Data Mining

CS57300
Purdue University

Bruno Ribeiro

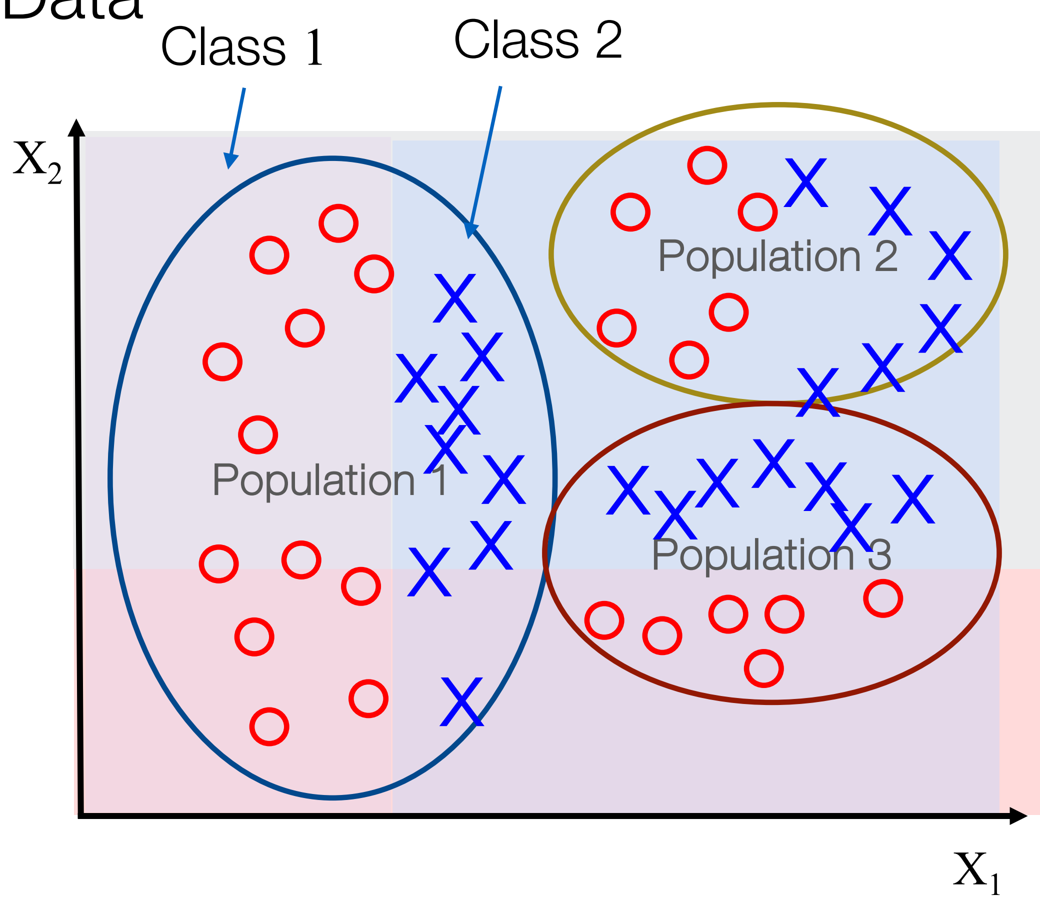
February 15th, 2018

Today’s Goal

- Ensemble Methods
  - Supervised Methods
    - Meta-learners
  - Unsupervised Methods

Understanding Ensembles

The Data



Three simple classification problems if we can break them down

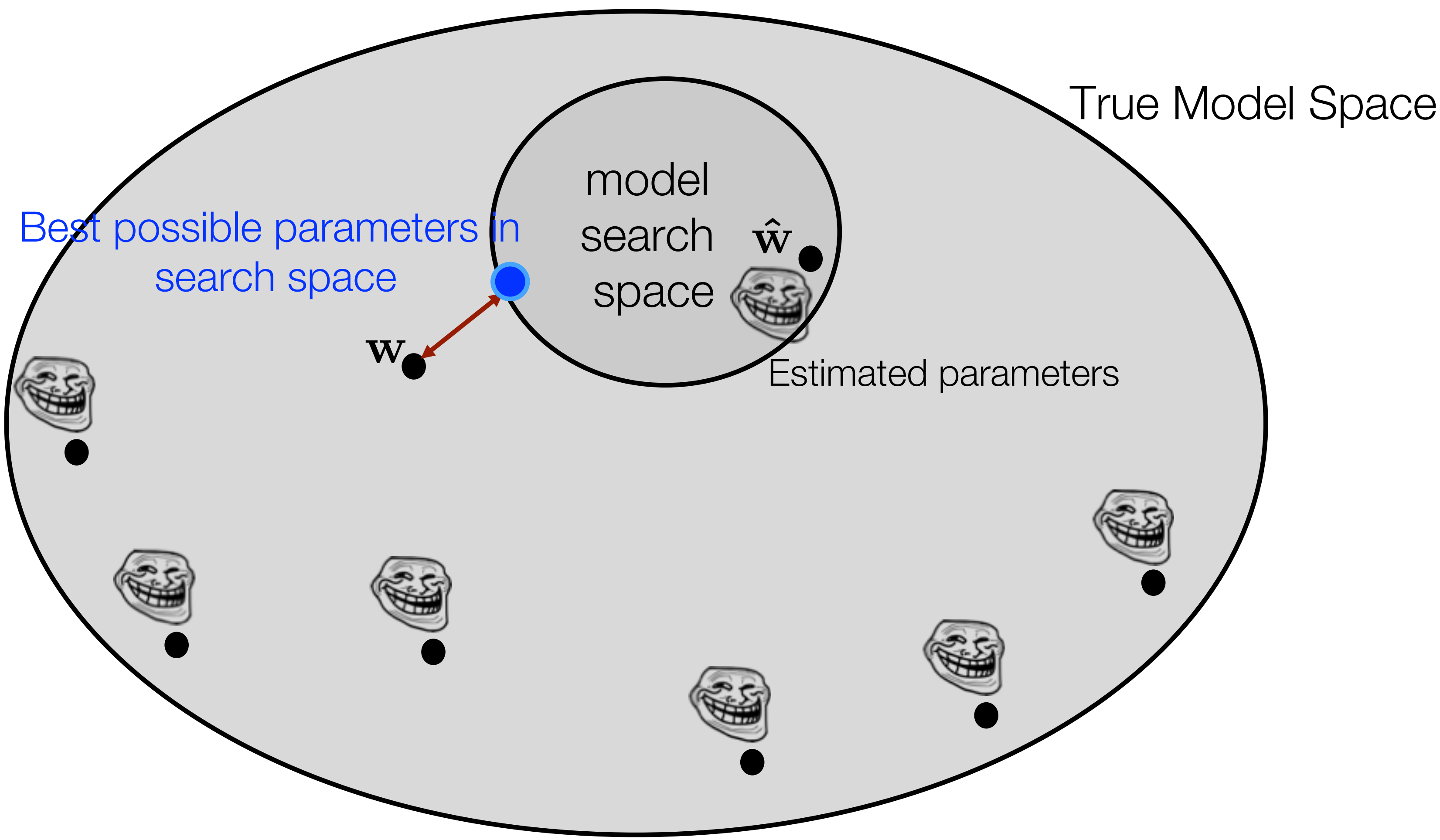
If no classes, clustering problem is also simple if clustering algorithm knows shape of cluster

Why just not choose the **Best** classifier?

- Choosing the ”best” classifier is a tricky business
  - *Why select just one of multiple similarly good hypotheses*
- Ensembles are a way to include multiple hypotheses in the decision
  - Most methods can be thought as approximations to a Bayesian approach that considers a continuous pool of models

- Can be justified by Bias-Variance reduction:
  - *Variance*: error from sensitivity to small fluctuations in the training set
  - *Bias*: erroneous assumptions in the model

Model Search Space [Lecture 7]



- Bias related to arrow  = parameter values look better than **w** in the training data
- Variance related to number of "trolls"

Bias-Variance Reduction

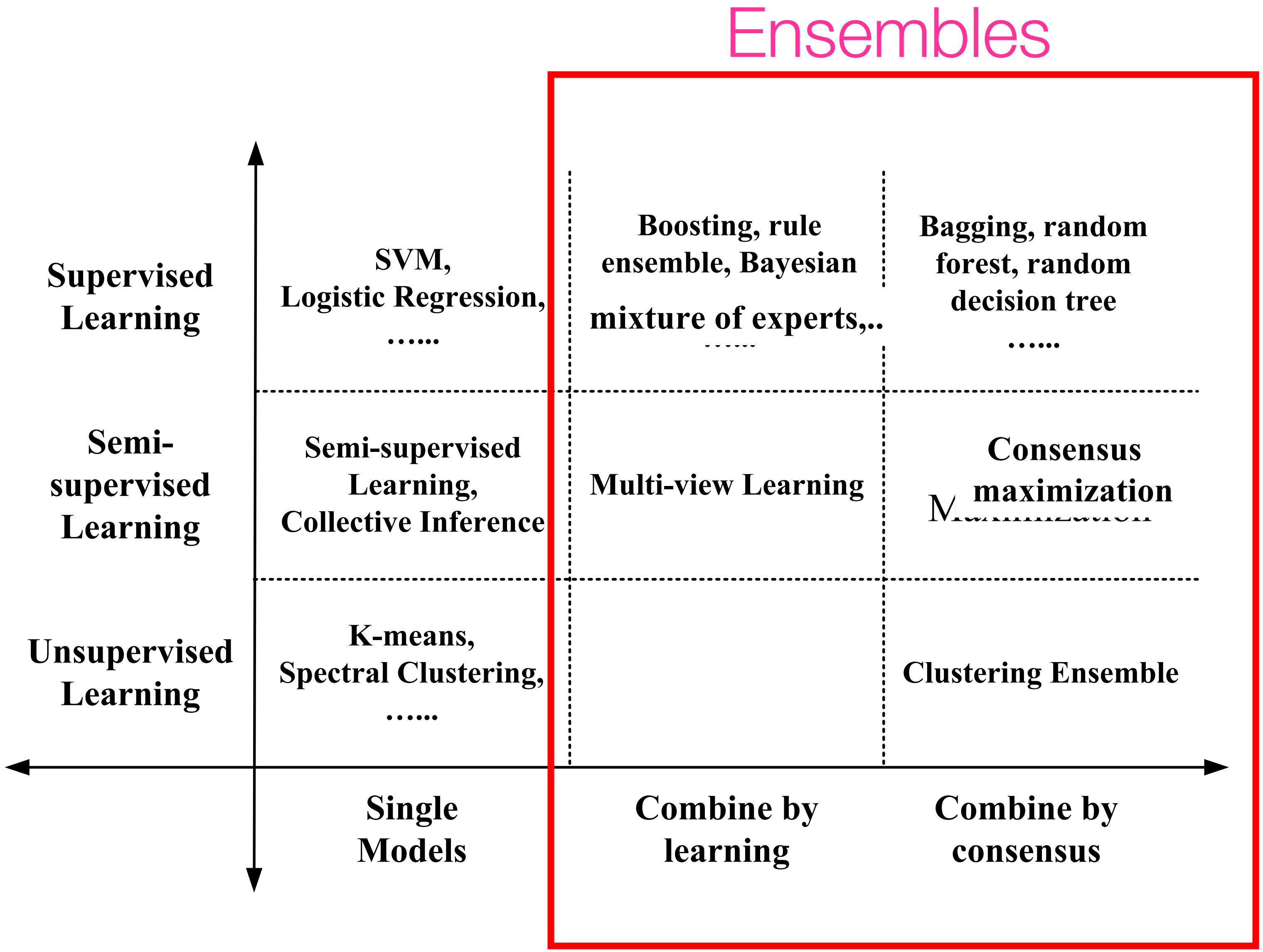
- *Variance reduction:* if the training sets are completely independent, it will always help to average an ensemble to reduce variance without affecting bias (e.g., bagging). Reduces sensitivity to individual data points

- *Bias reduction:* for simple models, average of models has much greater model complexity than single model (e.g., hyperplane classifiers, Gaussian densities).
    - Averaging models can reduce bias substantially by increasing model complexity, and control variance by fitting one component at a time (e.g., boosting)

Advantages of Ensembles

- No need to select best model
- Put all models to work, make them work together
- Ensembles are generally more accurate than any individual members:
  - As long as constituent members of the ensemble are
    - Accurate (better than guessing)
    - Diverse (different errors on new examples)

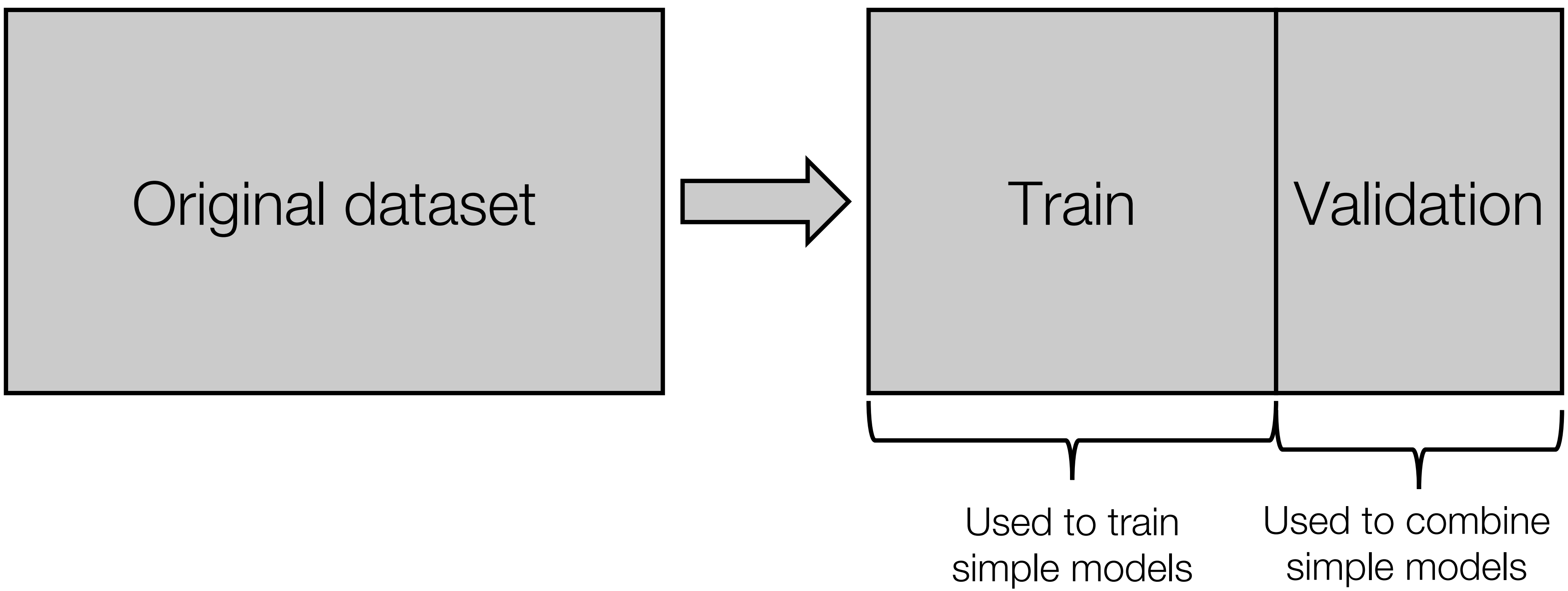
Overview



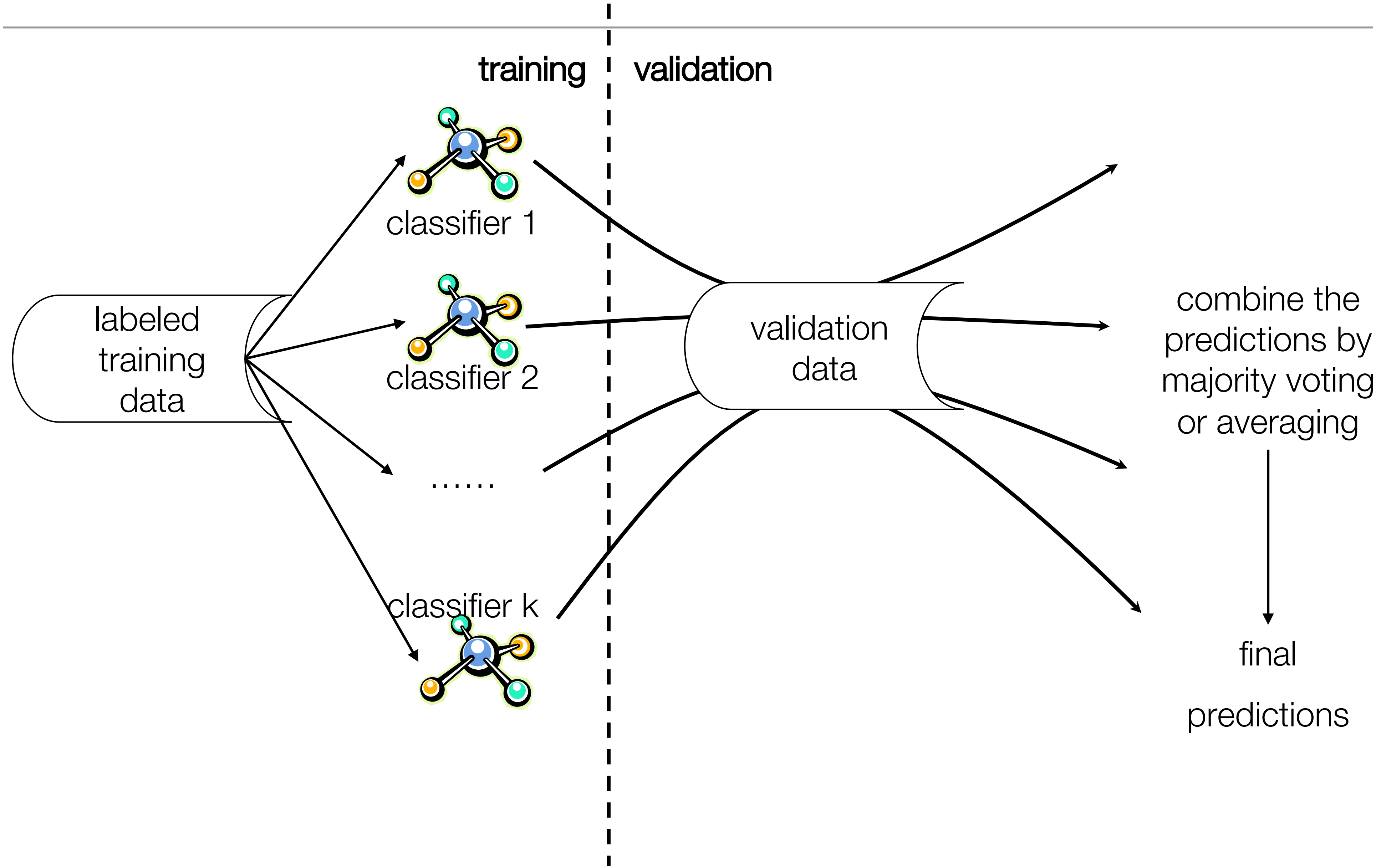
Ack: Gao, Fan, Han 2010

Training and Validation Data

- Randomly divide original data $\{x_1,\ldots,x_N\}$ into training and validation
- We will need this separation to train some ensembles



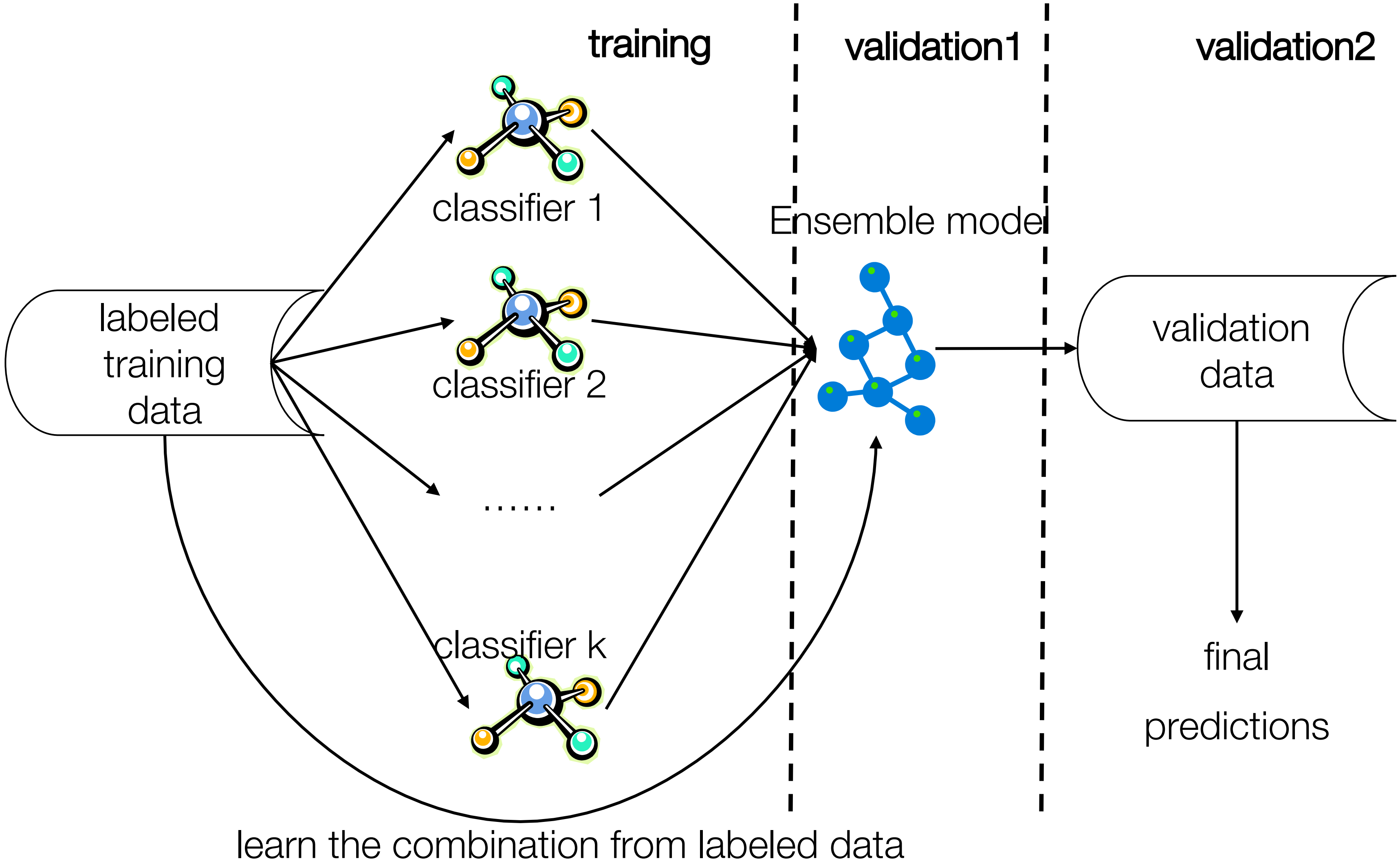
Ensemble of Classifiers—Consensus



Algorithms: bagging, random forest, random decision tree, model averaging of probabilities……

Ack: Gao, Fan, Han 2010

Ensemble of Classifiers—Learn to Combine



Algorithms: boosting, stacked generalization, rule ensemble, Bayesian model averaging, mixture of experts……

Ack: Gao, Fan, Han 2010

Pros and Cons

	Combine by learning	Combine by consensus
Pros	Get useful feedbacks from labeled data Can potentially improve accuracy	Do not need labeled data Can improve the generalization performance
Cons	Need to keep the labeled data to train the ensemble May overfit the labeled data Cannot work when no labels are available	No feedbacks from the labeled data Require the assumption that consensus is better

Ack: Gao, Fan, Han 2010

Ensemble Training Strategies

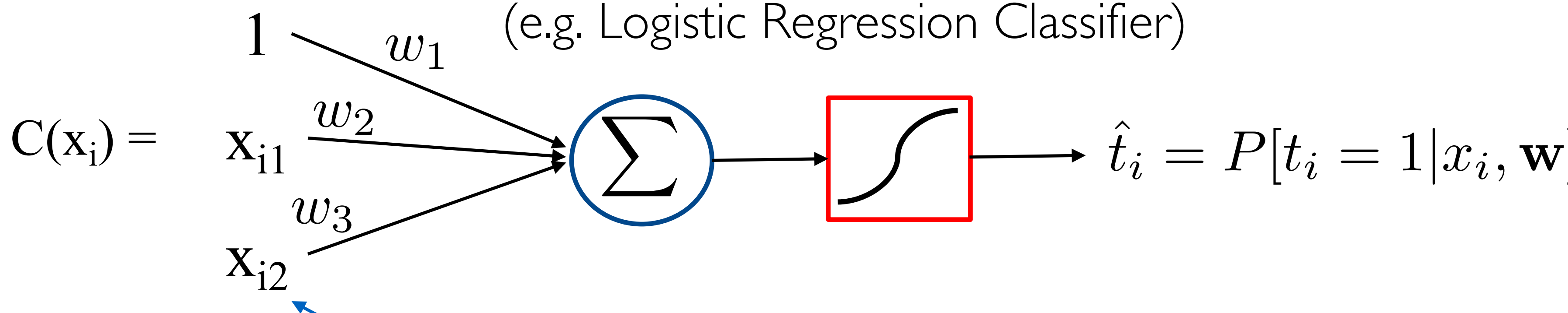
- Methods differ in training strategy and assembling
- Parallel training with different training sets
    - Bagging (bootstrap aggregation) – bootstrap to train separate models, average their predictions
    - Cross-validated committees – disjoint subsets of training sets
- Sequential or Joint training
    - Boosting: Iteratively re-weights training examples so each new classifier focuses on hard examples (supervised only)
    - Mixture of experts: parallel training with objective encouraging division of labor
    - Must be careful w.r.t. model complexity

Learn to Combine: Supervised Ensembles w/ Parallel Training

Learning to Combine with Logistic Regression

Can be any classifier

Supervised Learning Classifier
(e.g. Logistic Regression Classifier)



What if "input" x_{ik} was
the decision $\hat{y}_i^{(k)}$ of another classifier $G_k(x_i)$?

High-level classifier is known as meta learner

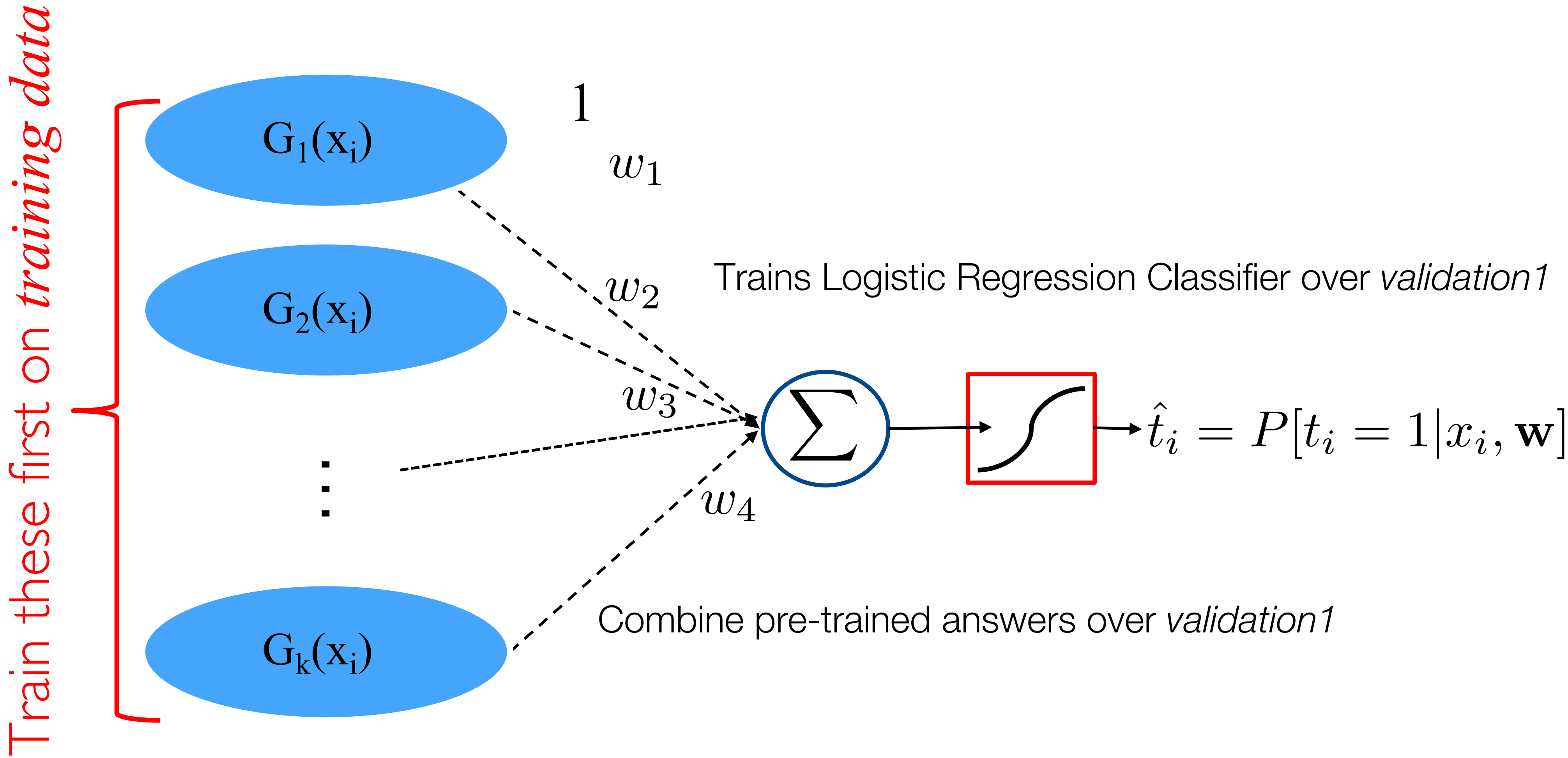
General Algorithm (Learn to Combine)

- Problem
  - Given a data set $D=\{x_1,x_2,\ldots,x_n\}$ and their corresponding labels $T=\{t_1,t_2,\ldots,t_n\}$ from a set of labels L
  - An ensemble approach computes:
    - A set of classifiers $\{f_1,f_2,\ldots,f_K\}$, each of which maps data to a class label: $f_j(x)=l,\ \ l \in L$
      - All trained in parallel
    - A combination of classifiers f^* which minimizes generalization error: $f^*(x)= g(f_1(x), f_2(x),\ldots, f_K(x))$

Ack: Gao, Fan, Han 2010

Example of *Learn to Combine*
Stacked generalization to reduce bias

Stacked Generalization Example



Train	Validation1	Validation2

Stacked generalization (stacking) [Wolpert, 1992]

- *Stacked generalization* combines multiple models through a meta learner
- Simplest approach
- Stacking usually combines models of different types.
- The algorithm:
  1. Split the original training set into *training, validation1, validation2*
  2. Train k=1,…,K models (*base learners*) on *training data*
  3. Apply the models over ∀x∈*validation1*: for classifier k, predict label $f_k(x)$
  4. Train a classifier on *validation1* using the predictions from (3) as input and the correct labels as output
  5. *Evaluation ensemble with validation2*

Example of *Combine*
Bagging to reduce variance

works for small sample size

Bagging

- *Bootstrapping*: Repeatedly draw *n* samples from training dataset *D*
  - Sample with replacement
  - Contains around 63.2% original records in each sample
- *Bagging: bootstrap aggregation* (Breiman 1994)
- Algorithm:
  1. Generate *K* bootstrap samples from your original training set
  2. Train on bootstrap sample $k=1,\ldots,K$ to get model f_k
  3. Average the model predictions

$$f^{\star}(x_i) = \frac{1}{K}\sum_{k=1}^{K} f_k(x_i)$$

  - For regression: average predictions
  - For classification: average class probabilities (or take the majority vote if only hard outputs available)

- The more bootstraps the better, so use as many as you have time for

Bagging, Error Reduction

- Error Reduction
  - Under mean squared error, bagging reduces variance and leaves bias unchanged
  - Consider idealized bagging estimator:

$$f^\star(x_i) = E\left[f_k(x_i)\right],$$

where the expectation is over all possible classifiers f_k

  - Bagging usually decreases MSE
  - Bagging is **not** effective with large training datasets
    - Why?

Bagging aims to reduce variance, but large dataset

A specific type of Bagging: Random Forest

Random Forests (1)

- Algorithm
  - Choose *T*: the number of trees to grow
  - Choose *m<M* (M is the number of total features) —number of features used to calculate the best split at each node (typically 20%)
  - For each tree
    - Choose a training set by choosing *n* times (*n* is the number of training examples) with replacement from the training set
    - For each node, randomly choose *m* features and calculate the best split
    - Fully grown and not pruned
  - Use majority voting among all T trees to classify a new example

*[Breiman01]

Ack: Gao, Fan, Han 2010

Random Forests (2)

- Discussions
  - Bagging+random features
  - Improves accuracy
    - Incorporates more diversity and reduce variance
  - Improved efficiency
    - Searching among subsets of features is much faster than searching among the complete set
    - Still quite slow but easily parallelizable

Ack: Gao, Fan, Han 2010

Random Decision Tree (1)

- Single-model learning algorithms
  - Fix structure of the model, minimize some form of errors, or maximize data likelihood (eg., Logistic regression, Naive Bayes, etc.)
  - Use some “free-form” functions to match the data given some “preference criteria” such as information gain, gini index and MDL. (eg., Decision Tree, Rule-based Classifiers, etc.)
- Such methods will make mistakes if
  - Data is insufficient
  - Structure of the model or the preference criteria is inappropriate for the problem
- Decision Trees
  - Make no assumption about the true model, neither parametric form nor free form
  - Do not prefer one base model over the other, just average them

Ack: Gao, Fan, Han 2010

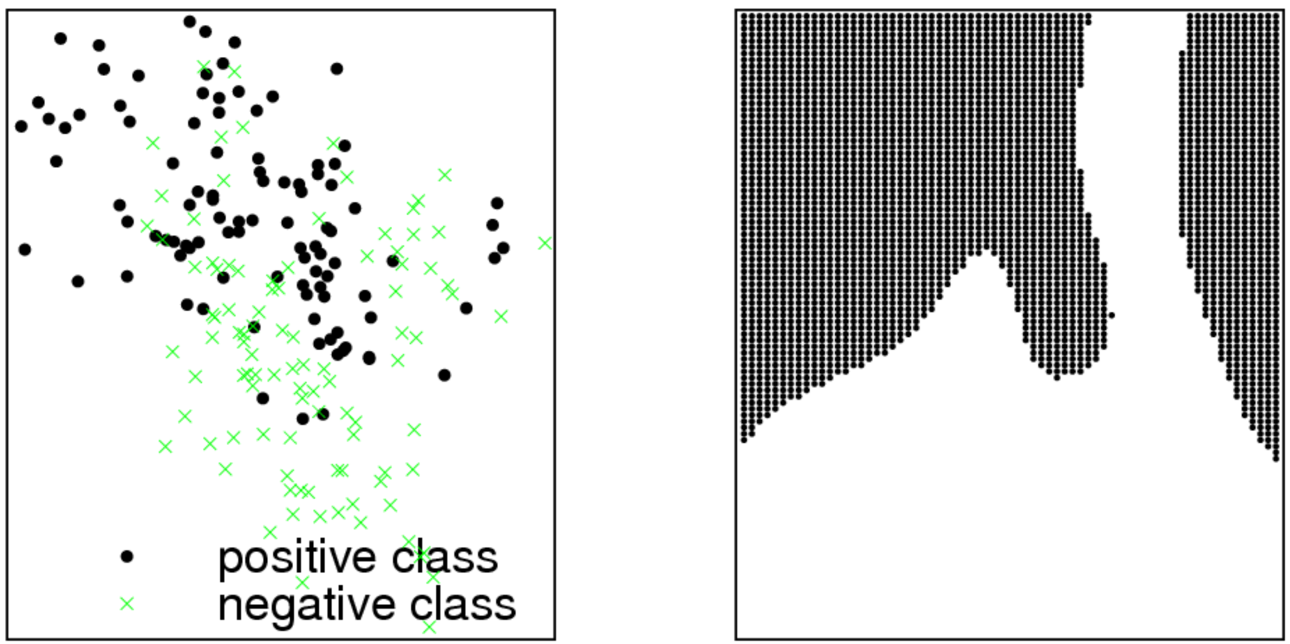
Random Decision Tree (2)

- Algorithm
  - At each node, an un-used feature is chosen at random
    - A discrete feature is unused if it has never been chosen previously on a given decision path starting from the root to the current node.
    - A continuous feature can be chosen multiple times on the same decision path, but each time a different threshold value must be chosen
  - We stop when one of the following happens:
    - A node becomes too small (<= 3 examples).
    - Or the total height of the tree exceeds some limits, such as the total number of features.
  - Prediction
    - Simple averaging over multiple trees

Ack: Gao, Fan, Han 2010

Optimal Decision Boundary

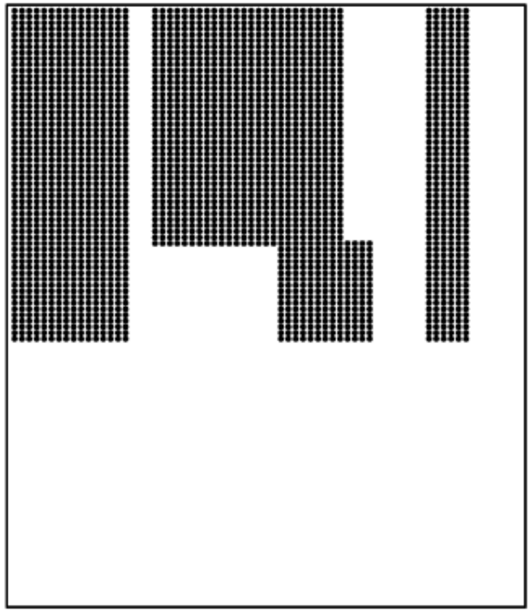
Figure 3.5: Gaussian mixture training samples and optimal boundary.



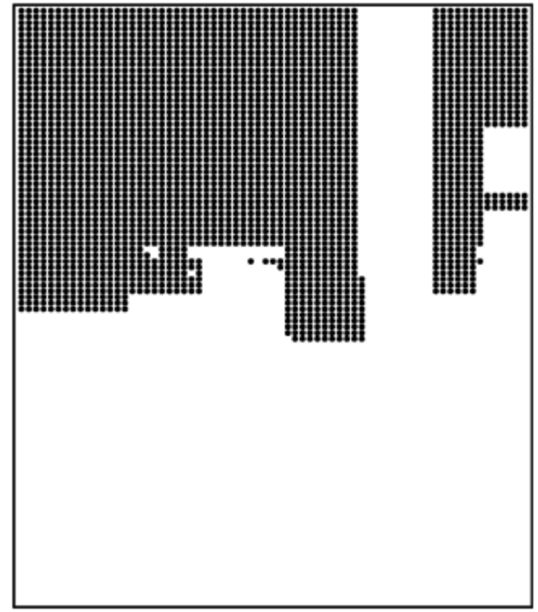
training samples

optimal boundary

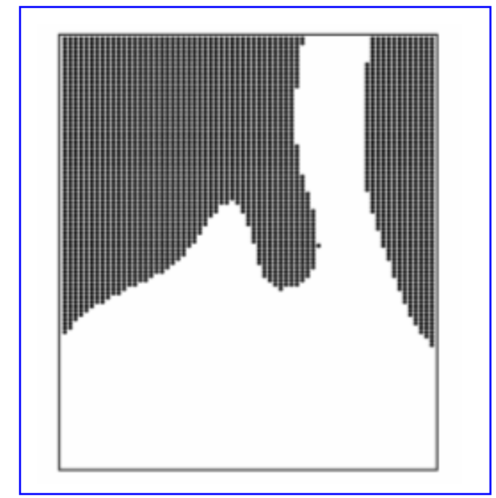
Ack: Tony Liu

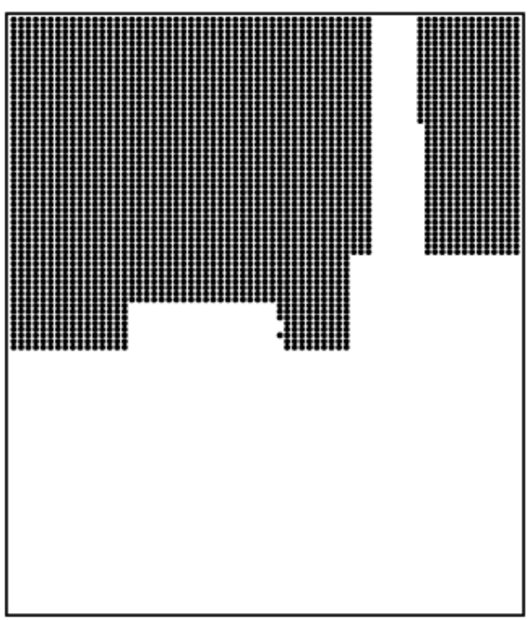


(a) unpruned C4.5

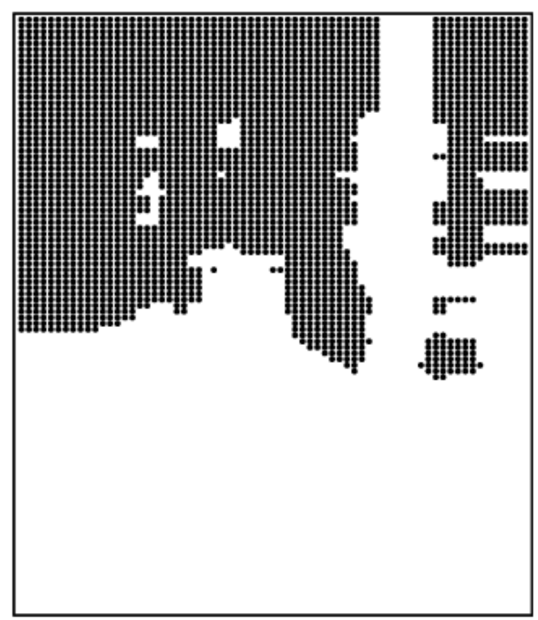


(b) Bagging





(c) Random Forests



(d) Complete-random tree ensemble

RDT looks like the optimal boundary

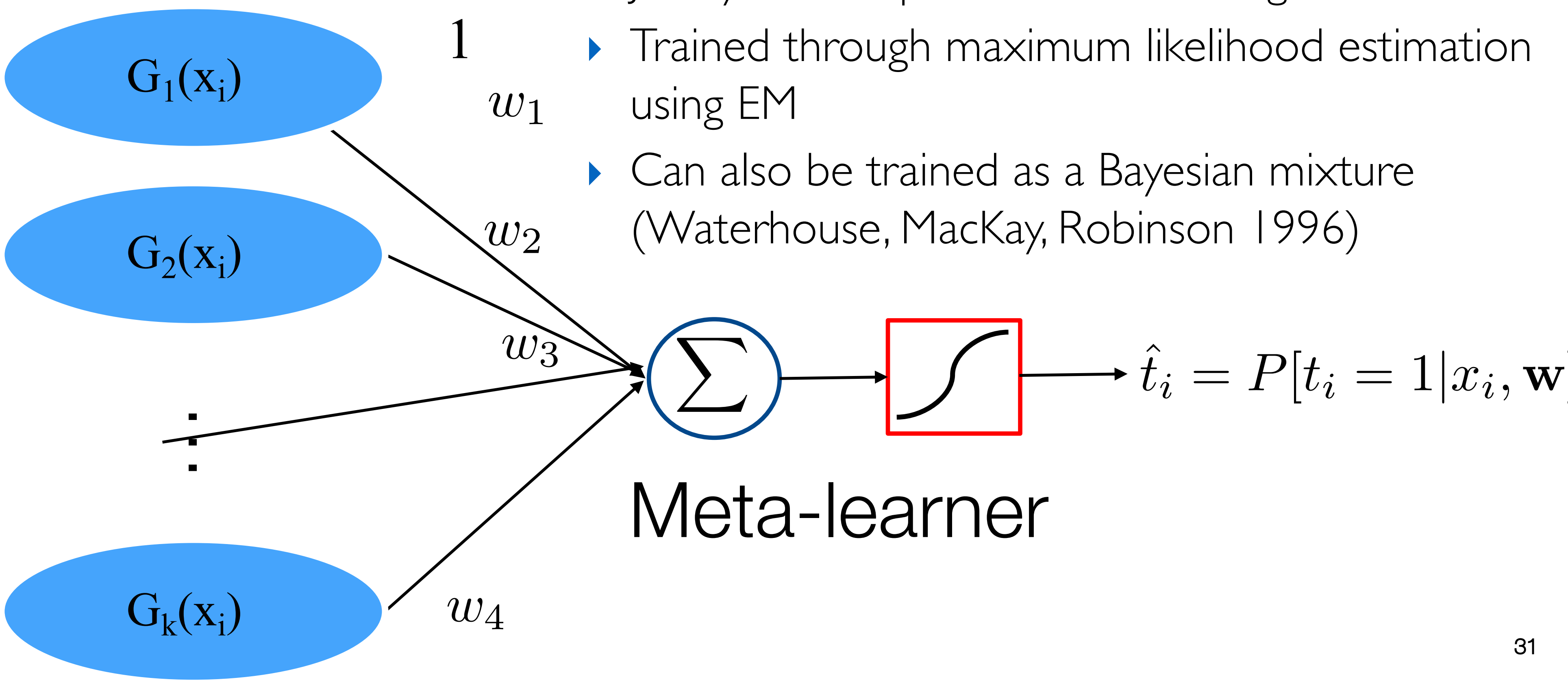
Ack: Tony Liu

Supervised Ensembles
- Sequential or Joint Learning Methods

Mixture of Experts: Basic Idea

- Hierarchical mixture of experts (HME) architecture (Jordan & Jacobs 1994)
- Similar to Stacked Generalization but trains base learners and meta learner *together (jointly)*

- Jointly train experts and combining classifier
- Trained through maximum likelihood estimation using EM
- Can also be trained as a Bayesian mixture (Waterhouse, MacKay, Robinson 1996)



Sequential Training: Boosting

- Classifiers trained sequentially

- Each classifier is trained given knowledge of the performance of previously trained classifiers: focus on hard examples

- Final classifier: weighted sum of component classifiers

Ack: Urtasun & Zeme

Boosting Justification

- Suppose you have a weak learner (a classifier that uses a very simple model) that can always get correct labels with $(0.5 + \varepsilon)$ probability when given a binary classification task (two labels)

- This seems like a weak assumption but…

- Can we apply this learning module many times to get a strong learner that gets close to zero error rate on the training data?

- (Freund & Shapire, 1996) theoretically showed how to do this and it actually led to an effective new learning procedure

Ack: Urtasun & Zeme

Boosting (Adaboost) Algorithm

- First train the base classifier on all the training data with equal importance weights

- Then re-weight the training data to emphasize the misclassified (or hard) cases and train a second model
  - How to re-weight the data?

- Keep training new models on re-weighted data

- Finally, use a weighted committee of all the models for the validation data
  - How to weight the models in the committee?

Ack: Urtasun & Zeme

Adaboost in Details

- Input: $\{x_i, t_i\}_{i=1,..,n}$, $t_i \in \{-1,1\}$ and a learning procedure f that outputs classification probabilities, Output: classifier $f^*(x)$
- Initialize example x_i weight: $w_i^{(1)} = 1/N \,, \quad \forall i$
- For $k=1,\ldots,K$
  - Learn classifier f_k by minimizing over

$$\mathrm{Err}_i = \sum_{i=1}^{n} w_i^{(k)} \mathbf{1}\{f_k(x_i) \neq t_i\}$$

  - Compute classifier coefficient

$$\alpha_k = \frac{1}{2} \log \frac{1 - \mathrm{Err}_k}{\mathrm{Err}_k}$$

  - Update observation weights

$$w_i^{(k+1)} = \frac{w_i^{(k)} \exp(-\alpha_k t_i f_k(x_i))}{\sum_{j=1}^{n} w_j^{(k)} \exp(-\alpha_k t_j f_k(x_j))}$$

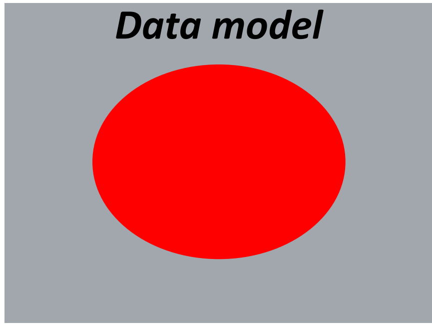
- Final classifier: $f^\star(x) = \mathrm{sign}\left(\sum_{k=1}^{K} \alpha_k f_k(x)\right)$

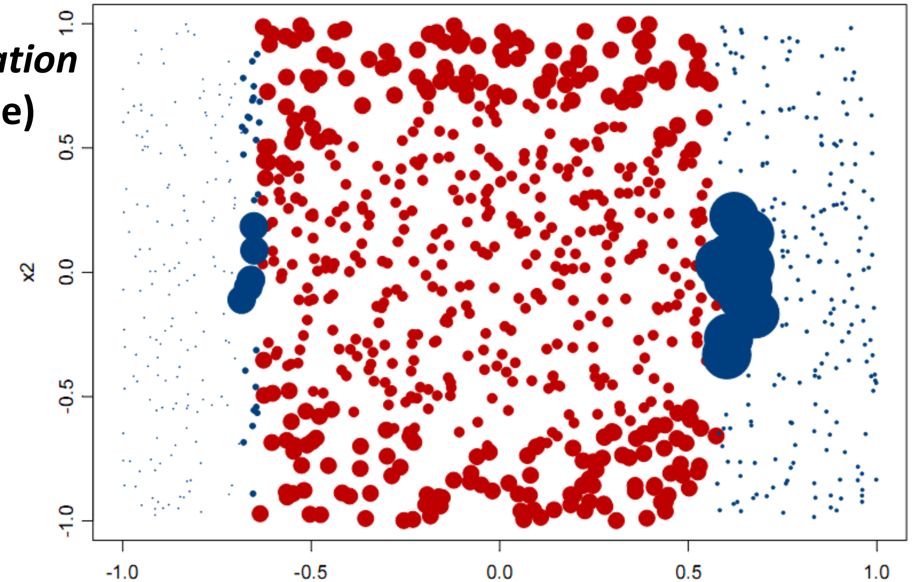
Slight Improvement over Adaboost Algorithm

- A more robust AdaBoost
  - Initially, set uniform weights on all the records
  - At each round
    - Create a **bootstrap** sample based on the weights
    - Train a classifier on the sample and apply it on the original training set
    - Observations that are wrongly classified will have their weights increased
    - Observations that are classified correctly will have their weights decreased
    - If the error rate is higher than 50% , start over
  - Final prediction is weighted average of all the classifiers with weight representing the training accuracy

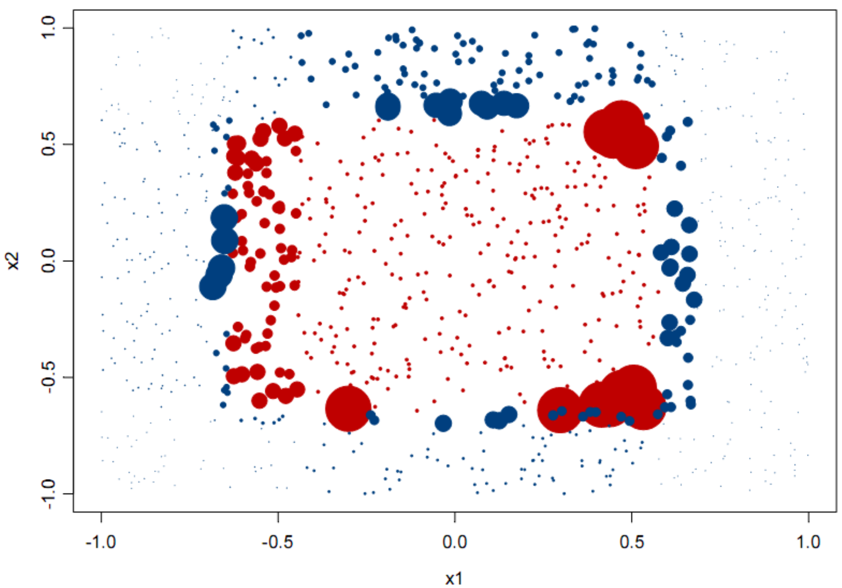
Ack: Gao, Fan, Han 2010

Classifications (colors) and
Weights (circle radius) after *1 iteration*
Of AdaBoost (3-leaf decision tree)

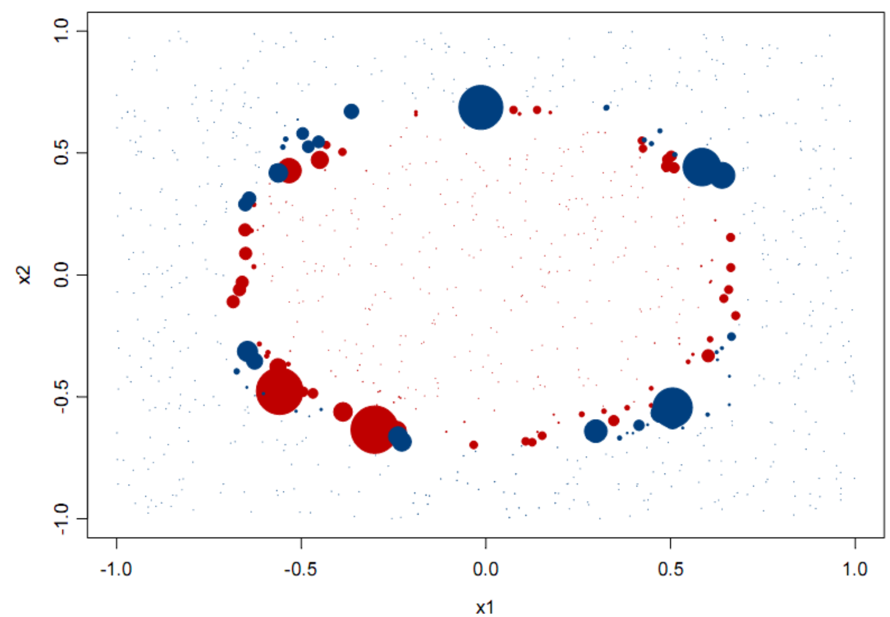




3 iterations



20 iterations



Ack: Elder, John. 2007.

Explaining Adaboost

- Theoretical Understanding
  - Among the classifiers of the form:

$$f^\star(x) = \sum_{k=1}^{K} \alpha_k f_k(x)$$

  - We seek to minimize the exponential loss function:

$$\sum_{i=1}^{n} \exp\left(-t_i f^\star(x_i)\right)$$

- Main Issue with Adaboost?
  - Not at all robust to mislabeling (noise in labels)
  - Weights of mislabeled observations keeps growing exponentially until classifier fits the noise
- Choice of classifiers f_k ?
  - Weak learners, very simple models (e.g., shallow decision tree)
  - Otherwise Adaboost easily “overfits” data

Side note about Boosting

Extensions:

- Zhu et al. (2005) “Multi-class AdaBoost” generalizes for K-class problems

State-of-the-art: Gradient Tree Boosting

Boosted trees > 50% of all Kaggle winning entries

All-in-one package (based on best practices):

Tianqi Chen, Carlos Guestrin, “XGBoost: A Scalable Tree Boosting System”, *KDD 2016*

Unsupervised Ensembles

Majority Voting

- General algorithm:

  1. Train multiple classifiers over same training data or same classifier over different training data (same classifiers give different answers when trained over different training data)

  2. Each votes on validation instance

  3. Take majority as classification

Strong assumption that consensus is better

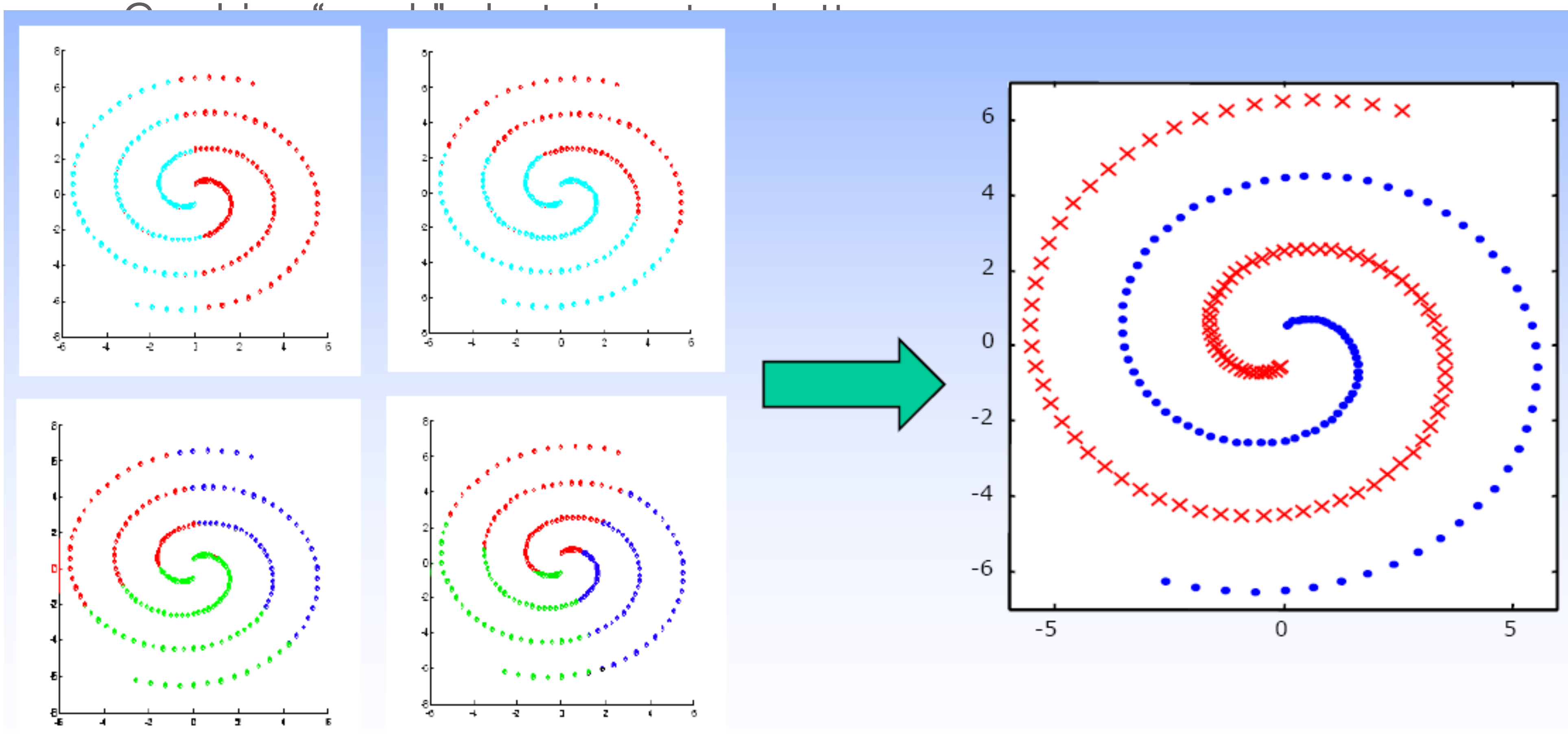
Clustering Ensemble

- Problem
  - Given an unlabeled data set $D=\{x_1,x_2,\ldots,x_n\}$
  - An ensemble approach computes:
    - A set of clustering solutions $\{C_1,C_2,\ldots,C_k\}$, each of which maps data to a cluster: $f_j(x)=m$
    - A unified clustering solutions f^* which combines base clustering solutions by their consensus
- Challenges
  - The correspondence between the clusters in different clustering solutions is unknown (label switching problem)
  - Unsupervised
  - Combinatorial optimization problem is NP-complete

Ack: Gao, Fan, Han 2010

Motivations

- Goal



[Punch,Topchy, Jain 2005]

Ack: Gao, Fan, Han 2010

An Example

base clustering models

objects

	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}$
v_1	1	1	1	1
v_2	1	2	2	2
v_3	2	1	1	1
v_4	2	2	2	2
v_5	3	3	3	3
v_6	3	4	3	3

These may not represent the same cluster!

The goal: get the consensus clustering

[Gionis, Mannila, Tsaparas 2007]

Ack: Gao, Fan, Han 2010

Hard Correspondence (1)

- Re-labeling+voting
  - Find the correspondence between the labels in the partitions and fuse the clusters with the same labels by voting
    [Dudoit, Fridlyand 2003; Dimitriadou, Weingessel, Homik 2001]

Re-labeling

Voting

	C_1	C_2	C_3
v_1	1	3	2
v_2	1	3	2
v_3	2	1	2
v_4	2	1	3
v_5	3	2	1
v_6	3	2	1

	C_1	C_2	C_3
v_1	1	1	1
v_2	1	1	1
v_3	2	2	1
v_4	2	2	2
v_5	3	3	3
v_6	3	3	3

C^*
1
1
2
2
3
3

Ack: Gao, Fan, Han 2010

Hard Correspondence (2)

- Details
  - Hungarian method to match clusters in two different clustering solutions
  - Match to a reference clustering or match in a pairwise manner
- Problems
  - In most cases, clusters do not have one-to-one correspondence

Soft Correspondence (1)

- **Notations**
  - Membership matrix $M_1, M_2, \ldots, M_k$
  - Membership matrix of consensus clustering M
  - Correspondence matrix $S_1, S_2, \ldots, S_k$
  - $M_i\, S_i = M$

	C_1	C_2	C_3
v_1	1	3	2
v_2	1	3	2
v_3	2	1	2
v_4	2	1	3
v_5	3	2	1
v_6	3	2	1

$$M_2 \qquad S_2 \qquad M$$

$$\begin{bmatrix} & & 1 \\ & & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

[Long, Zhang, Yu, 2005]

Soft Correspondence (2)

- Consensus function
  - Minimize disagreement $\min \sum_{j=1}^{k} \| M - M_j S_j \|^2$
  - Constraint 1: column-sparseness
  - Constraint 2: each row sums up to 1
  - Variables: M, S_1, S_2, … ,S_k $M = \frac{1}{k} \sum_{j=1}^{k} M_j S_j$
- Optimization
  - EM-based approach
  - Iterate until convergence
    - Update S using gradient descent
    - Update M as