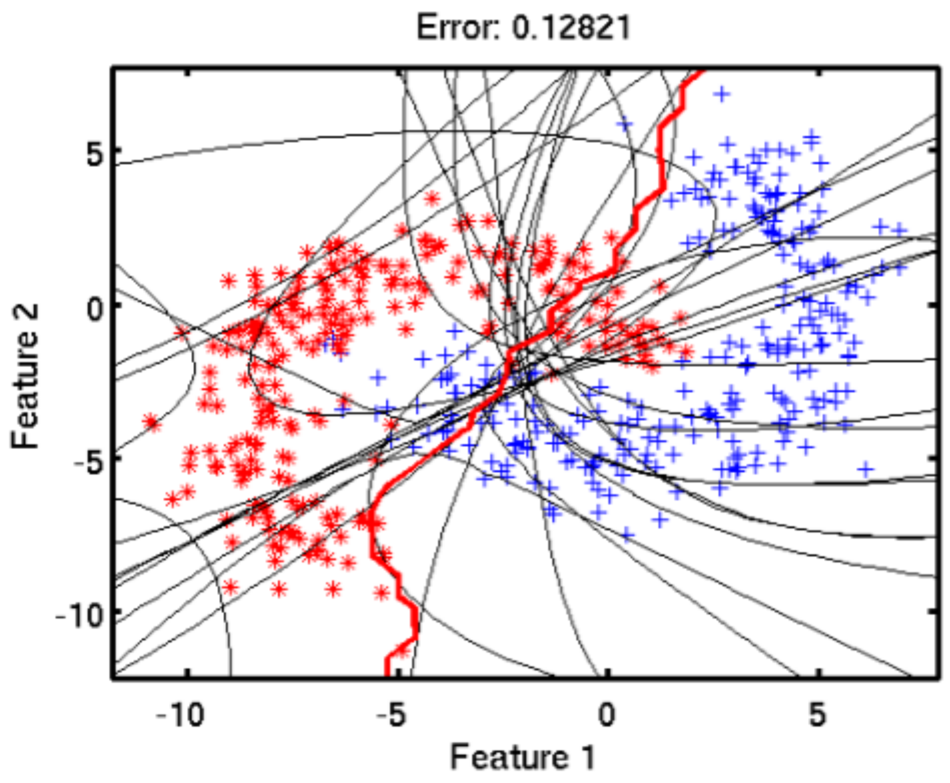
- PR Tools:

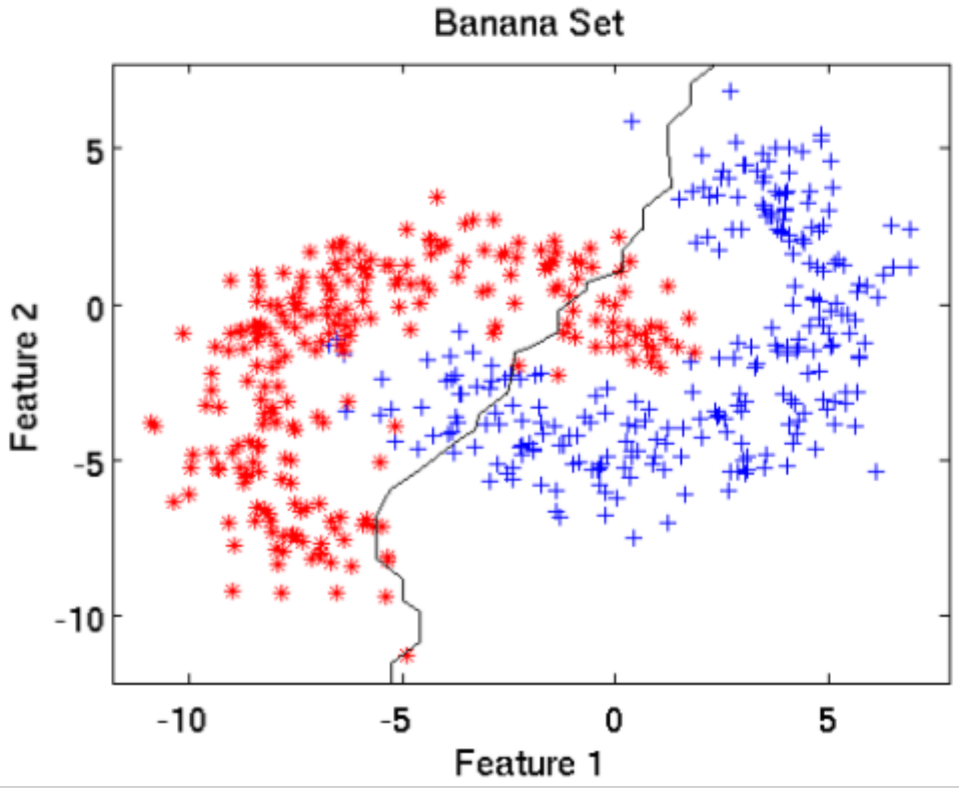
```
>> A = gendatb(500, 1);
>> [W,V,ALF] = adaboostc(A,qdc,20,[],1);
>> scatterd(A)
>> plotc(W)
```

□

- Uses Quadratic Bayes Normal Classifier with default settings, 20 iterations.

Example



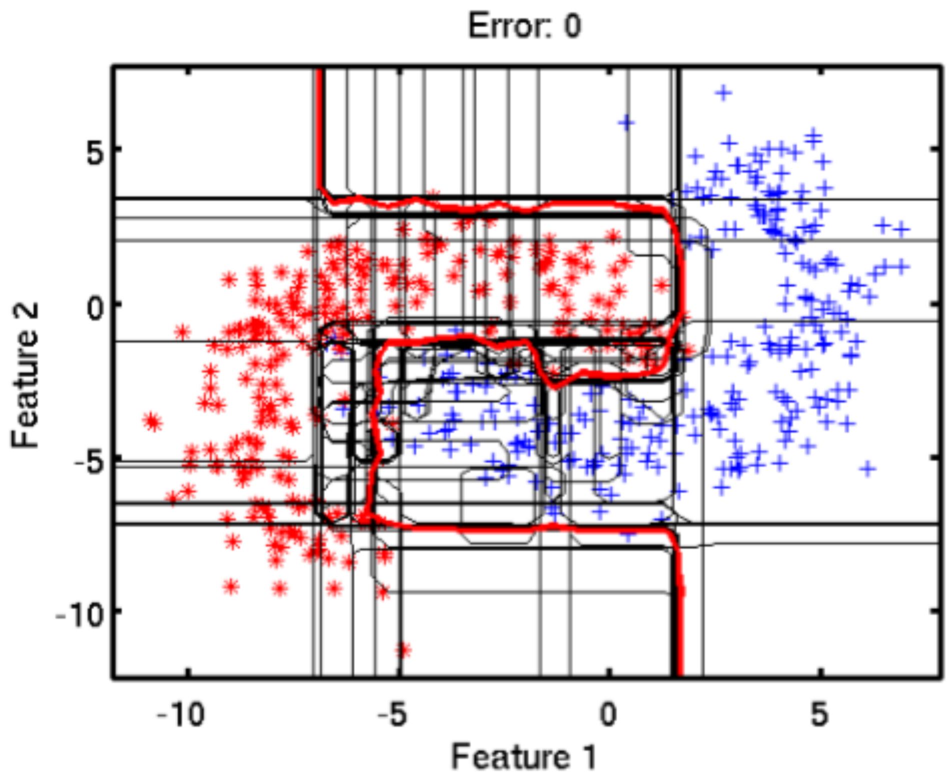


Each QDC classification boundary (black), Final output (red)

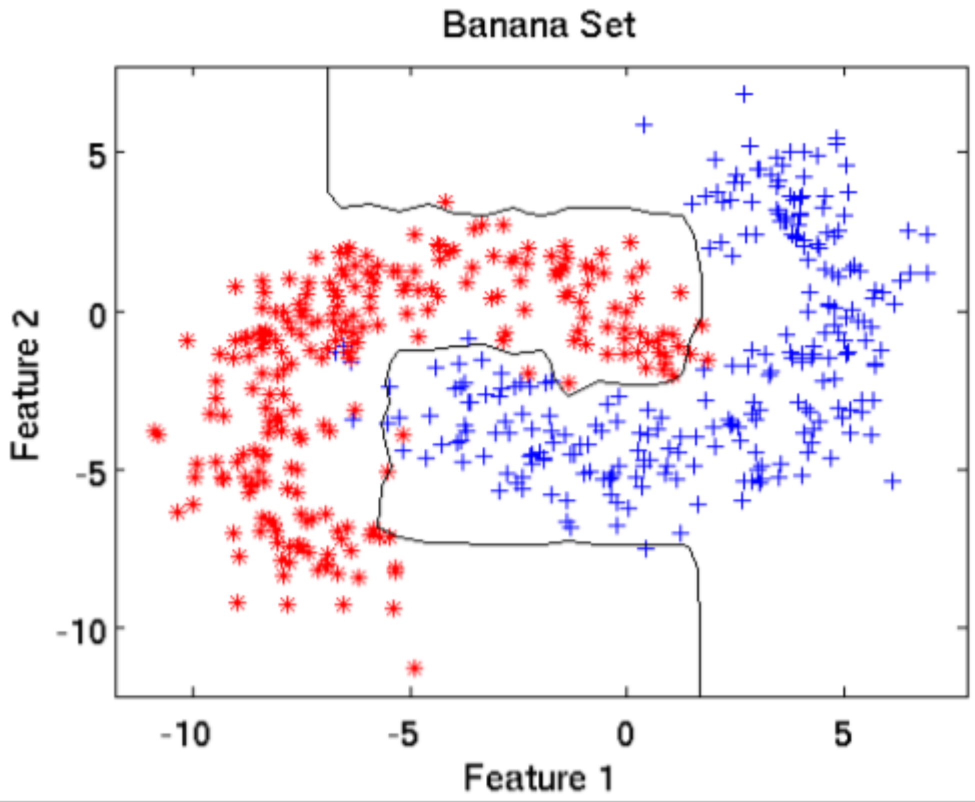
Final output of AdaBoost with 20 QDC classifiers

AdaBoost: QDC

Experiments



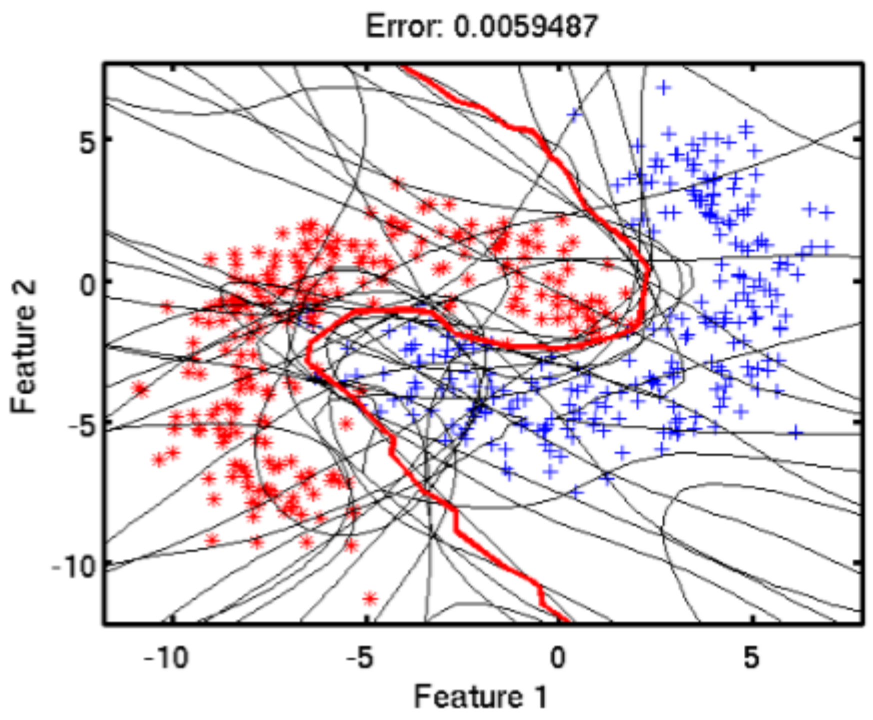
AdaBoost using 20 decision trees with default settings



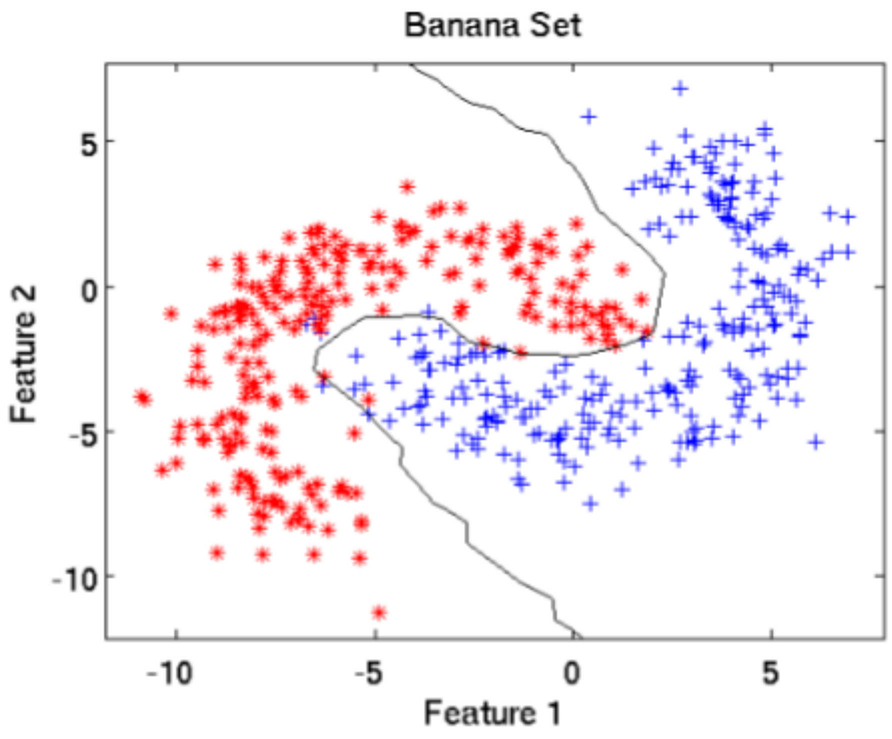
Final output of AdaBoost with 20 decision trees

AdaBoost: Decision Tree

Experiments



AdaBoost using 20 neural nets
[bpxnc] default settings

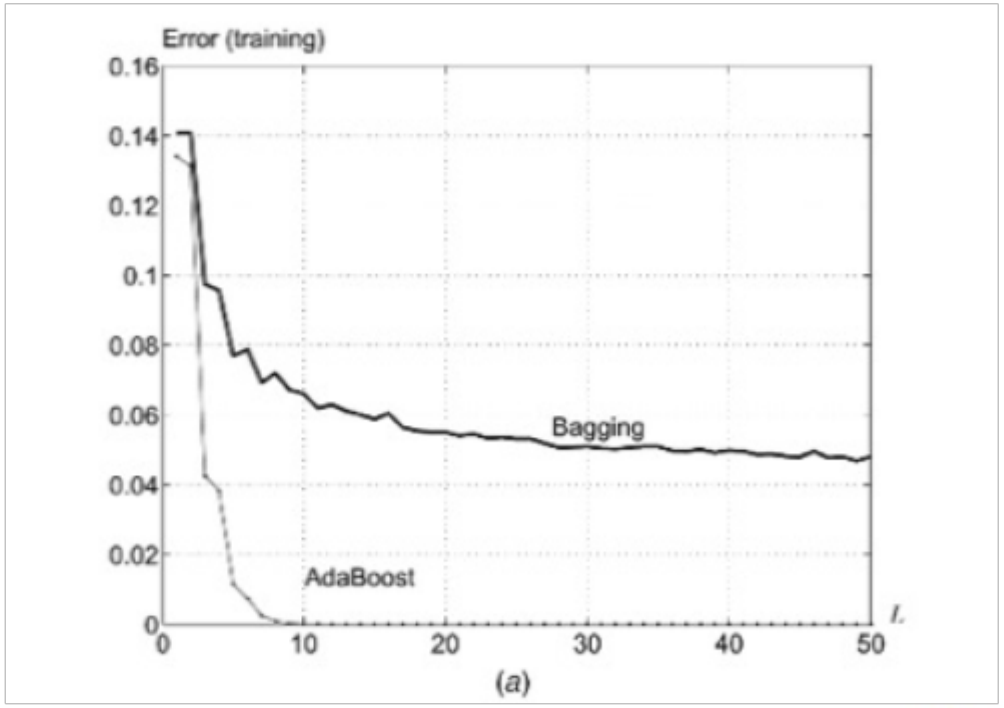


Final output of AdaBoost with 20
neural nets

AdaBoost: Neural Net

Bagging & Boosting

- Comparing bagging and boosting:



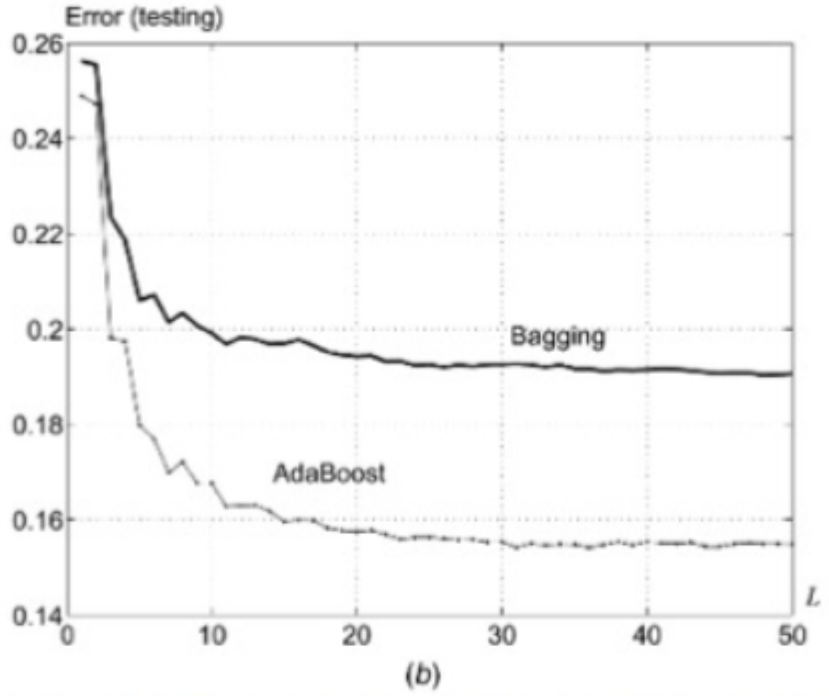


Fig. 7.11 *Training and testing error of bagging and AdaBoost for the rotated check-board example.*

Kuncheva

References

- 1 - A. Krogh and J. Vedelsby (1995).Neural network ensembles, cross validation and activelearning. In D. S. Touretzky G. Tesauro and T. K. Leen, eds., Advances in Neural Information Processing Systems, pp. 231-238, MIT Press.