Ensemble Learning

MAS DSE 220 Class

Natasha Balac, Ph.D.

Overview

- Theory
- Methods
  - Resampling Methods
    - Bagging
    - Randomization
    - Boosting
  - Hybrid Methods
    - Stacking
- Implementations

Concepts

- Occam's Razor
  - "among the theories that are consistent with the data, select the simplest one"
- Epicurus' Principle
  - Principle of Multiple Explanations
    - "keep all theories that are consistent with the data"
- The Condorcet Jury Theorem
  - "Aggregation of information from groups can results in improved decisions vs. an individual "expert" decision"

Ensemble-based Systems in Decision Making

- For many tasks, we often seek second opinion before making a decision, sometimes many more
  - Consulting different doctors before a major surgery
  - Reading reviews before buying a product
  - Requesting references before hiring someone
- We consider decisions of multiple experts in our daily lives
- Why not follow the same strategy in automated decision making?
- Multiple classifier systems, committee of classifiers, mixture of experts, ensemble based systems

Ensemble-based Classifiers

- Ensemble based systems provide favorable results compared to single-expert systems for a broad range of applications & under a variety of scenarios
- How to
  - generate individual components of the ensemble systems (base classifiers), and
  - how to combine the outputs of individual classifiers?
- Popular ensemble based algorithms
  - Bagging, boosting, AdaBoost, stacked generalization, and hierarchical mixture of experts
- Commonly used combination rules
  - Algebraic combination of outputs, voting methods, behavior knowledge space & decision templates

Why Ensemble Based Systems?

- Statistical reasons
  - A set of classifiers with similar training performances may have different generalization performances
  - Combining outputs of several classifiers reduces the risk of selecting a poorly performing classifier
- Large volumes of data
  - If the amount of data to be analyzed is too large, a single classifier may not be able to handle it; train different classifiers on different partitions of data
- Too little data
  - Ensemble systems can also be used when there is too little data; resampling techniques

Why Ensemble Based Systems?

- Divide and Conquer

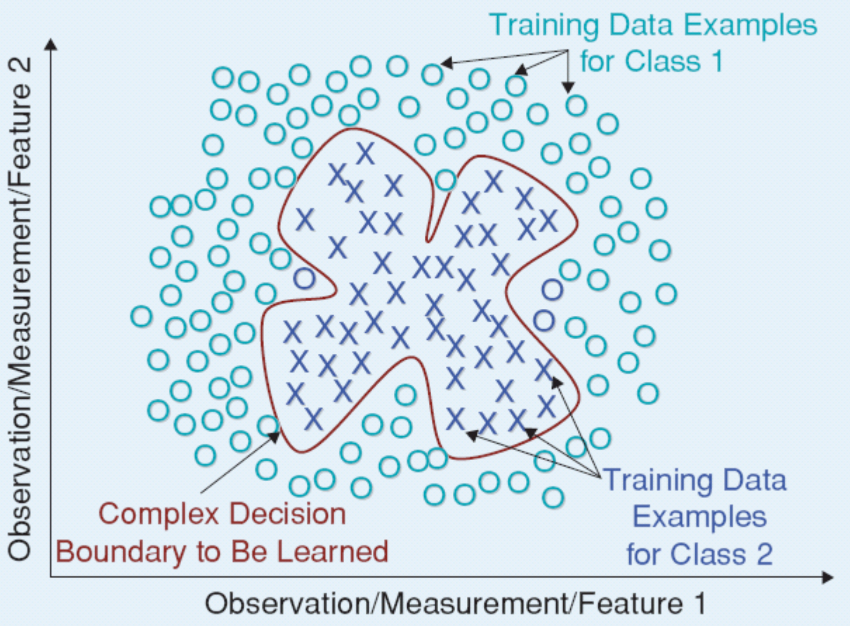


Figure 1. Complex decision boundary that cannot be learned by linear or circular classifiers.

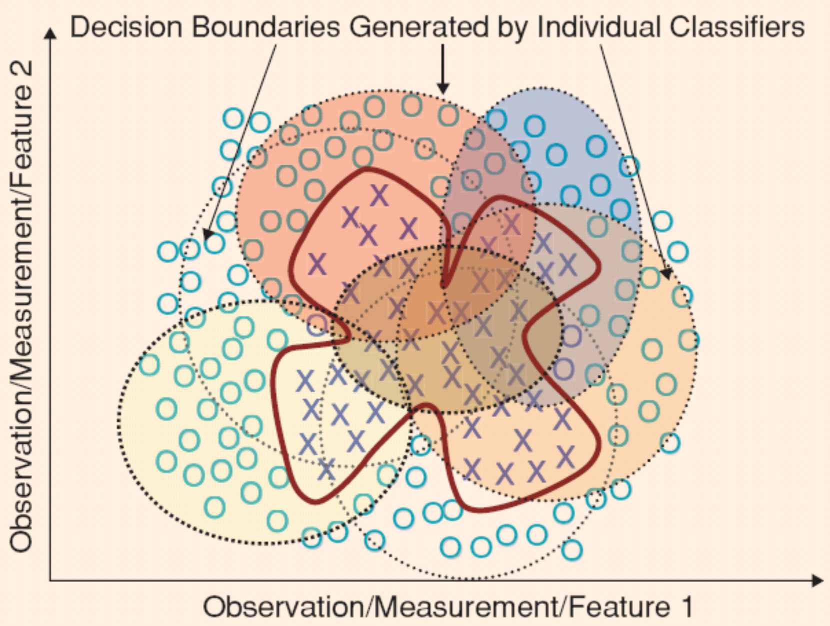


Figure 2. Ensemble of classifiers spanning the decision space.

Why Ensemble Based Systems?

- Data Fusion
  - Given several sets of data from various sources, where the nature of features is different (heterogeneous features), training a single classifier may not be appropriate (e.g., MRI data, EEG recording, blood test,..)
  - Applications in which data from different sources are combined are called data fusion applications
  - Ensembles have successfully been used for fusion
- All ensemble systems must have two key components:
  - Generate component classifiers of the ensemble
  - Method for combining the classifier outputs

Brief History of Ensemble Systems

- Dasarathy and Sheela (1979) partitioned the feature space using two or more classifiers
- Schapire (1990) proved that a strong classifier can be generated by combining weak classifiers through boosting; predecessor of AdaBoost algorithm
- Two types of combination:
  - classifier selection
    - Each classifier is trained to become an expert in some local area of the feature space; one or more local experts can be nominated to make the decision
  - classifier fusion
    - All classifiers are trained over the entire feature space; fusion involves merging the individual (weaker) classifiers to obtain a single (stronger) expert of superior performance

Ensemble Learning Approach

- Basic idea
  - Create multiple models
    - Build different “experts” by diversification
    - Select models that maximizes some performance criterion on a test set
  - Combine models into a single decision
    - Trained ensemble, represents a single hypothesis
  - Result
    - Improve reliability
    - Improve predictive performance

Ensemble Shortcomings

- Models are only as good as their “experts”
  - Divergent ideas within a committee has the potential to strengthen an individual recommendation
  - ensembles tend to yield better results when there is a significant diversity among the models

Why Ensemble Pros and Cons

- Supervised Learning Technique
- Advantage
  - often improves predictive performance and reliability
- Disadvantage
  - Produces output that is very hard to analyze
    - Loss of Interpretability: which factors contribute to the improved decision
    - Learning systems or performance system (vs. explanation system)
  - Requires additional compute cycles

Key Criteria

- Diversify
  - Decentralization
    - Specialized, local knowledge (colloquial wisdom)
  - Diversity of Opinion
    - private information, “even if its just and eccentric interpretation of the known facts”
  - Independence
    - opinions aren’t determined by the opinions of those around them
- Combine
  - Mechanisms for turning private judgments into collective decision

Key Criteria: How

- Diversify
  - Resampling : vary the data used to train a given algorithm
    - Bagging : "bootstrap aggregation"
    - Randomization: bagging + "variable resampling"
    - Boosting : resampling through adaptive weighting
  - Hybrid learning : vary the algorithms trained on a given dataset
    - Stacking : "stacked generalization“
- Combine
  - Static : integration procedure is fixed, such as vote
    - retain the majority or mean/median of the individual predictions
  - Dynamic : base predictions are combined using an adaptive procedure, or meta-learning

Creating Ensemble Models

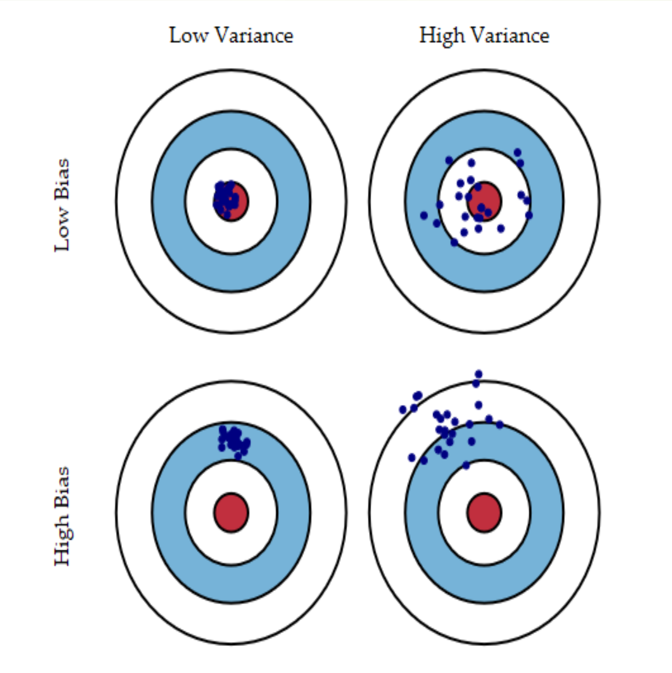
- Given
  - Training data set [D]
    - Labeled for supervised learning
  - Collection of inductive learning algorithms
- Create Diverse Models:
  - Vary Data
  - Vary Model Parameters
  - Vary Methods
- Combine Models
  - Dynamic or static procedures
- Goal
  - Reduce Variance
  - Reduce Bias

Creating An Ensemble

- Two questions:
  - How will the individual classifiers be generated?
  - How will they differ from each other?
- Answer determines the diversity of classifiers & fusion performance
- Seek to improve ensemble diversity by some heuristic methods

Bias-Variance Decomposition

- Bias: the difference between the expected (or average) prediction of the model and the actual value
- Variance: the variability of a model prediction for any given data point
- Total Expected Error = Bias + Variance



Overview

- Theory
- Methods
  - Resampling Methods
    - Bagging
    - Randomization
    - Boosting
  - Hybrid Methods
    - Stacking
- Implementations

Bagging

- Bagging, short for Bootstrap Aggregating, is one of the earliest ensemble based algorithms
- It is also one of the most intuitive and simplest to implement, with a surprisingly good performance

Bagging

- Create bootstrapped replicas of the training data
- Large number of (~200) training subsets are randomly drawn - with replacement - from the entire training data
- Each resampled training set is used to train a different classifier of the same type
- Individual classifiers are combined by taking a majority vote of their decisions
- Bagging is appealing for small training set; relatively large portion of the samples is included in each subset

Bagging: Method

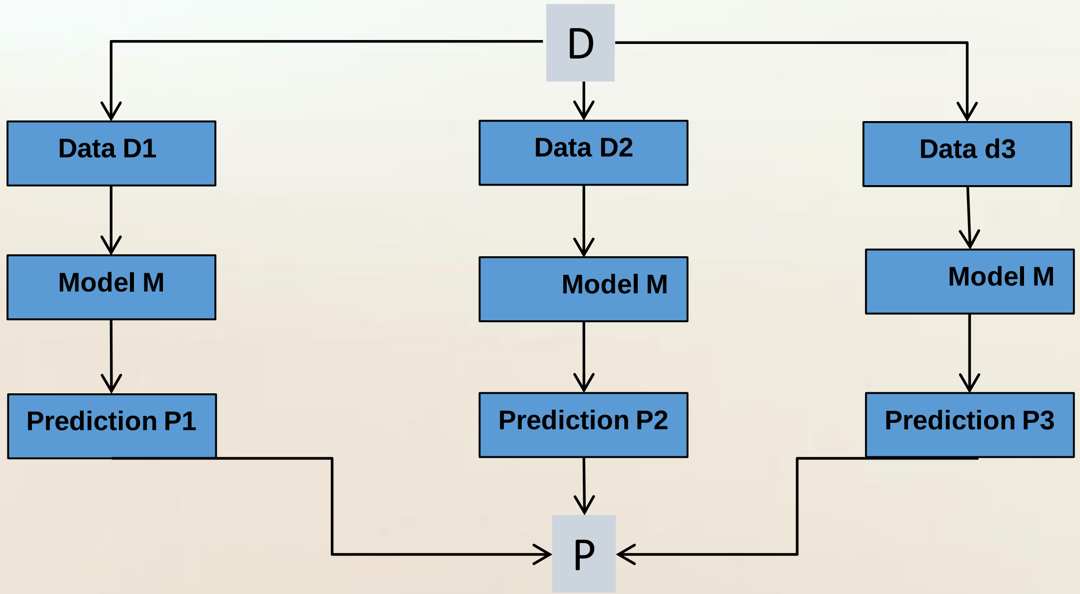
- Bootstrap Aggregation Theory
  - Simplest way of combing predictions
  - Combine multiple Independent models, of same type, using vote with equal weight
- Strategy
  - Diversify data sets via bootstrapping
  - Create multiple models build from each data set
  - Combining decisions of models
    - Using static “voting” (classification) or averaging/mean(regression)

Bootstrapping Review

- Method for estimating the generalization error of a learning method
- Resampling
  - Instances are randomly sampled with replacement to create new sample set to be applied to classifier to create model

Bagging Algorithm

- Sample several training sets (D1, D2, D3)
  - size n from Data set D
  - using bootstrapping
- Build a classifier for each training set (M)
- Combine predictions (P)



Sampling with Replacement

- Random & overlapping training sets to train three classifiers; they are combined to obtain a more accurate classification

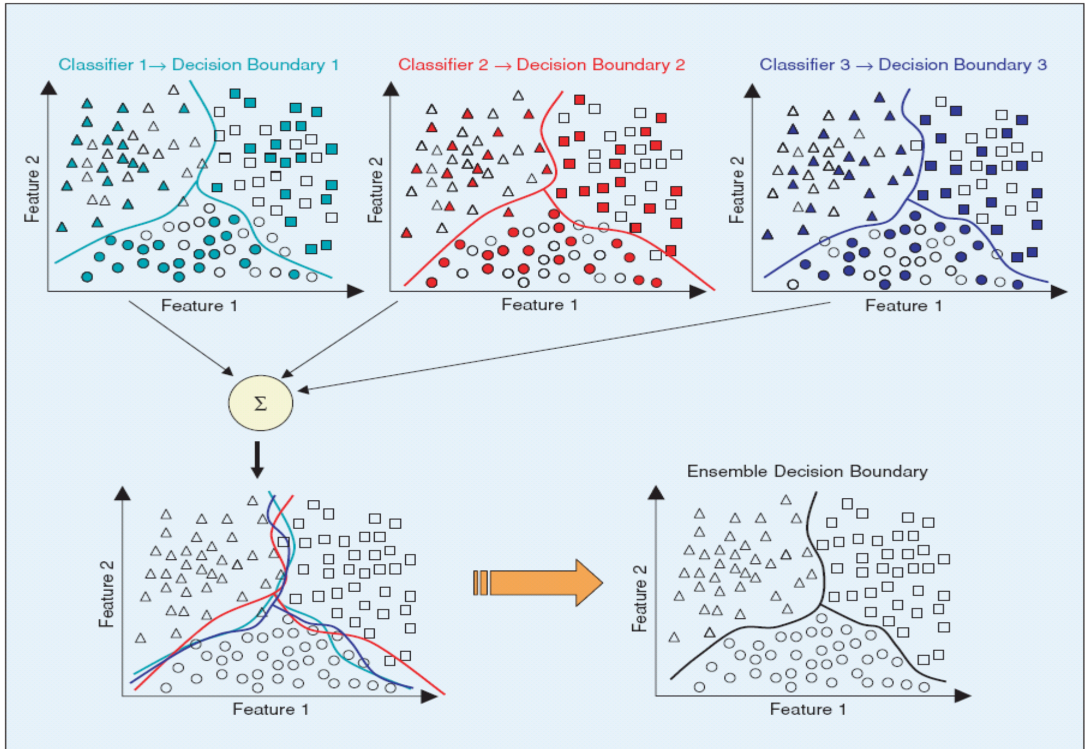


Figure 3. Combining classifiers that are trained on different subsets of the training data.

Bagging Pseudocode

Algorithm : Bagging

input:

Training data S with correct labels $\omega_i \in \Omega = \{\omega_1, \cdots, \omega_C\}$ representing C classes

Weak learning algorithm **WeakLearn**,

Integer T specifying number of iterations.

Percent (of fraction) F to create bootstrapped training data

Do $t = 1, \cdots, T$

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

End

Test : Simple Majority Voting - Given unlabled instance x

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

Variations of Bagging

- Random Forests
  - Generally constructed from decision trees
  - A random forest is created from individual decision trees, whose training parameters vary randomly
  - Such parameters can be bootstrapped replicas of the training data, as in bagging
  - But they can also be different feature subsets as in random subspace methods

Bagging Summary

- Uses Bootstrap resampling
  - Highly overlapping training sets
- Bagging reduces error due to variance
  - Reducing the overall expected error
  - Usually more classifiers/models the better
- Good with noisy data
  - Avoids over fitting
- Good with unstable learners
- Results are usually better then individual classifier

Randomization

- Randomization
  - Modifies learning algorithm
  - Can be applied to stable learners
- Randomization Implementations
  - Rotation Forests
    - Ensembles using rotated random subspaces
  - Random forests
    - Bag ensembles of random trees
  - Random Committee
    - Ensembles using different random number seeds
  - Random Subspace
    - Ensembles using rotated random subspaces

Boosting: Method

- Theory
  - Construct Strong Classifier by weighting voting of the weak classifiers
    - Strong learners are very difficult to construct
    - Constructing weaker learners is relatively easy
- Strategy
  - Construct Weak Classifier
    - Diversify using sequential adaptive resampling by weighting
  - “Boost” Weak Classifier to a strong learner
  - Combine Weak Classifiers
    - Integrate using weighted voting

Boosting

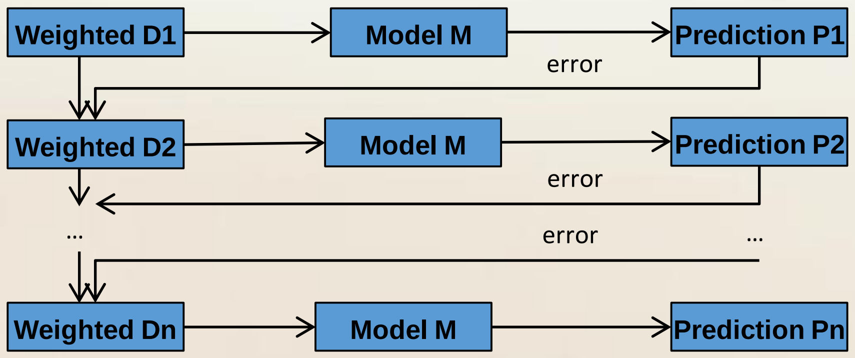
- Boost the performance of a weak learner to the level of a strong one
- Boosting creates an ensemble of classifiers by resampling the data
- Classifiers are combined by majority voting
  - resampling is strategically geared to provide the most informative training data for each consecutive classifier
- Boosting creates three weak classifiers:
  - First classifier C1 is trained with a random subset of the available training data
  - Training set for second classifier C2 is chosen as the most informative subset, given C1; half of the training data for C2 is correctly classified by C1, other half is misclassified by C1
  - Third classifier C3 is trained on instances on which both C1 & C2 disagree

Boosting

- Create classifier using training dataset
- Score each data point, indicating incorrect decisions or errors
- Retrain, focusing on incorrect decisions
- Repeat
- Final Prediction is weighted average of all the models

Boosting

- Construct Weak Classifiers
  - Using different data distributions
    - Start with uniform weight
    - Iterate through method
      - Increase weights of the examples which are not correctly learned
      - Decrease weights of example which are correctly learned by the weak learner
  - Focus on difficult example what are not correctly classified in previous step



AdaBoost

- AdaBoost (1997) is a more general version of the boosting algorithm; AdaBoost.M1 can handle multiclass problems
- AdaBoost generates a set of hypotheses (classifiers), and combines them through weighted majority voting of the classes predicted by the individual hypotheses
- Hypotheses are generated by training a weak classifier; samples are drawn from an iteratively updated distribution of the training set
- This distribution update ensures that instances misclassified by the previous classifier are more likely to be included in the training data of the next classifier
- Consecutive classifiers are trained on increasingly hard-to-classify samples

AdaBoost

- A weight distribution $D_t(i)$ on training instances x_i , $i=1,\dots,N$ from which training data subsets S_t are chosen for each consecutive classifier (hypothesis) h_t
- A normalized error is then obtained as β_t , such that for

  $0<\varepsilon_t<1/2$, they have $0<\beta_t<1$
- Distribution update rule:
  - The distribution weights of those instances that are correctly classified by the current hypothesis are reduced by a factor of β_t , whereas the weights of the misclassified instances are unchanged.
  - AdaBoost focuses on increasingly difficult instances
- AdaBoost raises the weights of instanced misclassified by h_t , and lowers the weights of correctly classified instances
- AdaBoost is ready for classifying unlabeled test instances. Unlike bagging or boosting, AdaBoost uses the weighted majority voting
- $1/\beta_t$ is therefore a measure of performance, of the t_{th} hypothesis and can be used to weight the classifiers

AdaBoost.M1

Algorithm AdaBoost.M1

Input:

Sequence of N examples $S = [(x_i, y_i)], i = 1, \cdots, N$ with labels $y_i \in \Omega, \Omega = \{\omega_1, \cdots \omega_C\}$;

Weak learning algorithm **WeakLearn**;

Integer T specifying number of iterations.

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

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

1. Select a training data subset S_t, draw from the distribution D_t.
2. Train **WeakLearn** with S_t, receive hypothesis h_t.
3. Calculate the error of

$$h_t : \varepsilon_t = \sum_{i: h_t(x_i) \neq y_i} D_t(i)$$

If $\varepsilon_t > 1/2$, **abort**.

4. Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$.
5. Update distribution

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

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

Test - Weight Majority Voting : Given an unlabeled instance x.

1. Obtain total vote received by each class

$$V_j = \sum_{t: h_t(x) = \omega_j} log \frac{1}{\beta_t}, j = 1, \cdots, C.$$

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

AdaBoost.M1

- AdaBoost algorithm is sequential; classifier (CK-1) is created before classifier CK

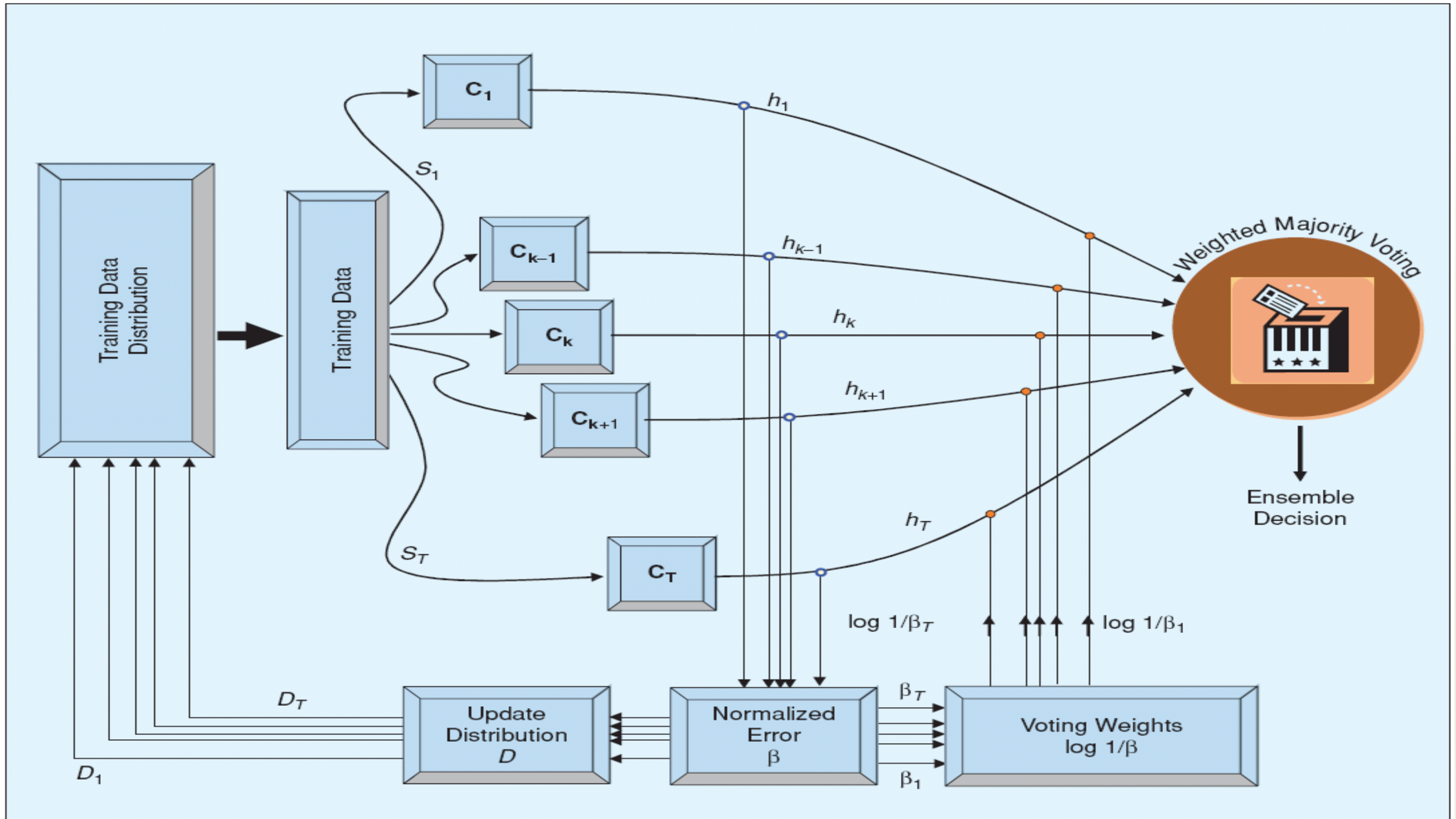
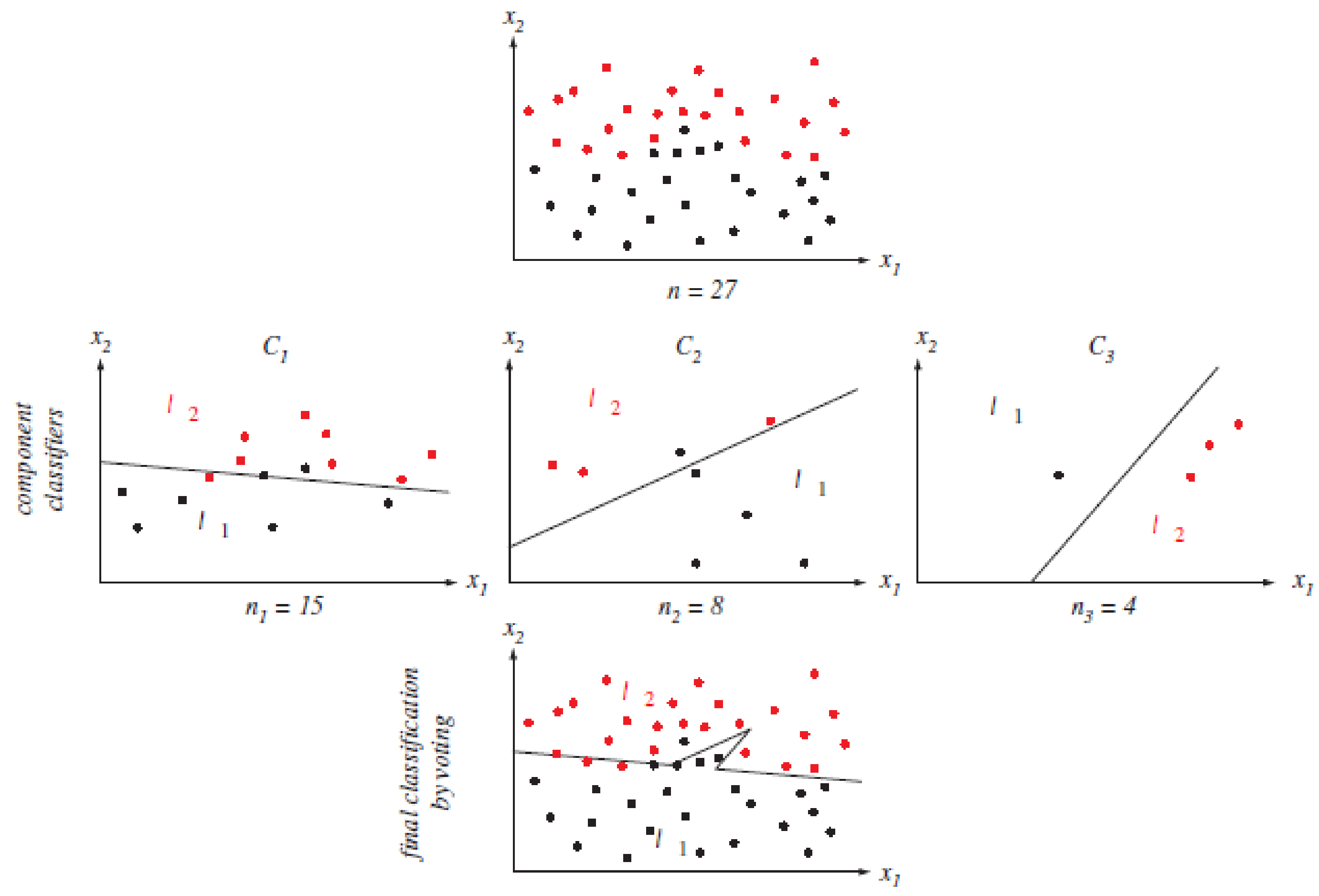


Figure 9. The AdaBoost.M1 algorithm.

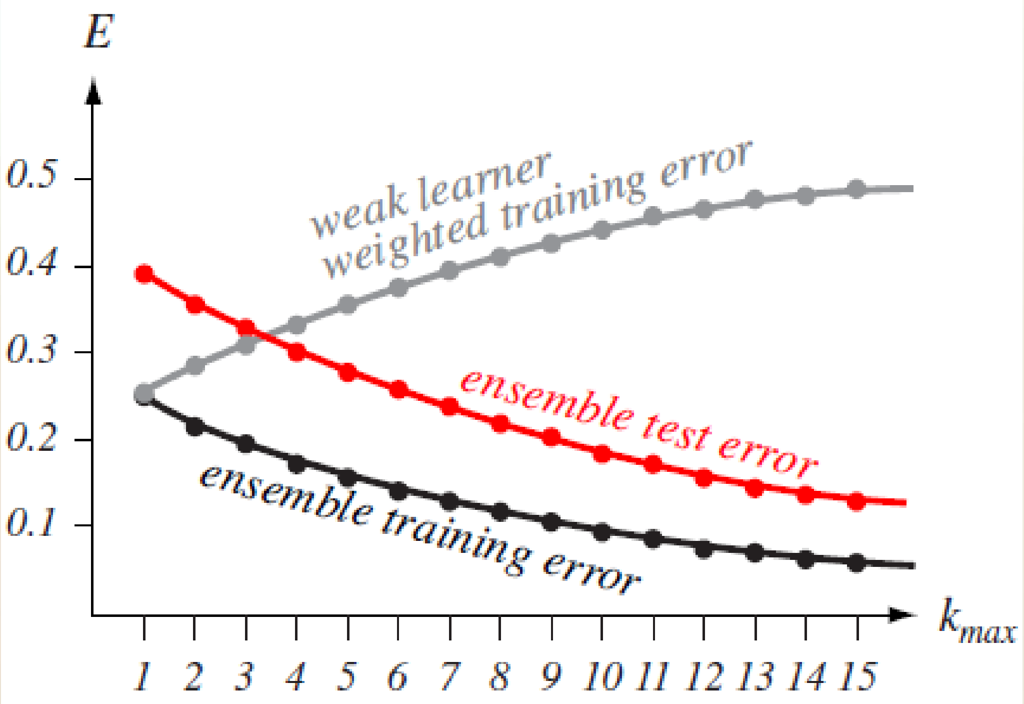
Boosting



x_2
x_1
$n = 27$
component classifiers
C_1
$n_1 = 15$
C_2
$n_2 = 8$
C_3
$n_3 = 4$
final classification by voting

AdaBoost Performance

- In most practical cases, the ensemble error decreases very rapidly in the first few iterations, and approaches zero or stabilizes as new classifiers are added



MART
Multiple Additive Regression Trees

- MART is a Boosting algorithm for regression

- Input: a learning sample $\{(x_i,y_i): i=1,\ldots,N\}$
- Initialize
  - $\hat{y}_0(x) = 1/N$ åi y_i ; $r_i=y_i$, $i=1,\ldots,N$
- For t=1 to T:
  - For i=1 to N, compute the residuals
  - $r_i \leftarrow r_i - \hat{y}_{t\text{-}1}(x_i)$
  - Build a regression tree from the learning sample $\{(x_i,r_i): i=1,\ldots,N\}$
- Return the model $\hat{y}(x)= \hat{y}0(x)+\hat{y}1(x)+\ldots+\hat{y}T(x)$

Boosting Methods

- Many types of boosting algorithms
- Boosting decision/regression trees improves their accuracy often dramatically
- For Boosting to work, the models need not to be perfect on the learning sample
- With trees, there are two possible strategies:
  - Use pruned trees (pre-pruned or post-pruned by cross-validation)
  - Limit the number of tree tests (and split first the most impure nodes)
- $\Rightarrow$ bias/variance tradeoff with respect to the tree size

Boosting: Summary

- Sensitive to noise
  - when base learners misclassify noisy examples, weights increase, causes over fitting to noise
- The power of boosting comes from adaptive resampling
- Like bagging, boosting reduces variance
- Reduces bias focusing on learning “hard cases”
- Results usually better then individual classifier or bagging

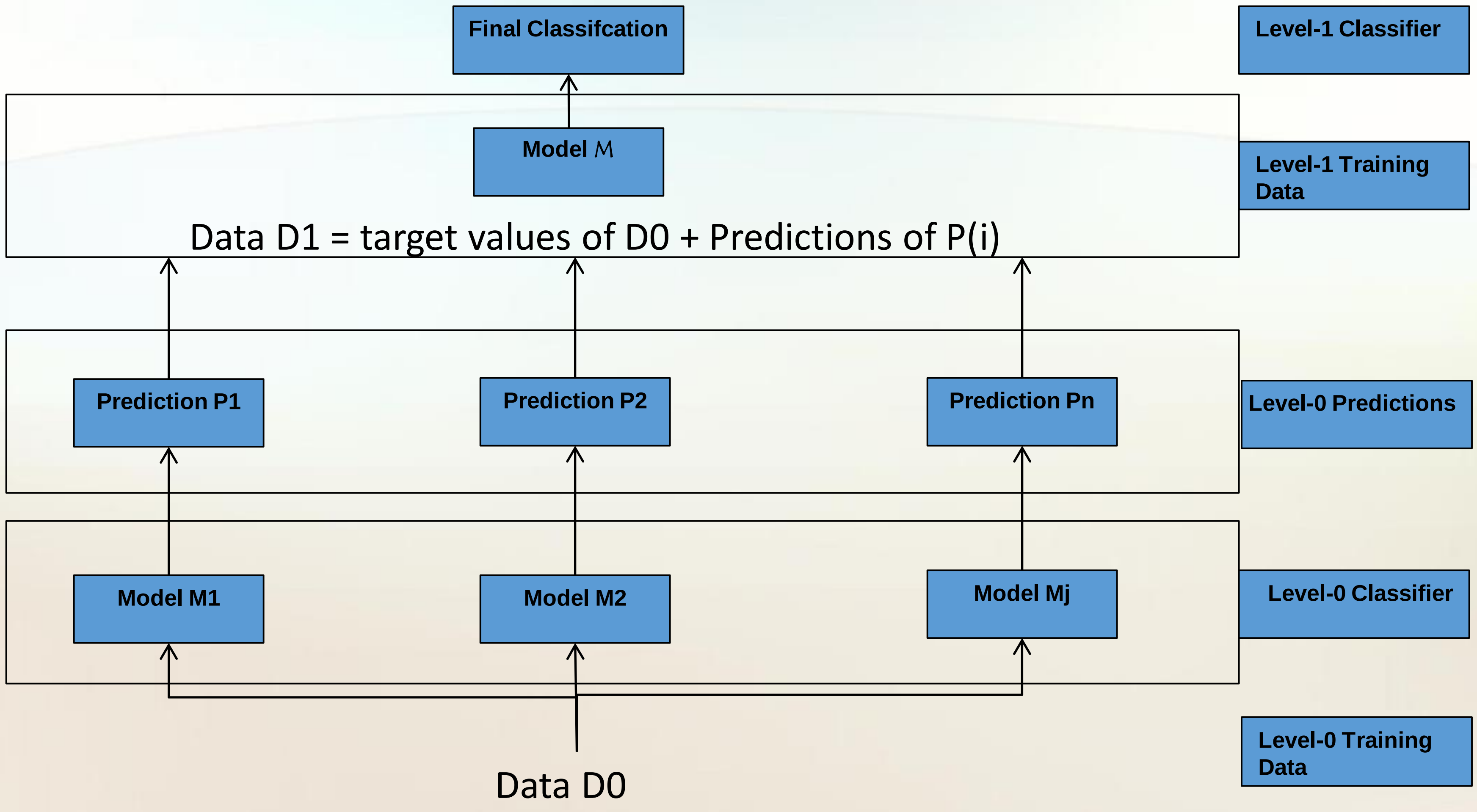
Stacking
Stacked Generalization

- Theory
  - Train on multiple models of different type
  - Combines multiple different learning algorithm models using a meta learner
- Strategy
  - Uses Meta learners to identify “reliable” classifiers

Stacking Pseudocode

- Train level-0, or base, models as usual
  - Train multiple learners (Level -0 /base learners)
    - Each uses subsample of D
- Train ‘combiner’ on validation segment D (Level-1 Model)
  - Level-1 data built from predictions of level-0 models on remainder of set
  - Level-1 Generalizers are models trained on level-1 data

Stacking



Final Classifcation
Level-1 Classifier
Model M
Level-1 Training Data
Data D1 = target values of D0 + Predictions of P(i)
Prediction P1
Prediction P2
Prediction Pn
Level-0 Predictions
Model M1
Model M2
Model Mj
Level-0 Classifier
Level-0 Training Data
Data D0

Stacked Generalization

- An ensemble of classifiers is first created, whose outputs are used as inputs to a second level meta-classifier to learn the mapping between the ensemble outputs and the actual correct classes
- C1, …,CT are trained using training parameters θ1 through θT to output hypotheses h1 through hT
- The outputs of these classifiers and the corresponding true classes are then used as input/output training pairs for the second level classifier, CT+1

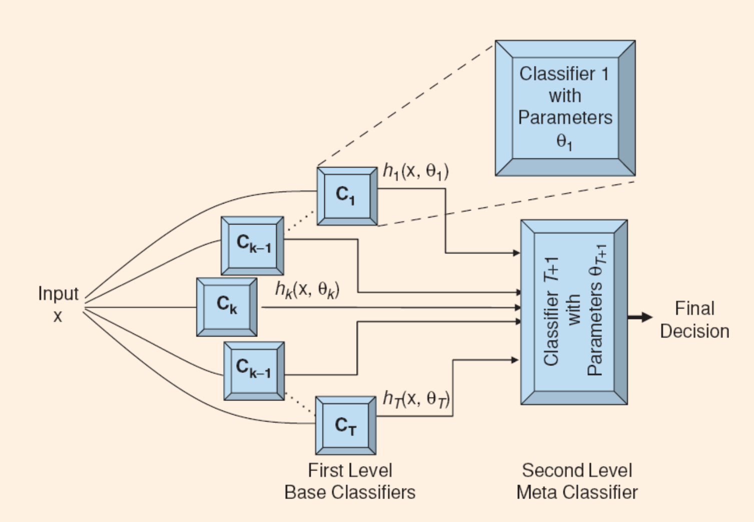


Figure 11. Stacked generalization.

Stacking Summary

- Stacking is a meta-learner
- Difficult to analyze theoretically
- Use probabilities as base learners if possible
  - it's better to use those as input to meta learner
- Choosing Level -1 Learners
  - In principle, any learning scheme
  - David Wolpert: "relatively global, smooth" model
    - Base learners do most of the work
    - Reduces risk of over fitting

Weka Implementations

- Bagging: Bagging
  - MetaCost
- Randomization
  - Random Committee
  - Random Forest
- Boosting
  - AdaBoostM1
  - MultiBoostAB
- Stacking: Stacking
  - StackingC

R Implementations

- Bagging:
  - Ipred
  - Adabag: Adaboost and Bagging
- Randomization
  - randomForest: Random Forest
  - Party: RF with faster tree growing
- Boosting
  - Ada: (AdaBoost + Friedmans mods)
  - Qbm: (Stochastic Gradient boosting)
  - Mboost: (Boosting applied to glm, gam)
- Stacking: Stacking
  - StackingC

Python Implementations

- Sklearn.ensemble Methods

ensemble.AdaBoostClassifier([...])	An AdaBoost classifier
ensemble.AdaBoostRegressor([base_estimator, ...])	An AdaBoost regressor
ensemble.BaggingClassifier([base_estimator, ...])	A Bagging classifier
ensemble.BaggingRegressor([base_estimator, ...])	A Bagging regressor
ensemble.ExtraTreesClassifier([...])	An extra-trees classifier
ensemble.ExtraTreesRegressor([n_estimators, ...])	An extra-trees regressor
ensemble.GradientBoostingClassifier([loss, ...])	Gradient Boosting for classification
ensemble.GradientBoostingRegressor([loss, ...])	Gradient Boosting for regression
ensemble.RandomForestClassifier([...])	A random forest classifier
ensemble.RandomTreesEmbedding([...])	An ensemble of totally random trees
ensemble.RandomForestRegressor([...])	A random forest regressor

Summary

- Ensemble Learning
  - Ensemble systems are useful in practice
  - Techniques to combine models to improve predictive performance
  - The error of an ensemble is less than the error of an individual model if:
    - Models are diverse and independent
    - Models performance is slightly better then random (err < 0.5)
- No single ensemble generation algorithm or combination rule is universally better than others
- Effectiveness on real world data depends on the classifier diversity and characteristics of the data