Data Mining II: Advanced Methods and Techniques

Lecture 6

Natasha Balac, Ph.D.

Machine Learning Methods

- So far we examined wide variety:
  - Decision trees/rules/tables
  - Instance-based schemes
  - Numeric prediction
  - Clustering
  - Neural networks
- There is more to data mining than selecting a method and running it over data
  - Many parameter choices
  - Very data set dependent
  - Overfitting

Engineering the input and output

- Scheme/parameter selection
  - Selection process should be treated as part of the learning process
- Modifying the input: attribute selection, discretization, data cleansing, transformations
- Modifying the output: combining classification models to improve performance
  - Bagging, boosting, stacking, error-correcting output codes

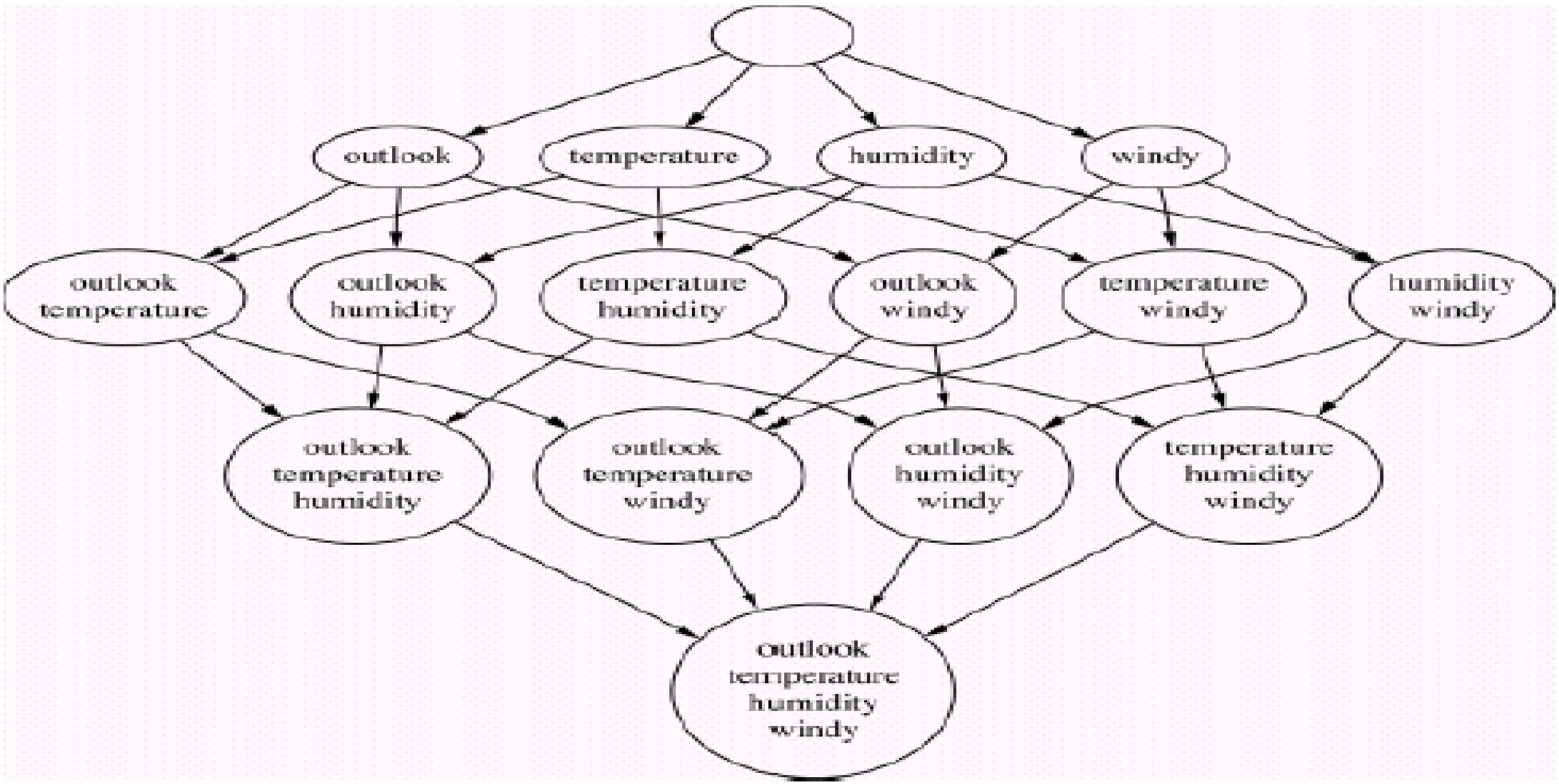
Attribute selection

- Adding a random (irrelevant) attribute can significantly degrade C4.5's performance
  - Problem: attribute selection based on smaller and smaller amounts of data
- IBL is also very susceptible to irrelevant attributes
  - Number of training instances required increases exponentially with number of irrelevant attributes
- Naïve Bayes doesn't does not suffer from this, but does from redundant
- Manual selection not easy
- Automatic methods look promising

Filter vs. Wrapper Method

- Wrapper – learning wrapped into the selection procedure
- Filter method
  - assessment based on general characteristics of the data
- One method
  - find subset of attributes enough to separate all instances
- Another method
  - use different learning scheme (C4.5, 1R) to select attributes
- IBL-based attribute weighting techniques can also be used
  - but can’t find redundant attributes

Attribute subsets for weather data set



Attribute selection
Searching the space of attributes

- Number of possible attribute subsets is exponential in the number of attributes
- Common greedy approach
  - forward selection
  - backward elimination
- More sophisticated search schemes
  - Bidirectional search
  - Best-first search: can find the optimum solution
  - Beam search: approximation to best-first search
  - Genetic algorithms

Scheme-specific selection

- Wrapper approach - attribute selection implemented as wrapper around learning scheme
  - Evaluation criterion
    - cross-validation performance
- Time consuming
  - adds factor k^2 even for greedy approaches with k attributes
    - Linearity in k requires prior ranking of attributes
- Essential for learning decision tables
  - Can be efficient for DTs and Naïve Bayes
- Selective Naïve Bayes
  - Naïve Bayes + forward selection

Discretizing

- Some methods deal only with nominal – some perform better
- Can be used to avoid making normality assumption in Naïve Bayes and Clustering
- Simple discretization scheme is used in 1R
- C4.5 performs local discretization
- Global discretization can be advantageous because it’s based on more data
  - Learner can be applied to discretized attribute or
  - It can be applied to binary attributes coding the cut points in the discretized attribute

Outlook	Temp.	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Attribute	Rules	Errors	Total errors
Outlook	Sunny → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	
Temperature	Hot → No*	2/4	5/14
	Mild → Yes	2/6	
	Cool → Yes	1/4	
Humidity	High → No	3/7	4/14
	Normal → Yes	1/7	
Windy	False → Yes	2/8	5/14
	True → No*	3/6	

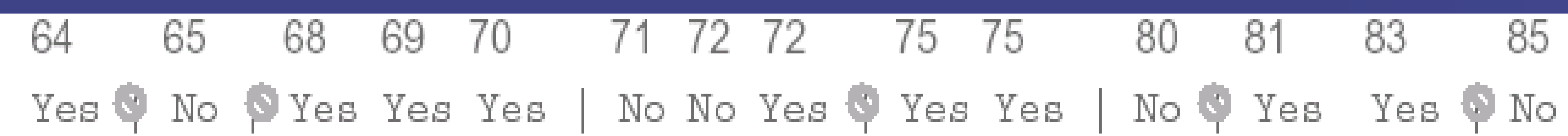
Numeric attributes

- Discretization: the range of the attribute is divided into a set of intervals
- Instances are sorted according to attribute's values
- Breakpoints placed where the class changes
- Example: temperature from weather data

```
64    65    68    69    70    71    72    72    75    75    80    81    83    85
Yes | No  | Yes   Yes   Yes | No    No    Yes | Yes   Yes | No  | Yes   Yes | No
```

Overfitting

- Discretization very sensitive to noise
- A single instance with an incorrect class label will most likely result in a separate interval
- Also: time stamp attribute will have zero errors
- Simple solution: enforce minimum number of instances in majority class per interval
- Weather data example (with minimum set to 3):



Final result

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

Attribute	Rules	Errors	Total errors
Outlook	Sunny → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	
Temperature	$\le$ 77.5 → Yes	3/10	5/14
	> 77.5 → No*	2/4	
Humidity	$\le$ 82.5 → Yes	1/7	3/14
	> 82.5 and $\le$ 95.5 → No	2/6	
	> 95.5 → Yes	0/1	
Windy	False → Yes	2/8	5/14
	True → No*	3/6	

Unsupervised Discretization

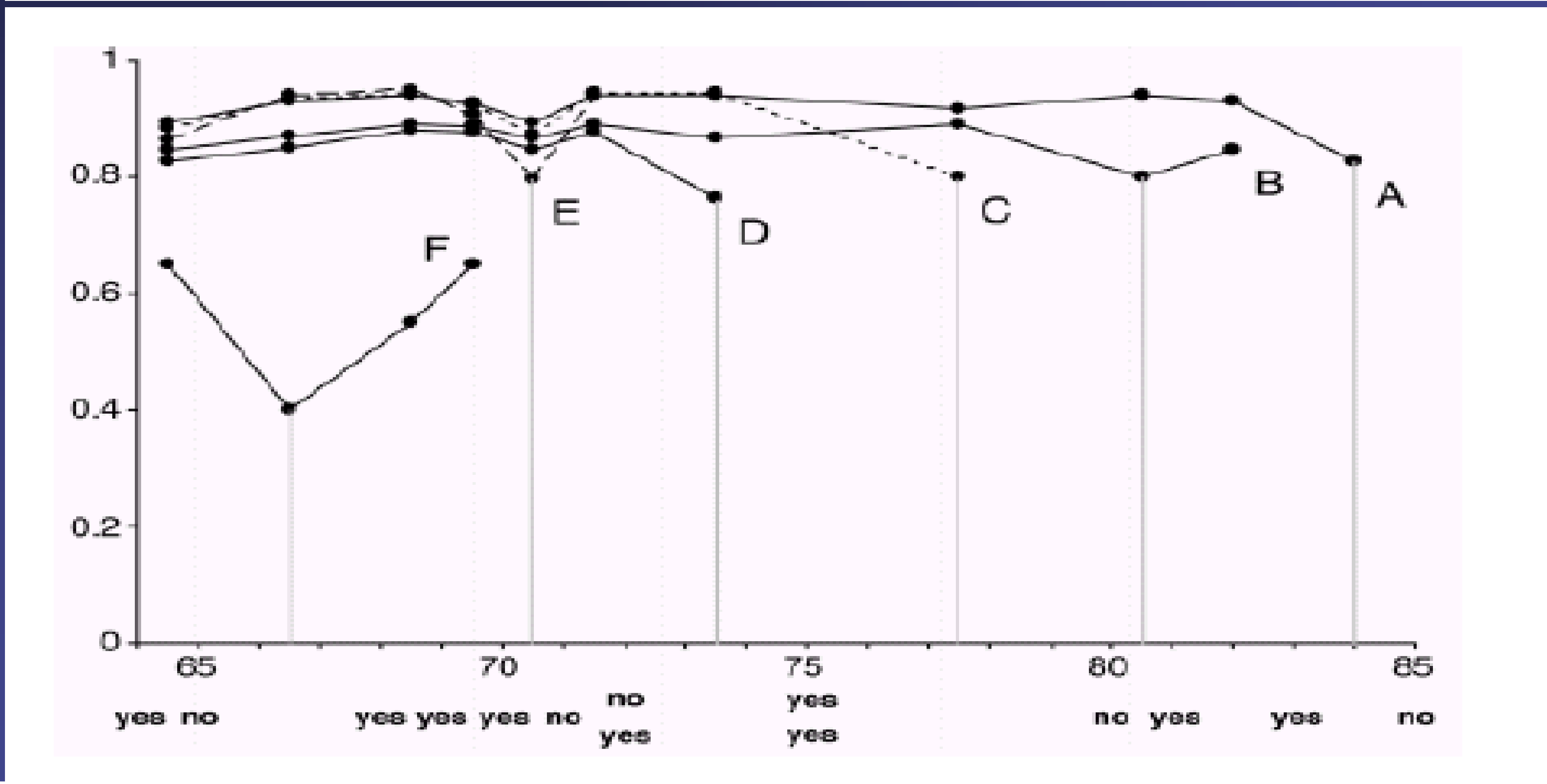
- Unsupervised discretization generates intervals without looking at class labels
  - Only possible way when clustering
- Two main strategies:
  - Equal-interval binning
  - Equal-frequency binning - called histogram equalization
- Inferior to supervised schemes in classification tasks

Entropy-based discretization

- Supervised method that builds a decision tree with pre-pruning on the attribute being discretized
  - Entropy used as splitting criterion
  - Minimum Description Length Principle used as stopping criterion
- Minimize the size of the "theory" plus the size of the information necessary to specify all the exceptions relative to the theory
- Application of MDLP:
  - "Theory" is the splitting point (log2[N-1] bits) plus class distribution in each subset
  - DL before/after adding splitting point is compared

Example: temperature attribute

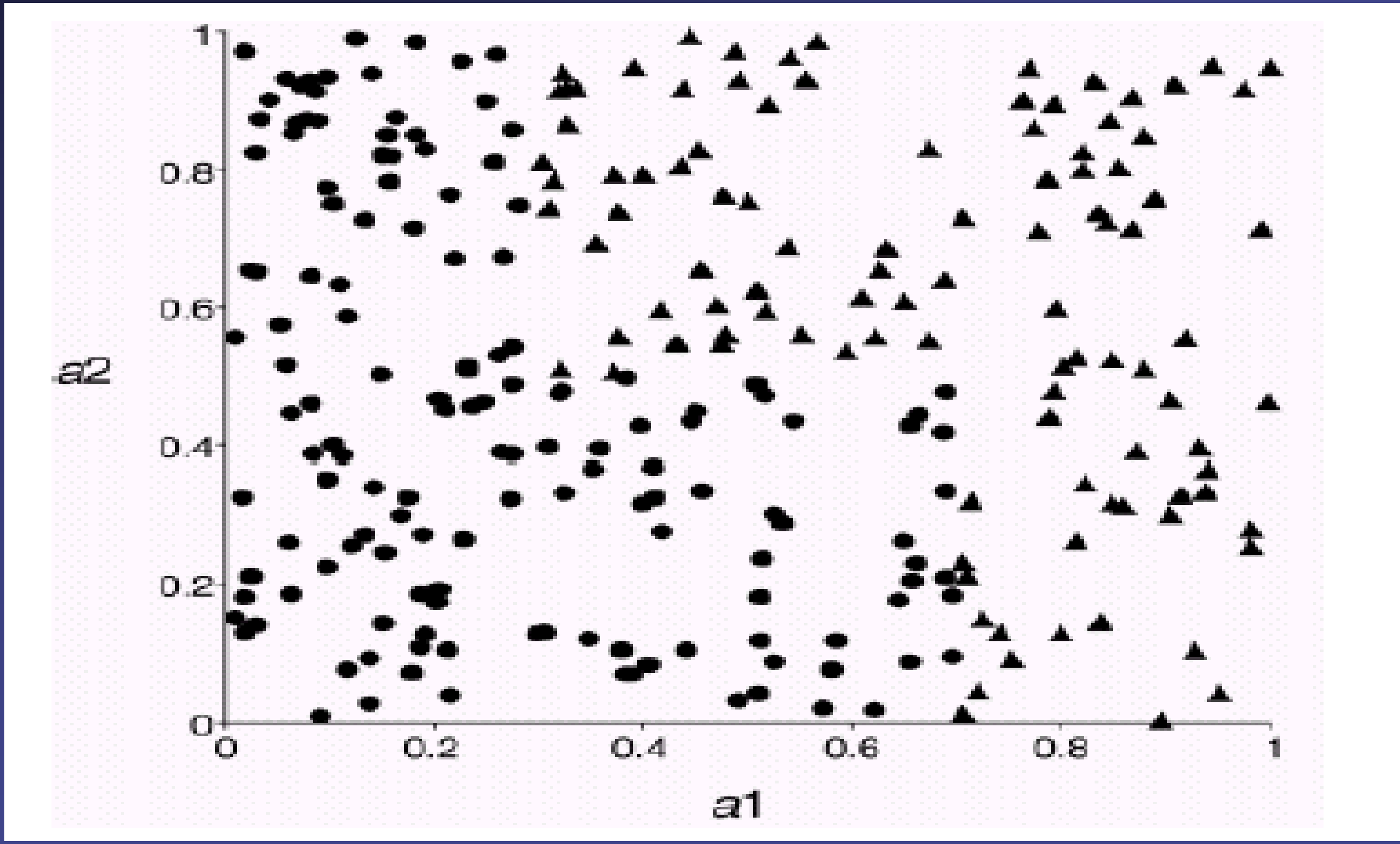
64 65 • 68 69 70 • 71 72 72 • 75 • 80 • 81 83 • 85
Y no F Y Y Y E no no Y D YY C no B Y Y A no



Other Discretization Methods

- Top-down procedure can be replaced by bottom-up method
- MDLP can be replaced by probability test
- Dynamic programming can be used to find optimum k-way split for given additive criterion
  - Requires time quadratic in number of instances if entropy is used as criterion
  - Can be done in linear time if error rate is used as evaluation criterion

Error-based vs. entropy-based



Converting Discrete to Numeric Attributes

- Scheme used by IB1
  - indicator attributes
- Doesn’t make use of potential ordering information
- M5’ generates ordering of nominal values and codes ordering using binary attributes
  - Strategy can be used for any attribute for which values are ordered
- In general subsets of attributes coded as binary attributes

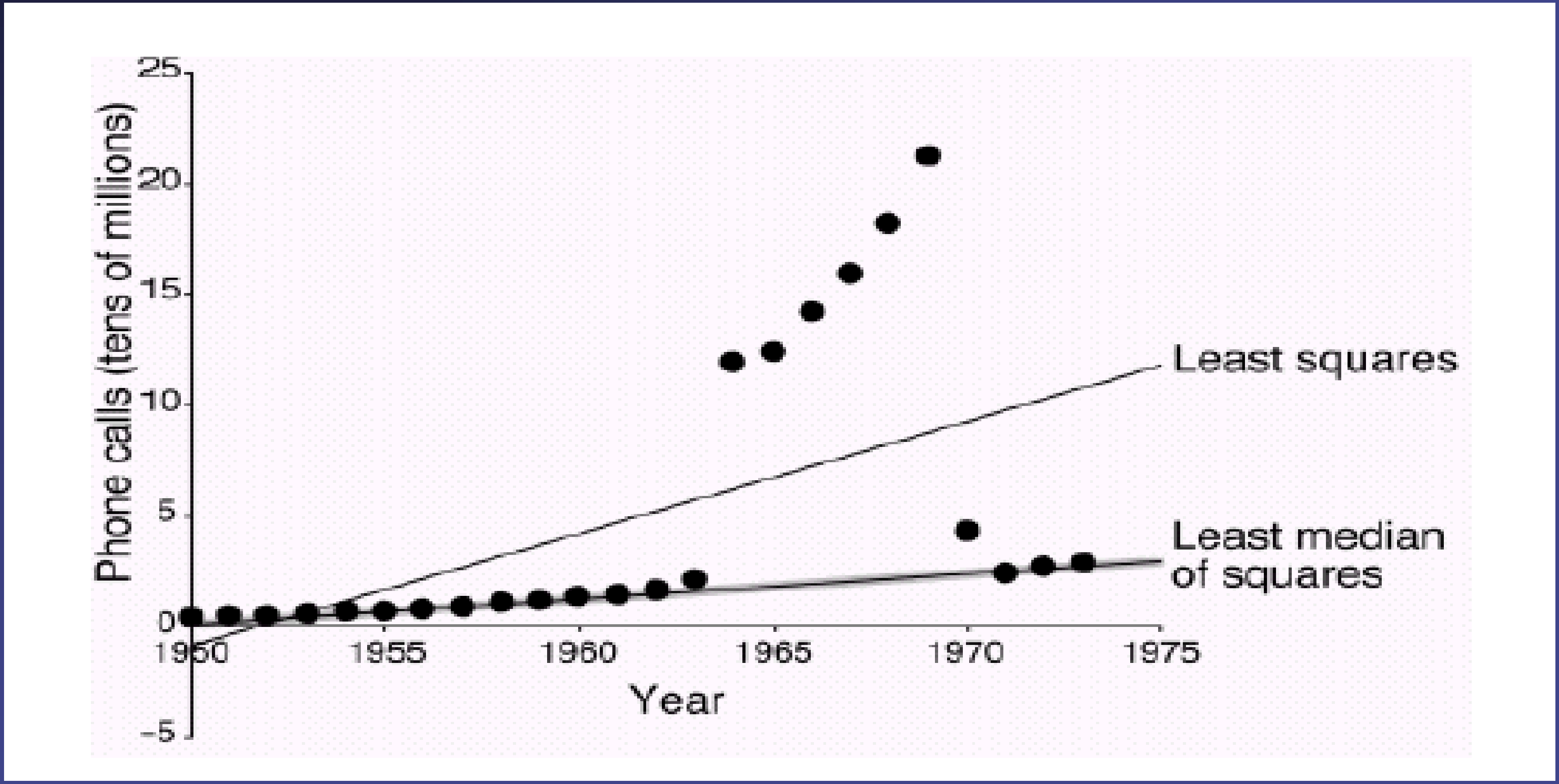
Automatic Data Cleansing

- Improving decision trees
  - relearn tree with misclassified instances removed
  - improvements only on size of the tree not accuracy
- Better strategy
  - let human expert check misclassified instances
- When systematic noise is present
  - it's better not to modify the data
- Attribute noise should be left in training set
- Unsystematic class noise in training set should be eliminated if possible

Robust regression

- Statistical methods that address problem of outliers are called robust
- Making regression more robust
  - Minimize absolute error instead of squared error
    - Weakness the effects of outliers
  - Remove outliers
    - 10% of points farthest from the regression plane
  - Minimize median instead of mean of squares
    - Very robust - copes with outliers in x and y direction
    - Finds narrowest strip covering half the observations

Example: least median of squares



Detecting anomalies

- Visualization best way of detecting anomalies
  - but often can’t be done
- Automatic approach: committee of different learning schemes
  - Decision tree, nearest-neighbor learner, and a linear discriminant function, etc.
  - Conservative approach
    - only delete instances which are incorrectly classified by all of them
  - Problem - might sacrifice instances of small classes

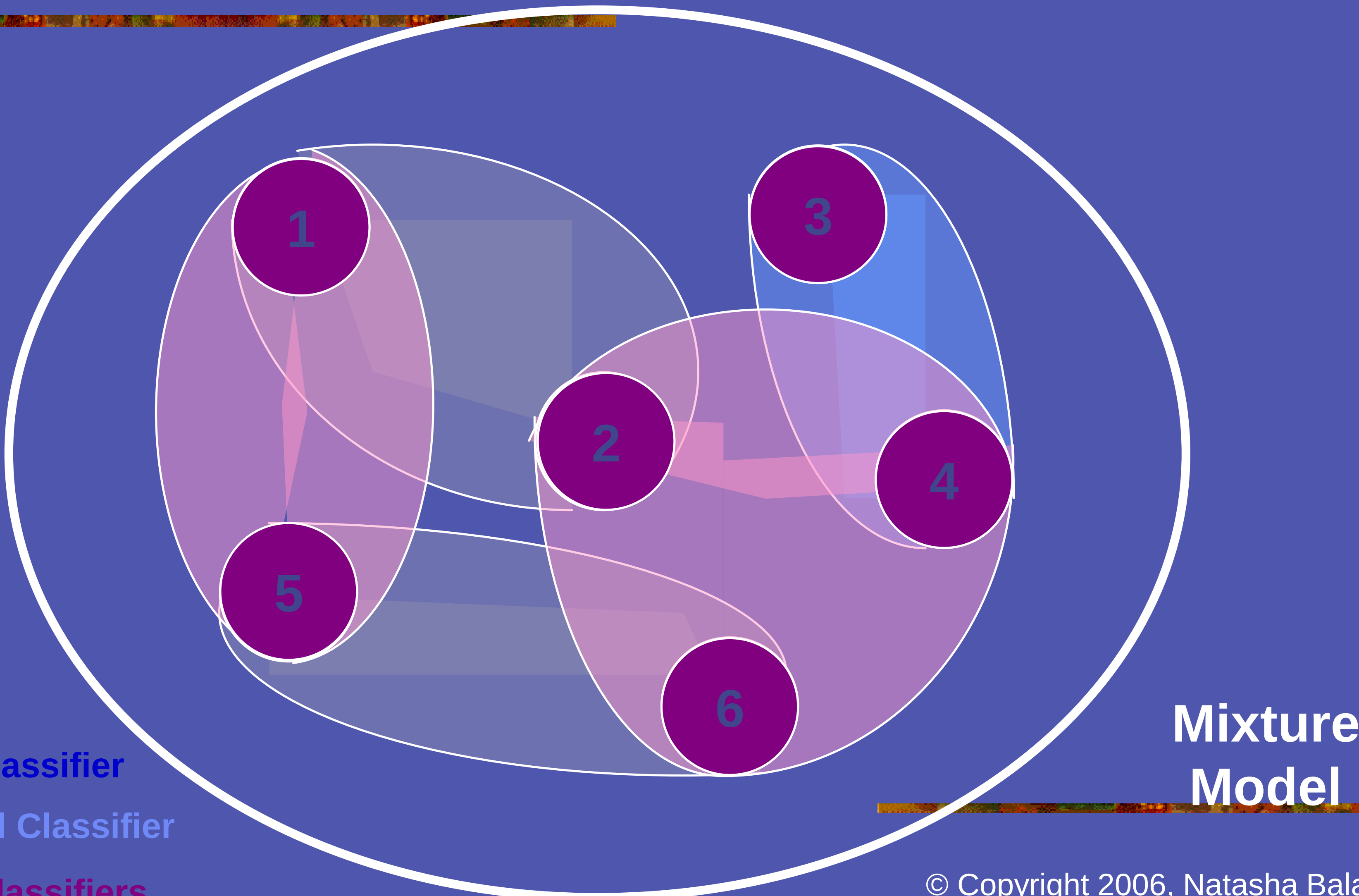
Combining multiple models

- Basic idea of “meta” learning schemes
  - build different “experts” and let them vote
- Advantage
  - often improves predictive performance
- Disadvantage
  - produces output that is very hard to analyze
- Schemes
  - Bagging, boosting, stacking
  - Can be applied to both classification and numeric prediction problems

Combining Classifiers

- Given
    - Training data set *D* for supervised learning
    - Collection of inductive learning algorithms
  - Return: new prediction algorithm that combines outputs from collection of prediction algorithms
- Desired Properties
  - Guarantees of performance of combined prediction
    - ability to improve weak classifiers

Improving Weak Classifiers



First Classifier

Second Classifier

Both Classifiers

Mixtures of Experts

- What Is A Weak Classifier?
  - One not guaranteed to do better than random guessing
  - Goal
    - combine multiple weak classifiers
    - get one at least as accurate as strongest
- Mixtures of Experts
  - “experts” express hypotheses
    - drawn from a hypothesis space

Bagging

- Employs simplest way of combining predictions:
  - voting/averaging
- Each model receives equal weight
- "Idealized" version of bagging
  - Sample several training sets of size n
    - instead of just having one training set of size n
  - Build a classifier for each training set
  - Combine the classifier's predictions
- This improves performance in almost all cases if learning scheme is unstable (decision trees)

Bootstrap Aggregating or Bagging

- "Two heads are better than one"
- Produce multiple classifiers from one data set
- Application of bootstrap sampling
  - Given: set *D* containing *m* training examples
  - Create *S*[*i*] by drawing *m* examples at random *with replacement* from *D*
  - *S*[*i*] of size *m*: expected to leave out 0.37 of examples from *D*
- Bagging
  - Create *k* bootstrap samples *S*[1], *S*[2], …, *S*[*k*]
  - Train distinct model on each *S*[*i*] to produce *k* classifiers
  - Classify new instance by classifier vote (equal weights)

Bias-variance decomposition

- Theoretical tool for analyzing how much specific training set affects performance of classifier
- Assume we have an infinite number of classifiers built from different training sets of size n
  - Bias of a learning scheme is the expected error of combined classifier on new data
  - Variance of a learning scheme is the expected error due to the particular training set used
  - Total expected error = bias + variance

More on bagging

- Bagging reduces variance by voting/averaging
  - reducing the overall expected error
  - In the case of classification there are pathological situations where the overall error might increase
  - Usually the more classifiers the better
- Problem
  - What if we only have one dataset?
- Solution
  - generate new datasets of size n by sampling with replacement from original dataset
- Can be very helpful if data is noisy

Bagging classifiers

model generation

```
Let n be the number of instances in the training data.
For each of t iterations:
  Sample n instances with replacement from training set.
  Apply the learning algorithm to the sample.
  Store the resulting model.
```

classification

```
For each of the t models:
  Predict class of instance using model.
Return class that has been predicted most often.
```

Bagging Results

- Experiments
  - [Breiman, 1996]: Given sample S of labeled data, do 100 times and report average
    - Divide S randomly into test set D_{test} (10%) and training set D_{train} (90%)
    - Learn decision tree from D_{train}

      $e_S \leftarrow$ **error of tree on** T
    - Do 50 times: create bootstrap $S[i]$, learn decision tree, prune using D

      $e_B \leftarrow$ **error of majority vote using trees to classify** T
  - [Quinlan, 1996]: Results using UCI Machine Learning Database Repository

When Should This Help?

- When learner is unstable
  - Small change to training set causes large change in output hypothesis
  - True for decision trees, neural networks; not true for *k*-nearest neighbor
- Experimentally, bagging can help substantially for unstable learners
  - Can possibly somewhat degrade results for stable learners

Boosting

- Also uses voting/averaging but models are weighted according to their performance
- Iterative procedure: new models are influenced by performance of previously built ones
  - New model is encouraged to become expert for instances classified incorrectly by earlier models
  - Intuitive justification - models should be experts that complement each other
- There are several variants of this algorithm

AdaBoost.M1

model generation

```
Assign equal weight to each training instance.
For each of t iterations:
  Apply learning algorithm to weighted dataset and store
     resulting model.
  Compute error e of model on weighted dataset and store error.
  If e equal to zero, or e greater or equal to 0.5:
    Terminate model generation.
  For each instance in dataset:
    If instance classified correctly by model:
       Multiply weight of instance by e / (1 - e).
  Normalize weight of all instances.
```

classification

```
Assign weight of zero to all classes.
For each of the t (or less) models:
  Add -log(e / (1 - e)) to weight of class predicted by model.
Return class with highest weight.
```

Boosting

- Can be applied without weights using resampling with probability determined by weights
  - Disadvantage: not all instances are used
  - Advantage: resampling can be repeated if error exceeds 0.5
- Emerged from computational learning theory
- Theoretical result
  - training error decreases exponentially
- Works if base classifiers not too complex and their error doesn’t become too large too quickly

Boosting

- Boosting often produces classifiers that are significantly more accurate on fresh data than bagging
  - But sometimes fails in practical situations
    - It can generate a classifier that is significantly less accurate than a single classifier build from the same data
      - Combined classifier overfits the data
- Boosting works with weak learners
  - only condition is that error doesn't exceed 0.5
- LogitBoost: more sophisticated boosting scheme

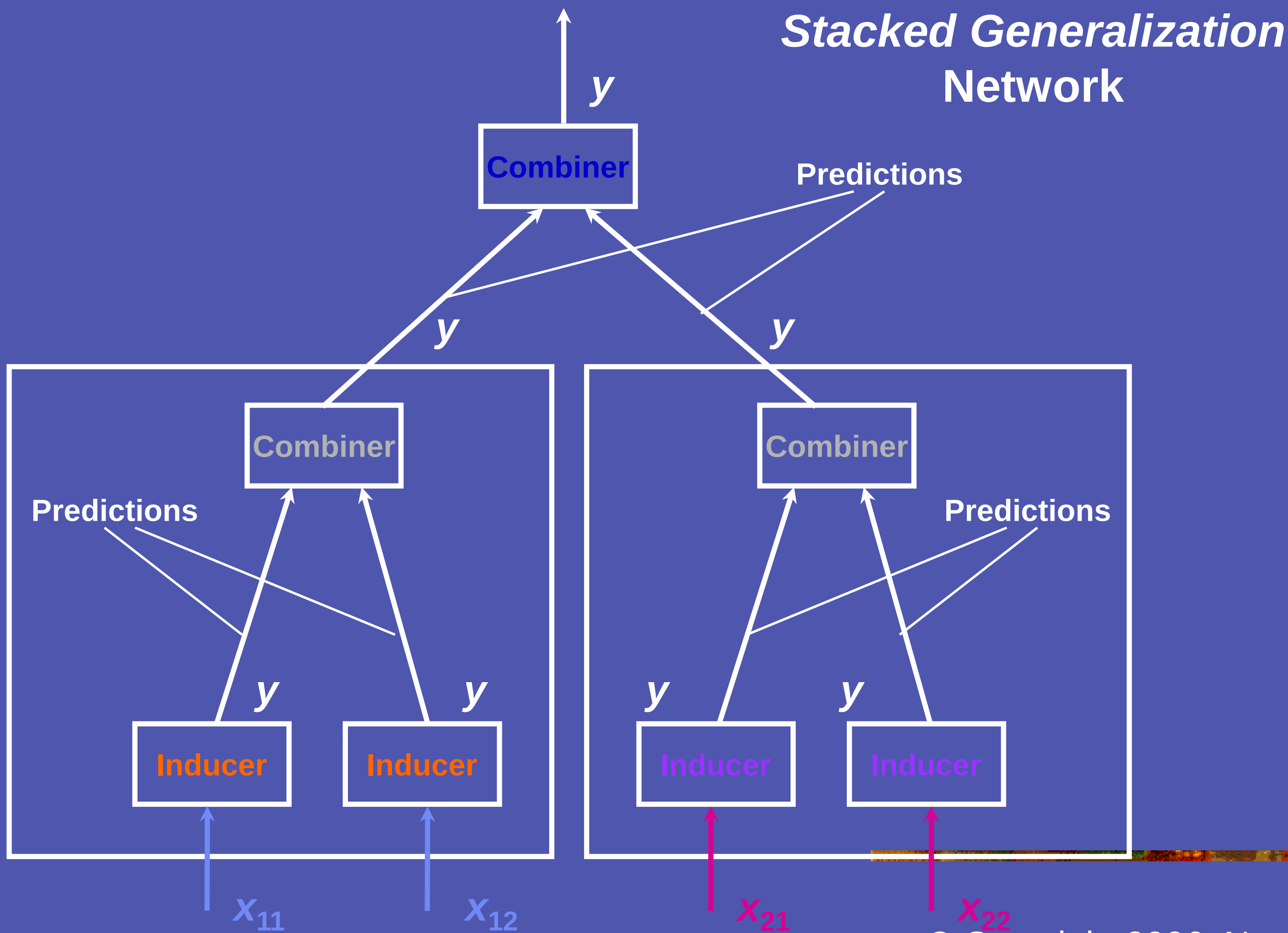
Stacked Generalization

- Stacked Generalization - <u>Stacking</u>
- Intuitive Idea
  - Train multiple learners
    - Each uses subsample of *D*
    - May be ANN, decision tree, etc.
  - Train ‘combiner’ on validation segment

Stacked Generalization
Stacking

- Used to combine models of different types
  - DT, Naïve Bayes and k-nearest neighbor
- With several algorithms available, instead of performing cross validation and selecting the best one
  - it's better to combine them
- Use un-weighted voting (as in bagging)?
  - makes sense only if the learning algorithms perform comparably well
  - it is not clear which classifier to trust
- Stacking uses a meta learner instead of voting – the goal of the meta
  - learner is to learn which classifiers are the reliable ones

Stacking



Stacked Generalization
Network
y
Combiner
Predictions
y
y
Combiner
Combiner
Predictions
Predictions
y
y
y
y
Inducer
Inducer
Inducer
Inducer
x_{11}
x_{12}
x_{21}
x_{22}

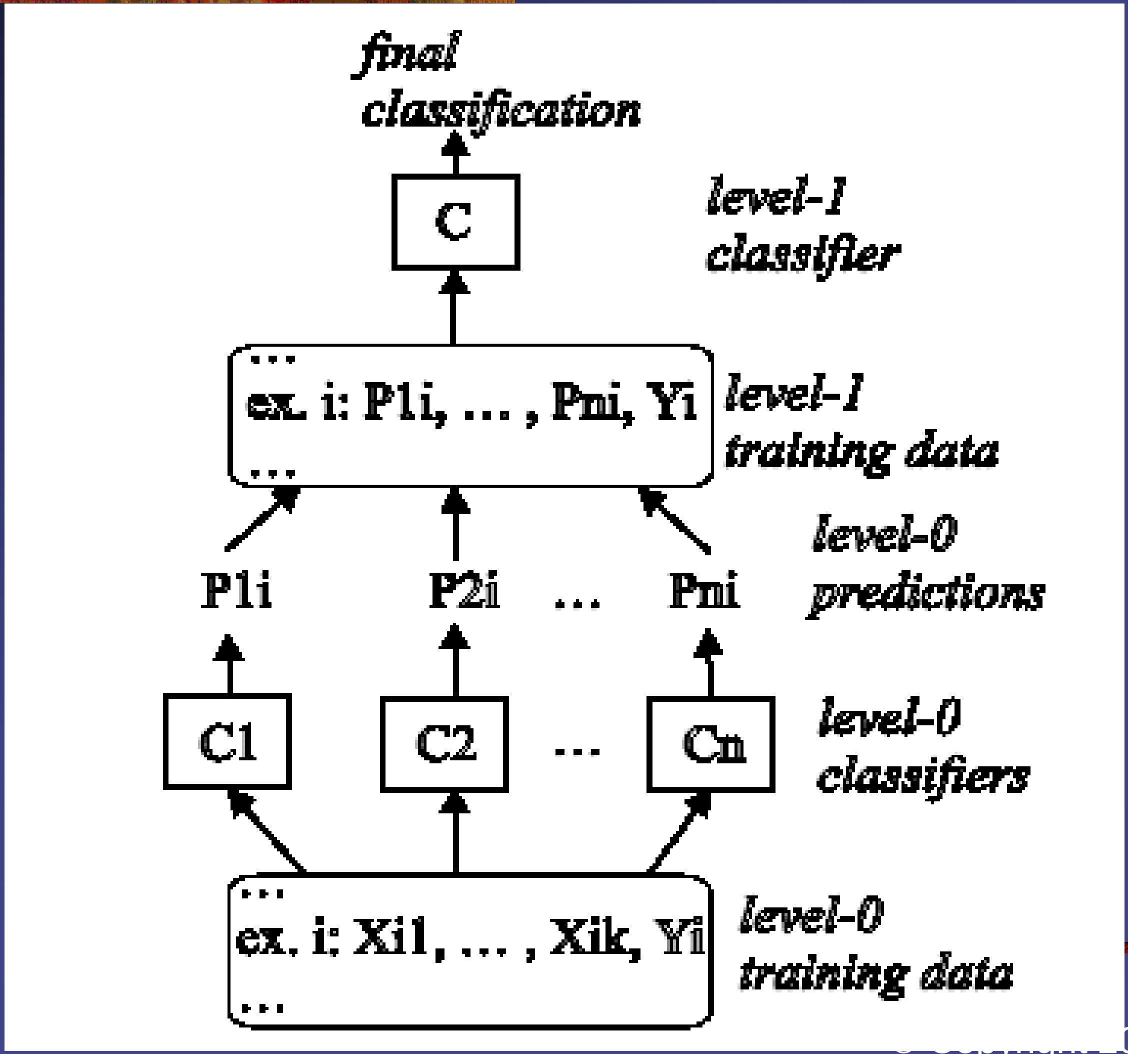
Stacking

- Hard to analyze theoretically - “black magic”
- Uses meta learner instead of voting to combine predictions of base learners
  - Predictions of base learners (level-0 models) are used as input for meta learner (level-1 model)
- Base learners usually different learning schemes
- Predictions on training data can’t be used to generate data for level-1 model!
  - Cross-validation-like scheme is employed

Combining Level-0 and Level-1 Models

- Meta learner is called *level–1 model*
- Base classifiers - *level-0 models*
- Outputs of the level-0 models are inputs to the level-1 model
- How to train the level-1 model?
- Idea 1: let each level-0 model classify a training instance, and attach to their predictions the actual classification as additional attribute, to form a training instance for the level-1 model
- Doesn’t work well in practice
  - prefers classifiers overfitting the training data

Stacking



final classification
C
level-1 classifier
...
ex. i: P1i, ... , Pni, Yi
...
level-1 training data
level-0 predictions
P1i
P2i
...
Pni
C1
C2
...
Cn
level-0 classifiers
...
ex. i: Xi1, ... , Xik, Yi
...
level-0 training data

Combining Level-0 and Level-1 Models

- Idea 2: separate the training data into training and validation set
- use the training data to train the level-0 classifiers
- apply the validation set to these classifiers and use the predictions to build training data for the level-1 model e
  - Even better: instead of training and validation set
    - use cross validation - slow but allows the level-0 models to make full use of the data
  - Most of the learning schemes output probabilities for each class label – use them instead of the single categorical prediction
- To classify a new example, the level-0 models are used to produce a vector which is an input to the level-1 model producing the final classification

Stacking

- If base learners can output probabilities it's better to use those as input to meta learner
- Which algorithm to use to generate meta learner?
  - In principle, any learning scheme can be applied
  - David Wolpert: "relatively global, smooth" model
    - Base learners do most of the work
    - Reduces risk of overfitting
- Stacking can also be applied to numeric prediction

Summary

- Combining Classifiers
  - Problem definition and motivation: improving accuracy in concept learning
  - General framework: collection of weak classifiers to be improved
- Weighted Majority (WM)
  - Weighting system for collection of algorithms
    - Weights each algorithm *in proportion to its training set accuracy*
    - Use this weight on test set predictions

Summary

- Bootstrap Aggregating (Bagging)
  - Voting system for collection of algorithms
  - Training set for each member: sampled with replacement
  - Works for unstable inducers
- Stacked Generalization (*aka* Stacking)
  - Hierarchical system for combining inducers (ANNs or other inducers)
  - Training sets for “leaves”: sampled with replacement; combiner: validation set

Summary

- Bagging and boosting use the same method of aggregating different models together - voting
- Bagging, boosting and stacking can be applied to both classification and numeric prediction
- Bagging and boosting combine models of the same type, stacking – of different types

Error-correcting output codes

- Very elegant method of transforming multiclass problem into two-class problem
- Simple scheme: as many binary class attributes as original classes using one-per-class coding

class vector	class
a	1000
b	0100
c	0010
d	0001

- Idea: use error-correcting codes instead

ECOCs

- Example:

class	class vector
a	1111111
b	0000111
c	0011001
d	0101010

  - What's the true class if base classifiers predict 1011111?
- We want code words for which minimum hamming distance between any pair of words d is large
  - Up to (d-1)/2 single-bit errors can be corrected

ECOCs

- Two criteria for error-correcting output codes:
  - Row-separation: minimum distance between rows
  - Column-separation: minimum distance between columns (and columns’ complements)
    - Why? Because if columns are identical, base classifiers will make the same errors
    - Error-correction is weakened if errors are correlated
- Only works for problems with more than 3 classes: for 3 classes there are only 23 possible columns

Exhaustive ECOCs

- With few classes exhaustive codes can be build (like the one on an earlier slide)
- Exhaustive code for k classes:
  - The columns comprise every possible k-string
  - Except for complements and all-zero/one strings
  - Each code word contains 2k-1-1 bits
- Code word for 1st class: all ones
- 2nd class: 2k-2 zeroes followed by 2k-2-1 ones
- ith class: alternating runs of 2k-i zeroes and ones, the last run being one short

ECOCs

- With more classes, exhaustive codes are infeasible
  - Number of columns increases exponentially
- Random code words have good error-correcting properties on average!
- More sophisticated methods exist for generating ECOCs using a small number of columns
- ECOCs don’t work with NN classifier
  - But: works if different attribute subsets are used to predict each output bit

Weighted Majority

- Weight-Based Combiner
  - Collect votes from pool of prediction algorithms for each training example
  - Decrease weight associated with each algorithm that guessed wrong
  - Combiner predicts weighted majority label
- Performance Goals
  - Improving training set accuracy
    - Want to combine weak classifiers
    - Want to bound number of mistakes in terms of minimum made by any one algorithm
  - Hope that this results in good generalization quality