Ensembles of Classifiers and Clusterings

Reasons for using Ensembles

- **Statistical reasons**:
  - Combining the output of several classifiers may reduce the risk of an unfortunate selection of a poorly performing classifier

Reasons for using Ensembles

- **Large Volumes of Data**:
  - Sometimes, the amount of data to be analyzed can be too large to be handled by a single classifier. Thus, we can:
    - Partition the data into smaller subsets;
    - Train different classifiers;
    - Combine their outputs using a combination rule

Reasons for using Ensembles

- **Too Little Data**:
  - A reasonable sized set of training data is crucial to learn the underlying data distribution. When available data is scarce, we can:
    - Draw overlapping random subsets of the available data using resampling techniques
    - Train different classifiers, creating the ensemble

Reasons for using Ensembles

- **Divide and Conquer:**
  - The given task may be too complex, or lie outside the space of functions that can be implemented by the chosen classifier method (e.g.: non-linear problem, and linear classifiers)
  - Appropriate combinations of simple (e.g., linear) classifiers can learn complex (e.g., non-linear) boundaries

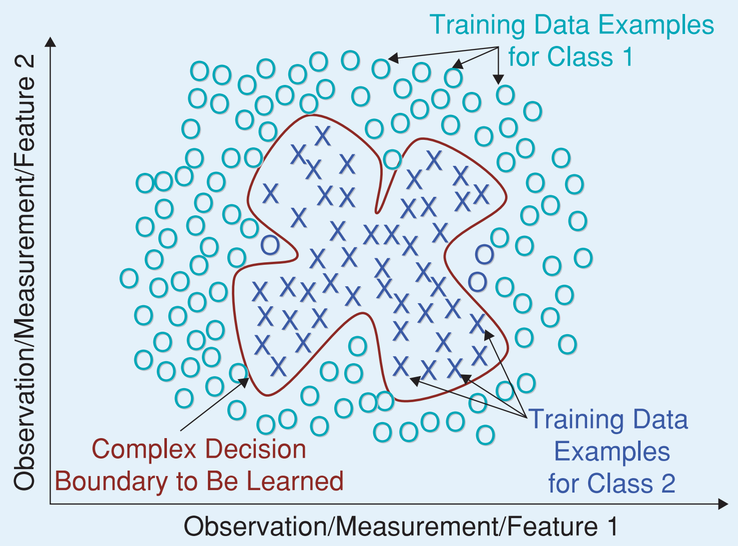


Figure 1. Complex decision boundary that cannot be learned by linear or circular classifiers.

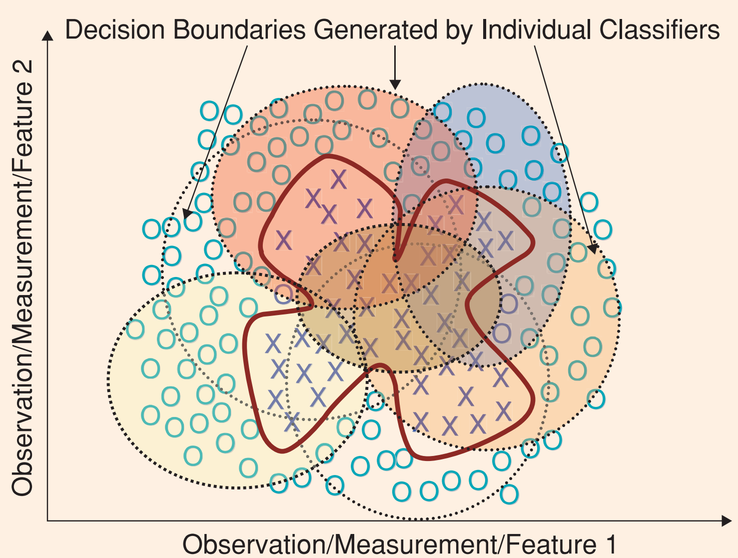


Figure 2. Ensemble of classifiers spanning the decision space.

Reasons for using Ensembles

- **Data Fusion**:
  - Several sets of data obtained from different sources, where the nature of features is different (e.g.: categorical and numerical features)
  - Data from each source can be used to train a different classifier, thus creating an ensemble

Components of an Ensemble

- Two key components:
  - A method to generate the individual classifiers of the ensemble
  - A method for combining the outputs of these classifiers

Diversity: The Key Feature

- The individual classifiers must be diverse, i.e., they make errors on different data
- Intuition: if they make the same errors, such mistakes will be carried into the final prediction
- Thus: the errors the classifiers make should be uncorrelated

Accuracy

- The component classifiers need to be “reasonably accurate” to avoid poor classifiers to obtain the majority of votes.

- <u>Intuition</u>: If the components of the ensemble are poor classifiers, they make a lot of errors, and those errors are carried out to the final prediction.

Accuracy and Diversity

- Requirements for accuracy and diversity have been quantified:
  - Under simple majority voting and *independent error conditions*, if all classifiers have the same probability of error of *less than 50%*, then the error of the ensemble decreases monotonically with an increasing number of classifiers.

How to achieve diversity

- Use different training data sets to train individual classifiers
- Such data sets are often obtained through resampling techniques (***bootstrapping*** or ***bagging***): training data subsets are drawn randomly, usually with replacement, from the entire training data

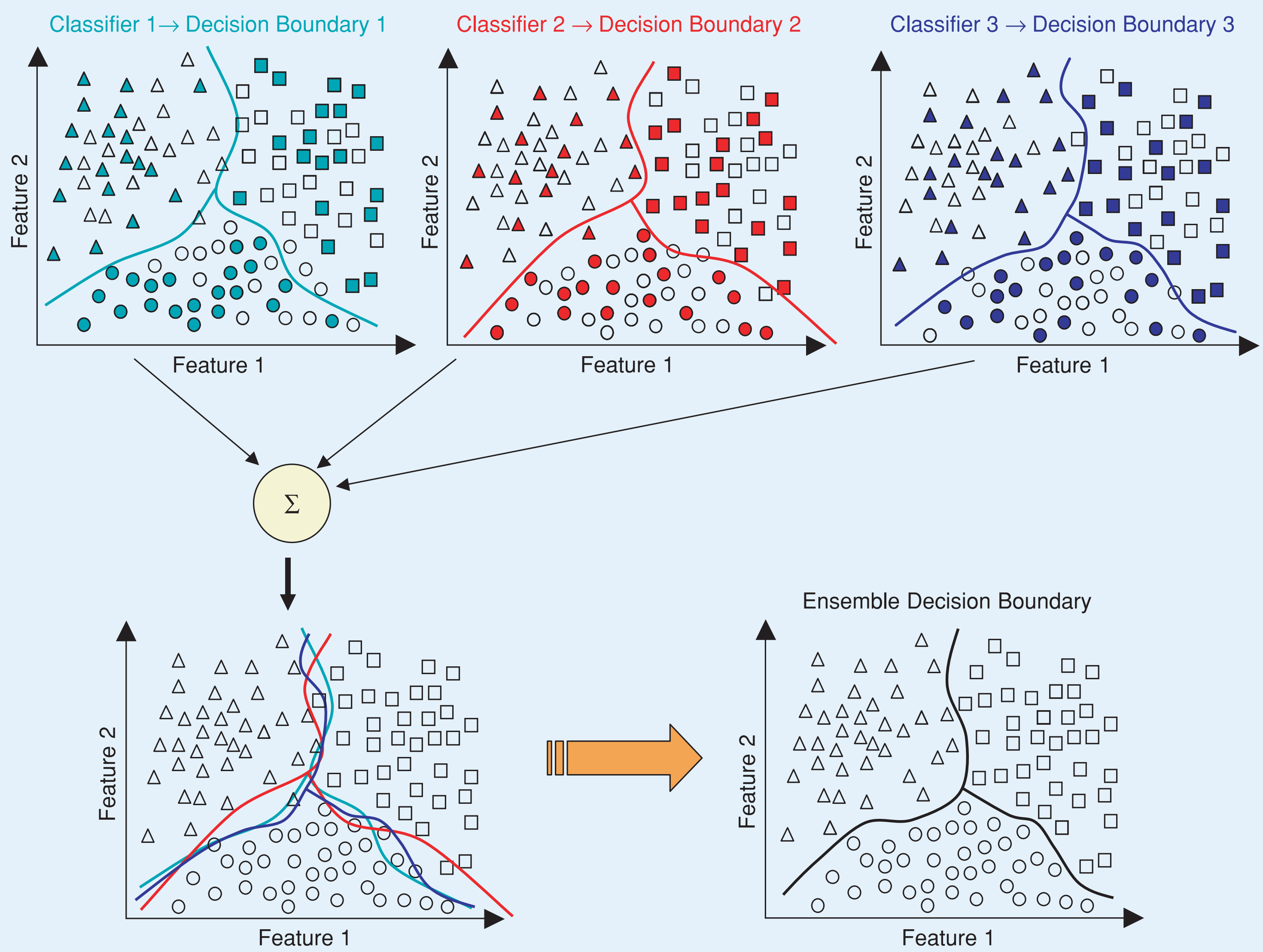


Figure 3. Combining classifiers that are trained on different subsets of the training data.

How to achieve diversity

- Use different training data sets to train individual classifiers
- If the training data subsets are drawn without replacement, the procedure is also called ***jackknife*** or ***k-fold*** data split: the entire data set is split into k blocks, and each classifier is trained only on k-1 of them. A different subset of k blocks is selected for each classifier

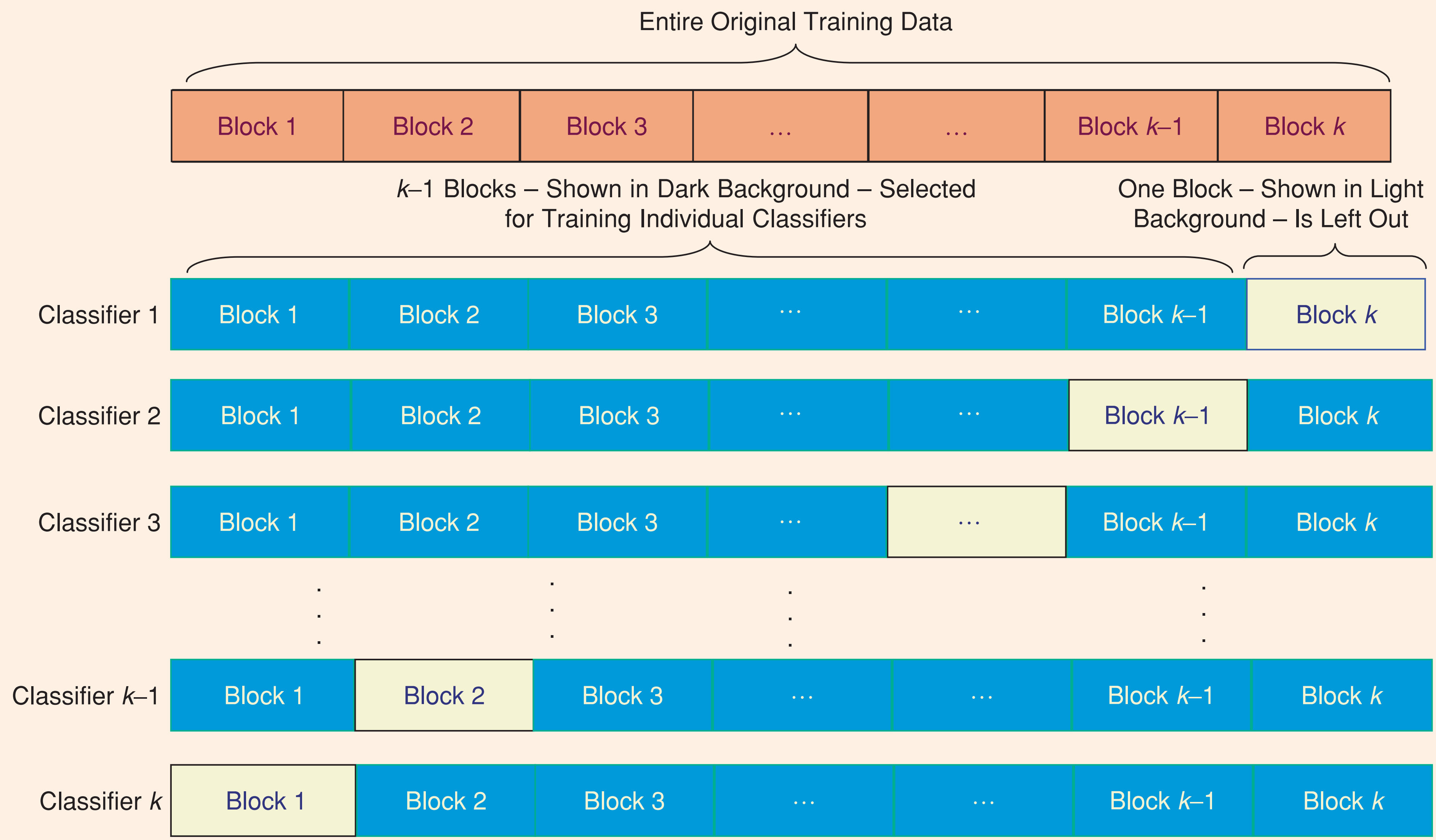


Figure 4. *k*-fold data splitting for generating different, but overlapping, training datasets.

How to achieve diversity

- When is bagging (bootstrapping) effective?
- To ensure diverse classifiers, the base classifier should be ***unstable***, that is, *small changes* in the training set should lead to *large changes* in the classifier output.

How to achieve diversity

- When is bagging (bootstrapping) effective?
- Large error reductions have been observed with *decision trees* and bagging. This is because decision trees are highly sensitive to small perturbations of the training data.

How to achieve diversity

- When is bagging (bootstrapping) effective?
- Bagging is not effective with *nearest neighbor classifiers*. Why? NN classifiers are highly stable with respect to variations of the training data
- It has been shown that the probability that any given training point is included in a data set bootstrapped by bagging is approximately 63.2%. It follows that the nearest neighbor will be the same in 63.2% of the classifiers
- Thus, the errors are highly correlated, and bagging becomes ineffective

How to achieve diversity

- Use different training parameters for different classifiers
- E.g., ensemble of neural networks trained with different weight initialization, or different number of layers/nodes
- If the base classifier is unstable with respect to the tuning parameters, diverse classifiers can be generated

How to achieve diversity

- Use different type of classifiers
- E.g., an ensemble of neural networks, decision trees, nearest neighbor classifiers, and support vector machines

How to achieve diversity

- Use different subsets of features to train the individual classifiers
- E.g., random feature subsets (random subspace method)
- This approach is effective with nearest neighbor (NN) methods, because NN techniques are highly sensitive to the chosen features

Boosting

Boosting

- Similar to bagging, boosting also creates an ensemble of classifiers by resampling the data, which are then combined by majority voting
- In boosting, though, the resampling strategy is geared to provide the **most informative** training data for each consecutive classifier

Boosting (Adaboost.M1)

Freund and Schapire, 1996

- Generates a set of classifiers, and combines them through weighted majority voting of the classes predicted by the individual classifiers
- Classifiers are trained using instances drawn from an iteratively updated distribution of the training data
- The distribution ensures that instances misclassified by the previous classifier are more likely to be included in the training data of the next classifier
- Thus, consecutive classifiers' training data are more geared towards increasingly hard-to-classify instances

Algorithm AdaBoost.M1

Input:

- Sequence of N examples $S = [(\mathbf{x}_i, y_i)]$, $i = 1, \cdots, N$ with labels $y_i \in \Omega$, $\Omega = \{\omega_1, \ldots, \omega_C\}$;
- Weak learning algorithm **WeakLearn**;
- Integer T specifying number of iterations.

Initialize $D_1(i) = \frac{1}{N}.$, $i = 1, \cdots, N$ (11)

Do for $t = 1, 2, \ldots, T$:

1. Select a training data subset S_t, drawn from the distribution D_t.
2. Train **WeakLearn** with S_t, receive hypothesis h_t.
3. Calculate the error of h_t: $\varepsilon_t = \sum_{i:h_t(\mathbf{x}_i) \neq y_i} D_t(i)$. (12)

   If $\varepsilon_t > 1/2$, **abort**.
4. Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$. (13)
5. Update distribution

$$D_t : D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(\mathbf{x}_i) = y_i \\ 1, & \text{otherwise} \end{cases} \quad (14)$$

   where $Z_t = \sum_i D_t(i)$ is a normalization constant chosen so that D_{t+1} becomes a proper distribution function.

Test – Weighted Majority Voting: Given an unlabeled instance $\mathbf{x}$,

1. Obtain total vote received by each class

$$V_j = \sum_{t:h_t(\mathbf{x})=\omega_j} \log \frac{1}{\beta_t}, \; j = 1, \ldots, C. \quad (15)$$

2. Choose the class that receives the highest total vote as the final classification.

Figure 8. The AdaBoost.M1 algorithm.

Boosting (property)

- Freund and Schapire proved that, provided that is always $\epsilon_t < 0.5$, the error rate of boosting on a given training data set, under the original uniform distribution, approaches zero exponentially fast as T increases.

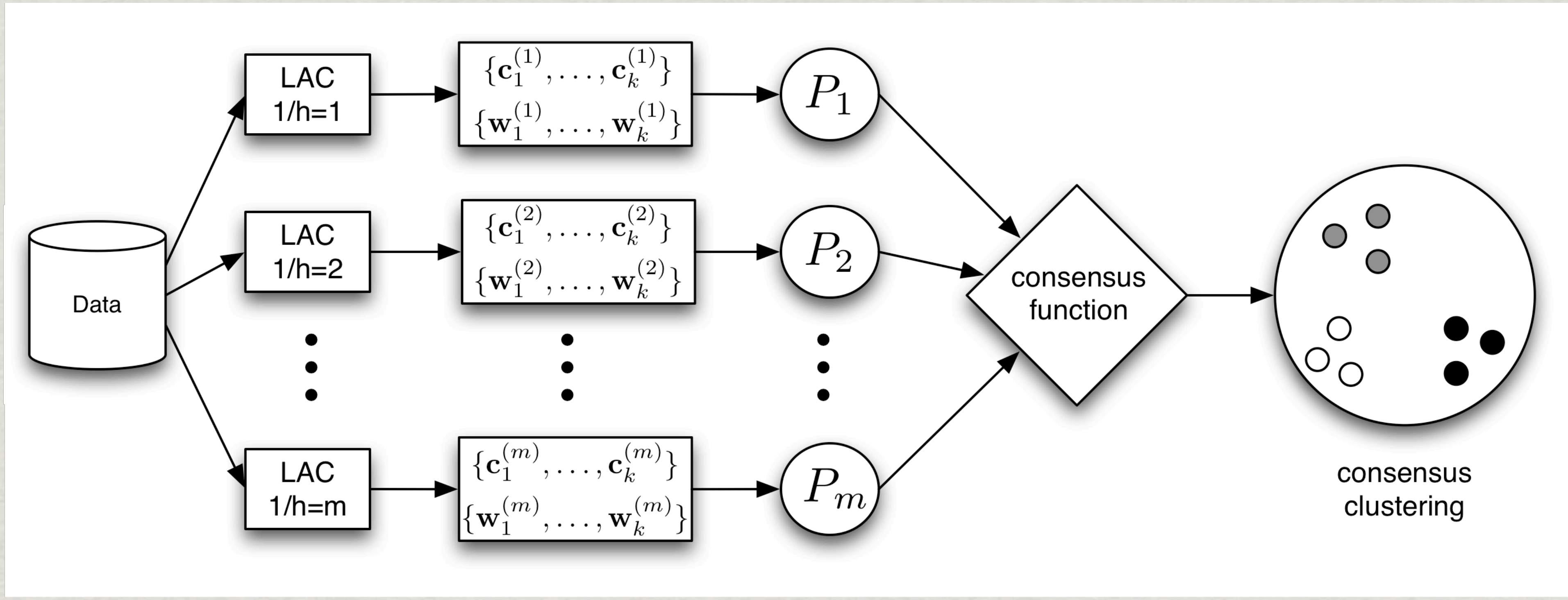
Boosting (property)

- Thus, a succession of weak classifiers can be boosted to a strong classifier that is at least as accurate as, and usually more accurate than, the best weak classifier on the training data.
- Of course, this gives no guarantee on the generalization performance on unseen instances.

Clustering Ensembles

- Clustering ensembles leverage the diversity of the input clusterings to generate a consensus clustering that is superior to the component ones;
- Clustering ensembles offer a solution to challenges inherent to clustering arising from its ill-posed nature;
- The major challenge is to find a consensus clustering that achieves an improved clustering of the data

The Clustering Ensemble process



- Goal: Aggregate a collection of base clusterings to produce a partition of the data that is more accurate that the component ones

Clustering Ensembles

- A clustering ensemble technique is characterized by two components:
  - The mechanism to generate diverse clusterings
  - The consensus function to combine the input clusterings into a final clustering

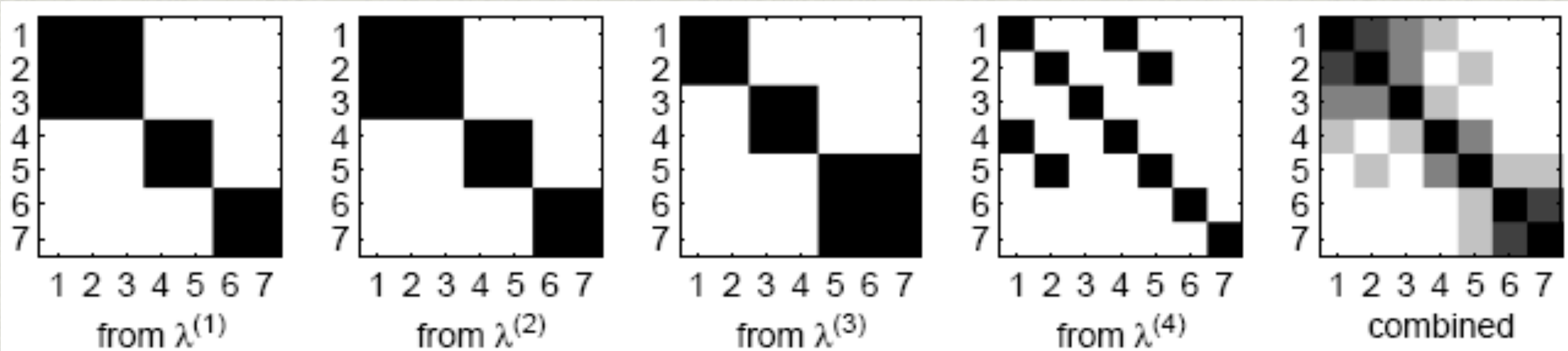
Clustering Ensembles

- Diverse component clusterings can be generated by:
  - Varying the number and/or location of initial centroids
  - Using different clustering algorithms
  - Sub-sampling features or data

Clustering Ensembles

- A popular methodology to build a consensus function is to use the co-association matrix:
  - Two points have similarity 1 if they belong to the same cluster; similarity 0 otherwise
  - This defines a binary similarity matrix for each clustering
  - Lets consider an example...

Clustering Ensembles



- Overall similarity matrix S: entry-wise average of the m individual matrices (m=4 above)

- An element of S represents the fraction of clusterings in which two data are in the same cluster

- S is used to re-cluster the data using a similarity-based clustering algorithm, e.g., hierarchical clustering

Clustering Ensembles

- A different popular mechanism for constructing a consensus maps the problem onto a graph-based partitioning setting:
    - From S, a similarity graph is induced: vertices correspond to data, and edge weights represent the similarity between the corresponding two vertices
    - A k-way partitioning of the vertices that minimizes the edge weight-cut is computed
    - The result gives the consensus clustering.