CSE4334/5334 Data Mining

10 Classification: Evaluation

Chengkai Li

Department of Computer Science and Engineering
University of Texas at Arlington
Fall 2018 (Slides courtesy of Pang-Ning Tan, Michael Steinbach and Vipin Kumar)

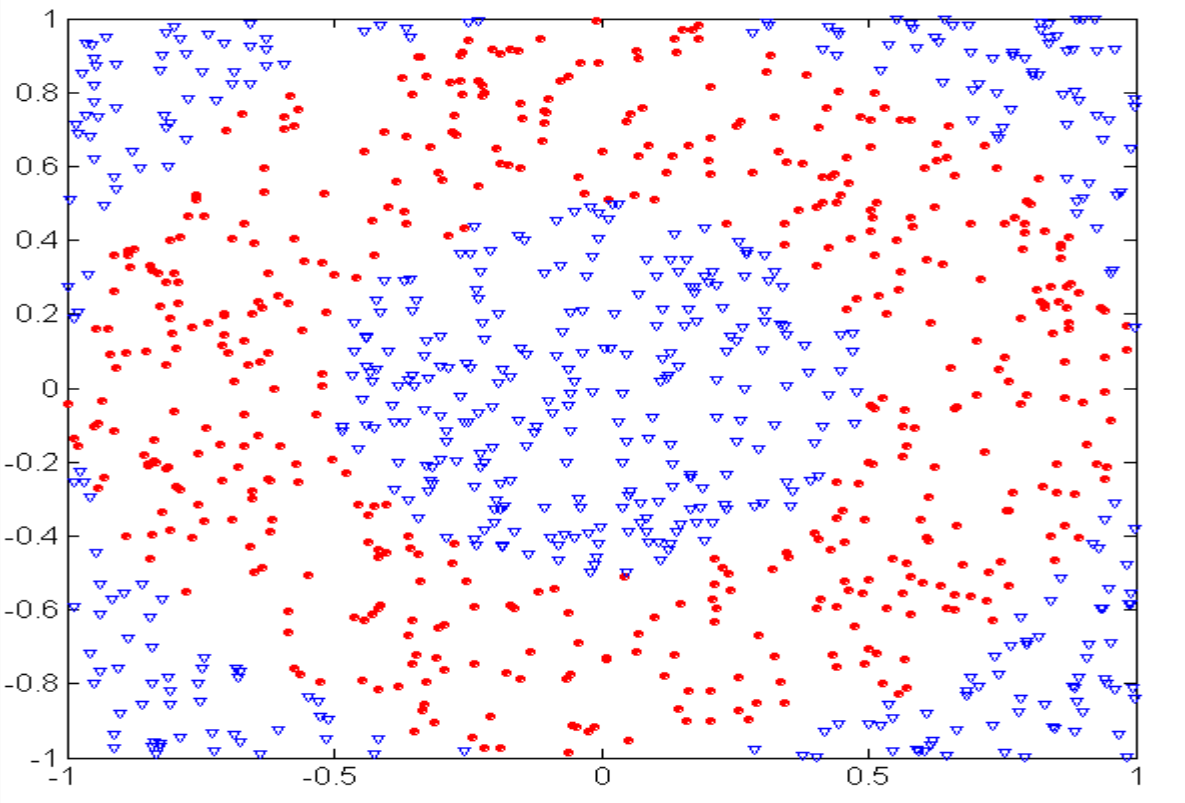
Practical Issues of Classification

Underfitting and Overfitting

Missing Values

Costs of Classification

Underfitting and Overfitting



500 circular and 500 triangular data points.

Circular points:

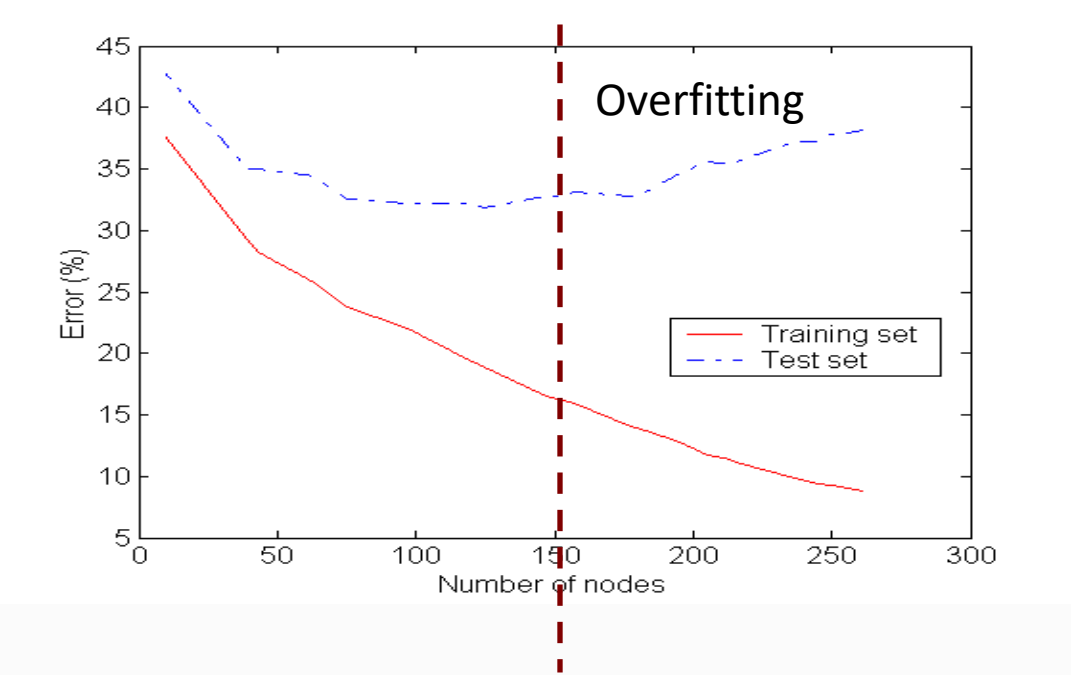
$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

Triangular points:

$\text{sqrt}(x_1^2+x_2^2) < 0.5$ or

$\text{sqrt}(x_1^2+x_2^2) > 1$

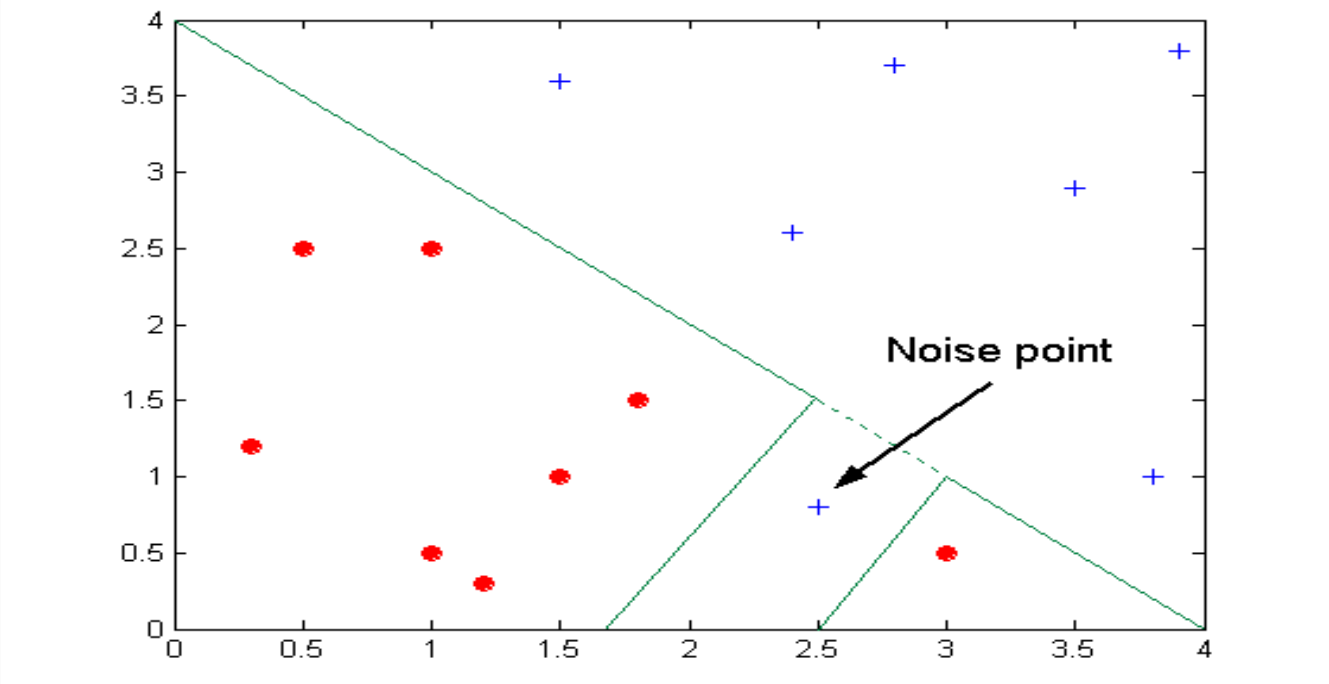
Underfitting and Overfitting



Underfitting: when model is too simple, both training and test errors are large

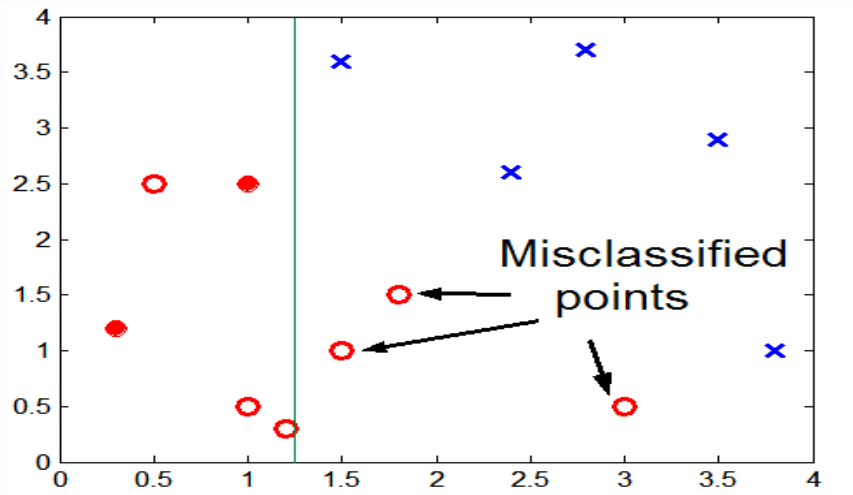
Overfitting: when model is too complex, test error increases even though training error decreases

Overfitting due to Noise



Decision boundary is distorted by noise point

Overfitting due to Insufficient Examples



Lack of data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region

- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task

Notes on Overfitting

Overfitting results in decision trees that are more complex than necessary

Training error no longer provides a good estimate of how well the tree will perform on previously unseen records

Need new ways for estimating errors

Estimating Generalization Errors

Re-substitution errors: error on training (Σ e(t))

Generalization errors: error on testing (Σ e'(t))

Methods for estimating generalization errors:

- Optimistic approach: e'(t) = e(t)
- Pessimistic approach:
  - For each leaf node: e'(t) = (e(t)+0.5)
  - Total errors: e'(T) = e(T) + N × 0.5 (N: number of leaf nodes)
  - For a tree with 30 leaf nodes and 10 errors on training (out of 1000 instances):

    Training error = 10/1000 = 1%

    Generalization error = (10 + 30×0.5)/1000 = 2.5%
- Reduced error pruning (REP):
  - uses validation data set to estimate generalization error

Occam's Razor

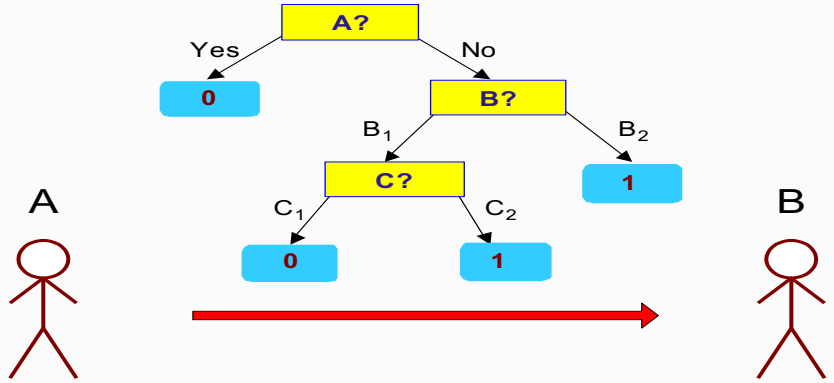
Given two models of similar generalization errors, one should prefer the simpler model over the more complex model

For complex models, there is a greater chance that it was fitted accidentally by errors in data

Therefore, one should include model complexity when evaluating a model

Minimum Description Length (MDL)

X	y
X_1	1
X_2	0
X_3	0
X_4	1
…	…
X_n	1



X	y
X_1	?
X_2	?
X_3	?
X_4	?
…	…
X_n	?

Cost(Model,Data) = Cost(Data|Model) + Cost(Model)

- Cost is the number of bits needed for encoding.
- Search for the least costly model.

Cost(Data|Model) encodes the misclassification errors.

Cost(Model) uses node encoding (number of children) plus splitting condition encoding.

How to Address Overfitting

Pre-Pruning (Early Stopping Rule)

- Stop the algorithm before it becomes a fully-grown tree
- Typical stopping conditions for a node:
  - Stop if all instances belong to the same class
  - Stop if all the attribute values are the same
- More restrictive conditions:
  - Stop if number of instances is less than some user-specified threshold
  - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
  - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

How to Address Overfitting…

Post-pruning

- Grow decision tree to its entirety
- Trim the nodes of the decision tree in a bottom-up fashion
- If generalization error improves after trimming, replace sub-tree by a leaf node.
- Class label of leaf node is determined from majority class of instances in the sub-tree
- Can use MDL for post-pruning

Example of Post-Pruning

Class = Yes	20
Class = No	10
Error = 10/30	

Training Error (Before splitting) = 10/30

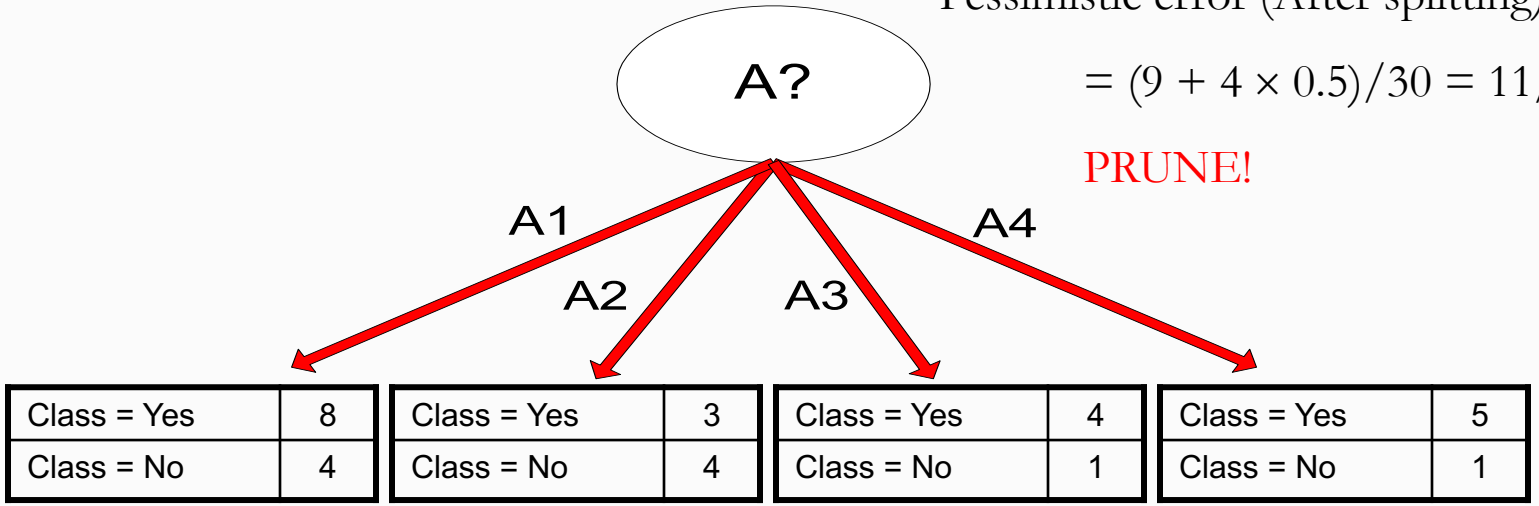
Pessimistic error = (10 + 0.5)/30 = 10.5/30

Training Error (After splitting) = 9/30

Pessimistic error (After splitting)

= (9 + 4 × 0.5)/30 = 11/30

PRUNE!



Class = Yes	8
Class = No	4

Class = Yes	3
Class = No	4

Class = Yes	4
Class = No	1

Class = Yes	5
Class = No	1

Examples of Post-pruning

- Optimistic error?

  Don't prune for both cases

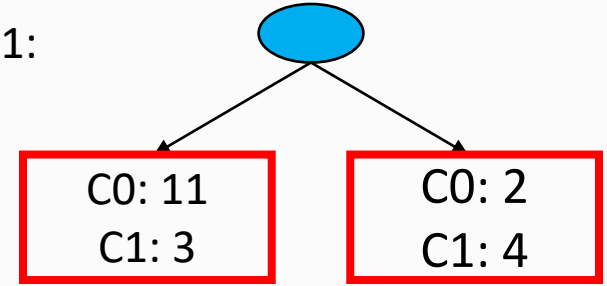
- Pessimistic error?

  Don't prune case 1, prune case 2

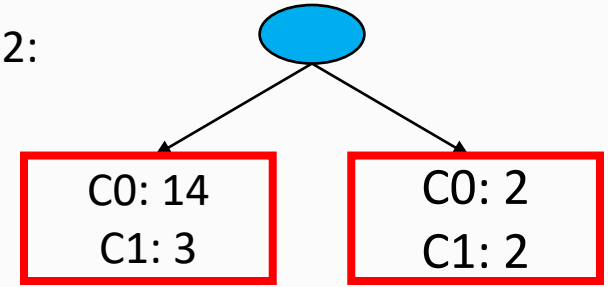
- Reduced error pruning?

  Depends on validation set

Case 1:



Case 2:



Handling Missing Attribute Values

Missing values affect decision tree construction in three different ways:

- Affects how impurity measures are computed
- Affects how to distribute instance with missing value to child nodes
- Affects how a test instance with missing value is classified

Computing Impurity Measure

Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	?	Single	90K	Yes

Missing value

Before Splitting:

Entropy(Parent)

$= -0.3 \log(0.3) - (0.7)\log(0.7) = 0.8813$

	Class = Yes	Class = No
Refund=Yes	0	3
Refund=No	2	4
Refund=?	1	0

Split on Refund:

Entropy(Refund=Yes) = 0

Entropy(Refund=No)

$= -(2/6)\log(2/6) - (4/6)\log(4/6) = 0.9183$

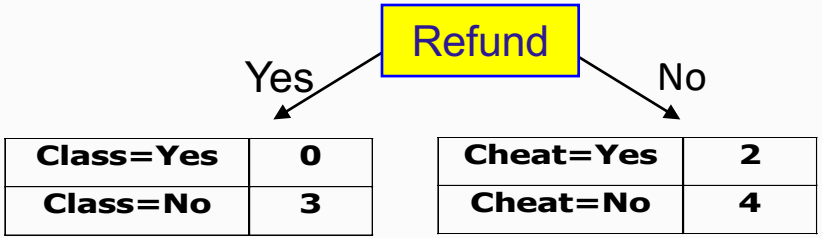
Entropy(Children)

$= 0.3\ (0) + 0.6\ (0.9183) = 0.551$

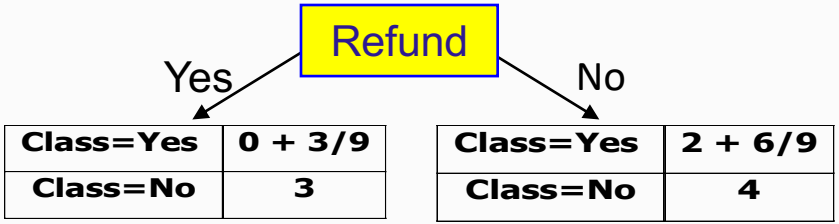
Gain $= 0.9 \times (0.8813 - 0.551) = 0.3303$

Distribute Instances

Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No



Tid	Refund	Marital Status	Taxable Income	Class
10	?	Single	90K	Yes



Probability that Refund=Yes is 3/9

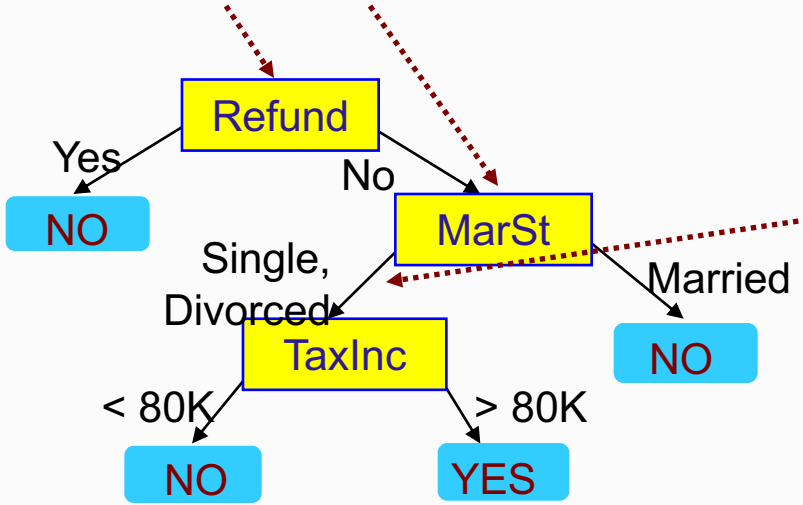
Probability that Refund=No is 6/9

Assign record to the left child with weight = 3/9 and to the right child with weight = 6/9

Classify Instances

New record:

Tid	Refund	Marital Status	Taxable Income	Class
11	No	?	85K	?



	Married	Single	Divorced	Total
Class=No	3	1	0	4
Class=Yes	0	1+6/9	1	2.67
Total	3	2.67	1	6.67

Probability that Marital Status = Married is 3/6.67

Probability that Marital Status ={Single,Divorced} is 3.67/6.67

Other Issues

Data Fragmentation

Search Strategy

Expressiveness

Tree Replication

Data Fragmentation

Number of instances gets smaller as you traverse down the tree

Number of instances at the leaf nodes could be too small to make any statistically significant decision

Search Strategy

Finding an optimal decision tree is NP-hard

The algorithm presented so far uses a greedy, top-down, recursive partitioning strategy to induce a reasonable solution

Other strategies?

- Bottom-up
- Bi-directional

Expressiveness

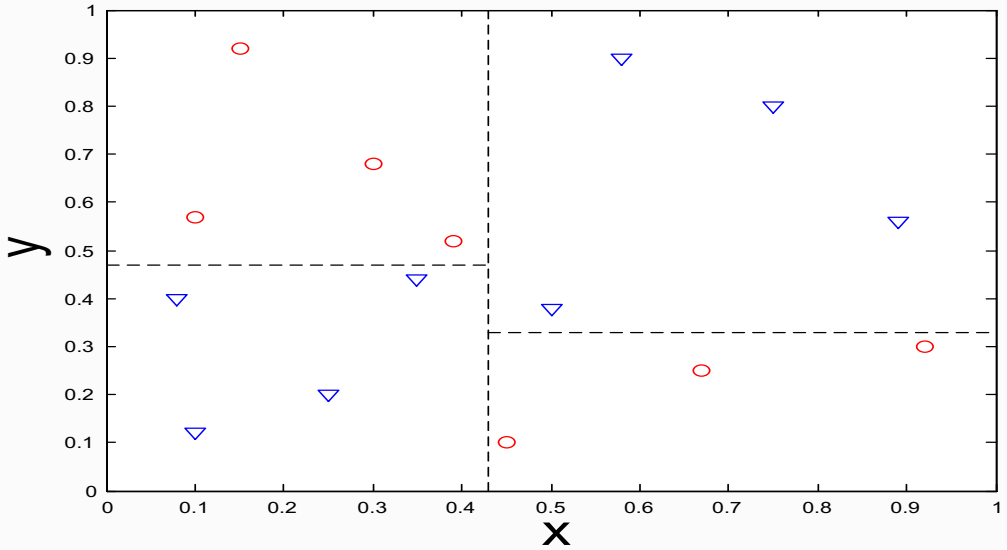
Decision tree provides expressive representation for learning discrete-valued function

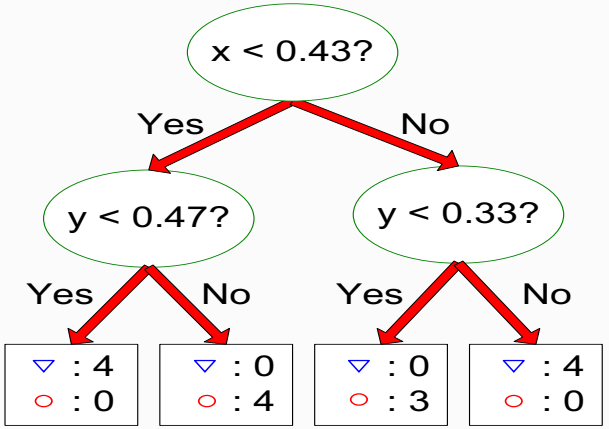
- But they do not generalize well to certain types of Boolean functions
  - Example: parity function:
    - Class = 1 if there is an even number of Boolean attributes with truth value = True
    - Class = 0 if there is an odd number of Boolean attributes with truth value = True
  - For accurate modeling, must have a complete tree

Not expressive enough for modeling continuous variables

- Particularly when test condition involves only a single attribute at-a-time

Decision Boundary

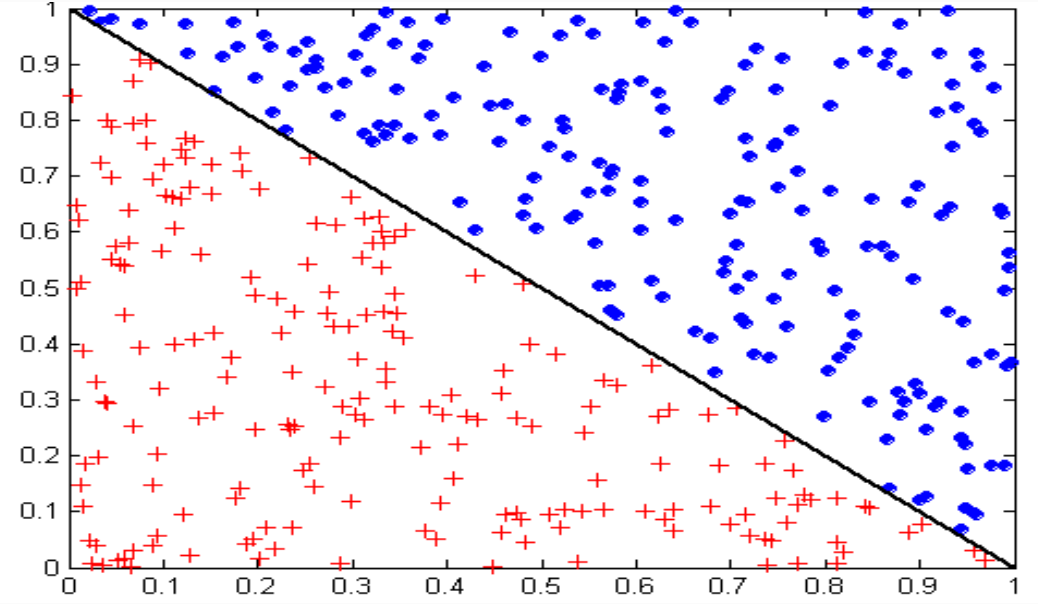


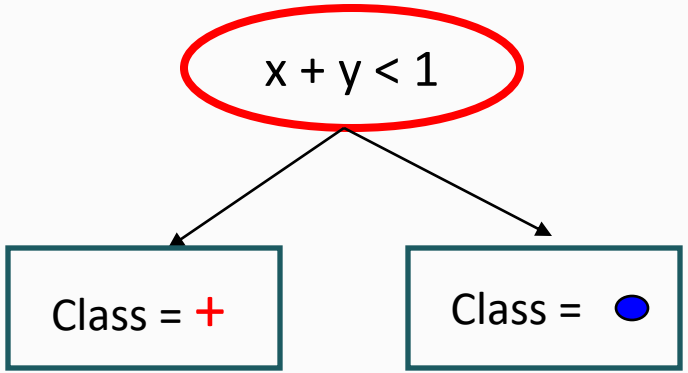


- Border line between two neighboring regions of different classes is known as decision boundary

- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time

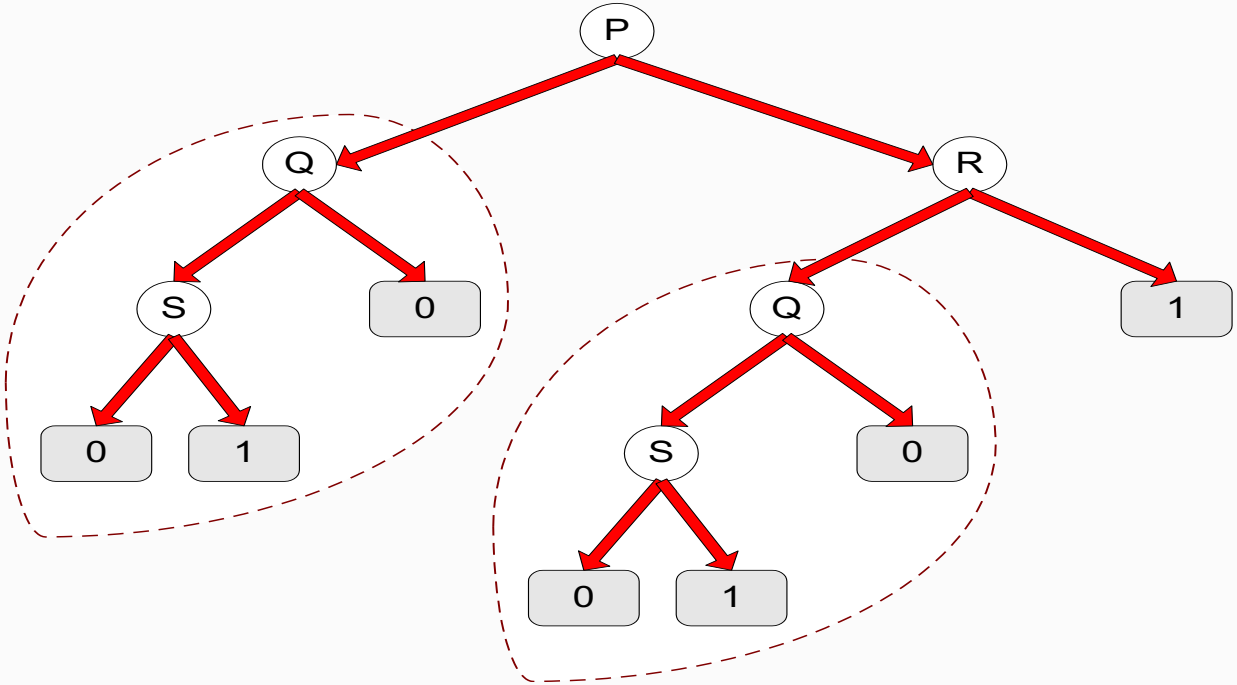
Oblique Decision Trees





- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive

Tree Replication



- Same subtree appears in multiple branches

Model Evaluation

Metrics for Performance Evaluation

- How to evaluate the performance of a model?

Methods for Performance Evaluation

- How to obtain reliable estimates?

Methods for Model Comparison

- How to compare the relative performance among competing models?

Model Evaluation

Metrics for Performance Evaluation

- How to evaluate the performance of a model?

Methods for Performance Evaluation

- How to obtain reliable estimates?

Methods for Model Comparison

- How to compare the relative performance among competing models?

Metrics for Performance Evaluation

Focus on the predictive capability of a model

- Rather than how fast it takes to classify or build models, scalability, etc.

Confusion Matrix:

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Metrics for Performance Evaluation…

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

Most widely-used metric:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

Limitation of Accuracy

Consider a 2-class problem

- Number of Class 0 examples = 9990
- Number of Class 1 examples = 10

If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %

- Accuracy is misleading because model does not detect any class 1 example

Cost Matrix

		PREDICTED CLASS	
ACTUAL CLASS	C(i|j)	**Class=Yes**	**Class=No**
	Class=Yes	C(Yes|Yes)	C(No|Yes)
	Class=No	C(Yes|No)	C(No|No)

C(i|j): Cost of misclassifying class j example as class i

Computing Cost of Classification

Cost Matrix	PREDICTED CLASS		
	C(i|j)	+	-
ACTUAL CLASS	+	-1	100
	-	1	0

Model M_1	PREDICTED CLASS		
		+	-
ACTUAL CLASS	+	150	40
	-	60	250

Model M_2	PREDICTED CLASS		
		+	-
ACTUAL CLASS	+	250	45
	-	5	200

Accuracy = 80%

Cost = 3910

Accuracy = 90%

Cost = 4255

Cost vs Accuracy

Count	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

Cost	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	p	q
	Class=No	q	p

Accuracy is proportional to cost if

1. C(Yes|No)=C(No|Yes) = q
2. C(Yes|Yes)=C(No|No) = p

$N = a + b + c + d$

$\text{Accuracy} = (a + d)/N$

$$\begin{aligned}\text{Cost} &= p\,(a + d) + q\,(b + c) \\ &= p\,(a + d) + q\,(N - a - d) \\ &= q\,N - (q - p)(a + d) \\ &= N\,[q - (q-p) \times \text{Accuracy}]\end{aligned}$$

Cost-Sensitive Measures

$$\text{Precision (p)} = \frac{a}{a+c}$$

$$\text{Recall (r)} = \frac{a}{a+b}$$

$$\text{F - measure (F)} = \frac{2}{\frac{1}{r}+\frac{1}{p}} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

F-measure (also called F-score, F1 score) is the harmonic mean of r and p.

- Precision is biased towards C(Yes|Yes) & C(Yes|No)
- Recall is biased towards C(Yes|Yes) & C(No|Yes)
- F-measure is biased towards all except C(No|No)

$$\text{Weighted Accuracy} = \frac{w_1 a + w_4 d}{w_1 a + w_2 b + w_3 c + w_4 d}$$

Model Evaluation

Metrics for Performance Evaluation

- How to evaluate the performance of a model?

Methods for Performance Evaluation

- How to obtain reliable estimates?

Methods for Model Comparison

- How to compare the relative performance among competing models?

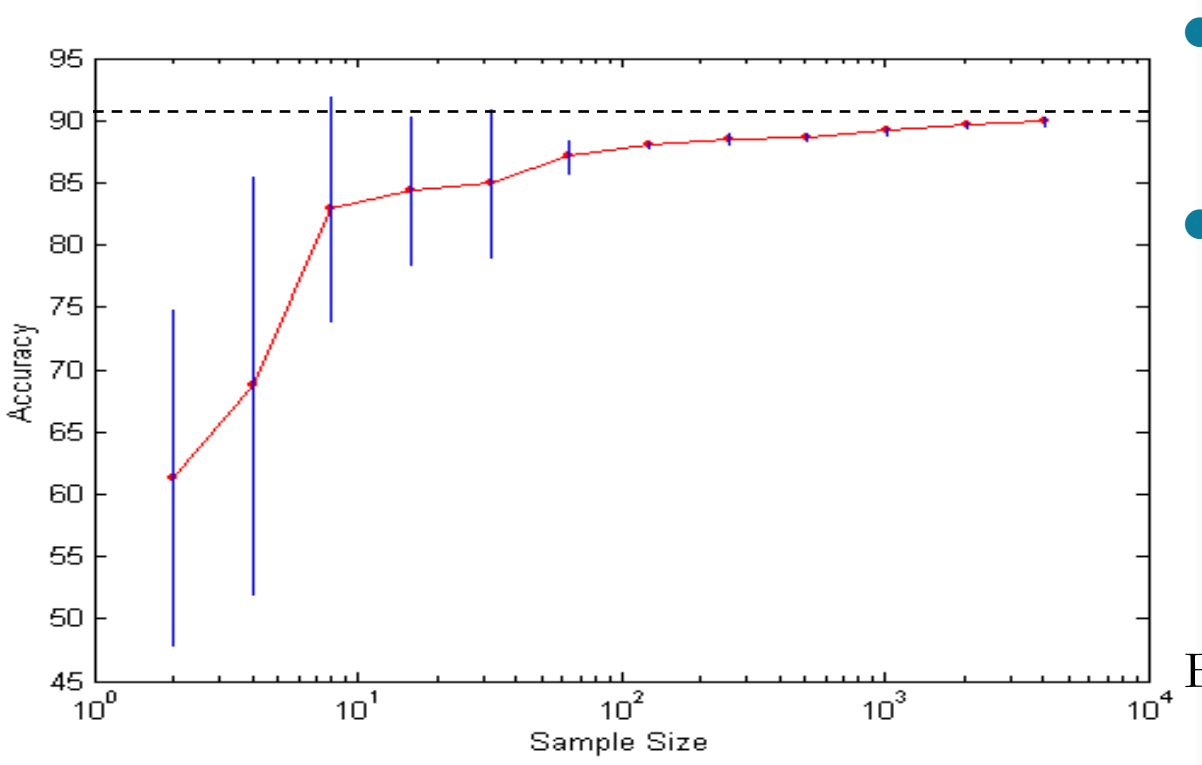
Methods for Performance Evaluation

How to obtain a reliable estimate of performance?

Performance of a model may depend on other factors besides the learning algorithm:

- Class distribution
- Cost of misclassification
- Size of training and test sets

Learning Curve



- Learning curve shows how accuracy changes with varying sample size
- Requires a sampling schedule for creating learning curve:
  - Arithmetic sampling (Langley, et al)
  - Geometric sampling (Provost et al)

Effect of small sample size:

- Bias in the estimate
- Variance of estimate

Methods of Estimation

Holdout

- Reserve 2/3 for training and 1/3 for testing

Random subsampling

- Repeated holdout

Cross validation

- Partition data into k disjoint subsets
- k-fold: train on k-1 partitions, test on the remaining one
- Leave-one-out: k=n

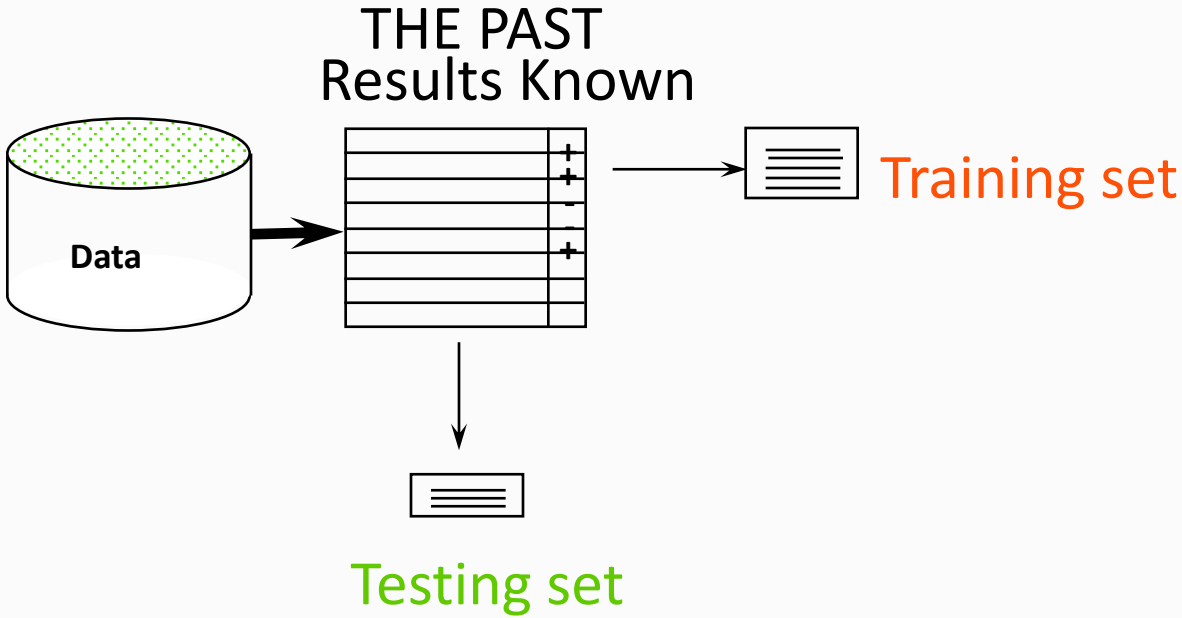
Stratified sampling

- oversampling vs undersampling

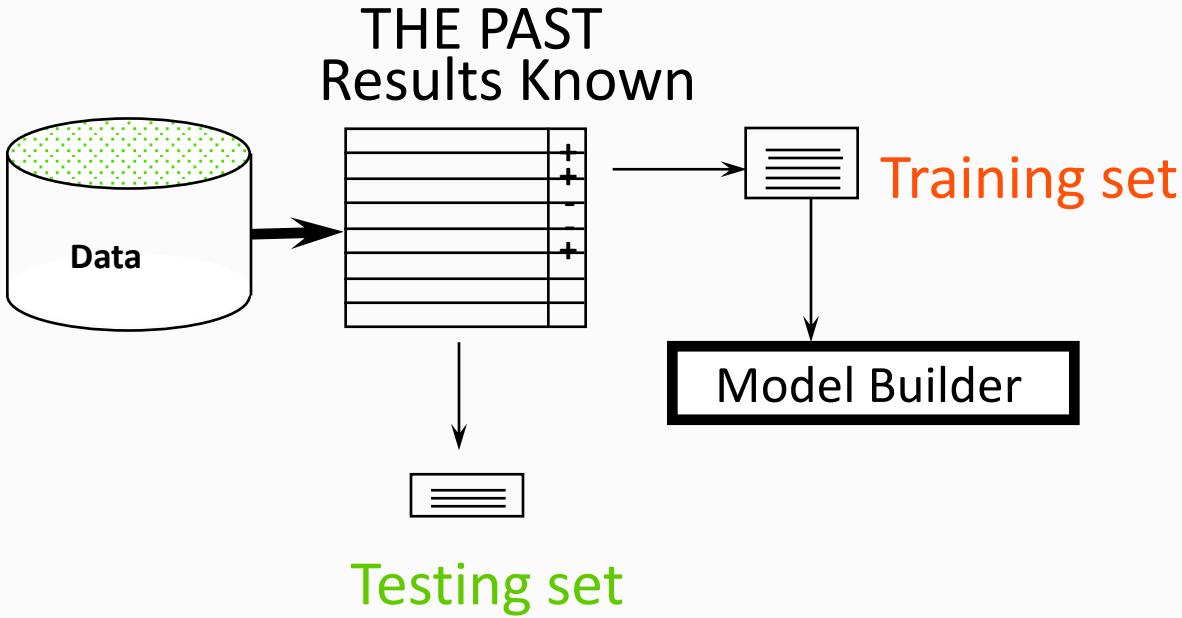
Bootstrap

- Sampling with replacement

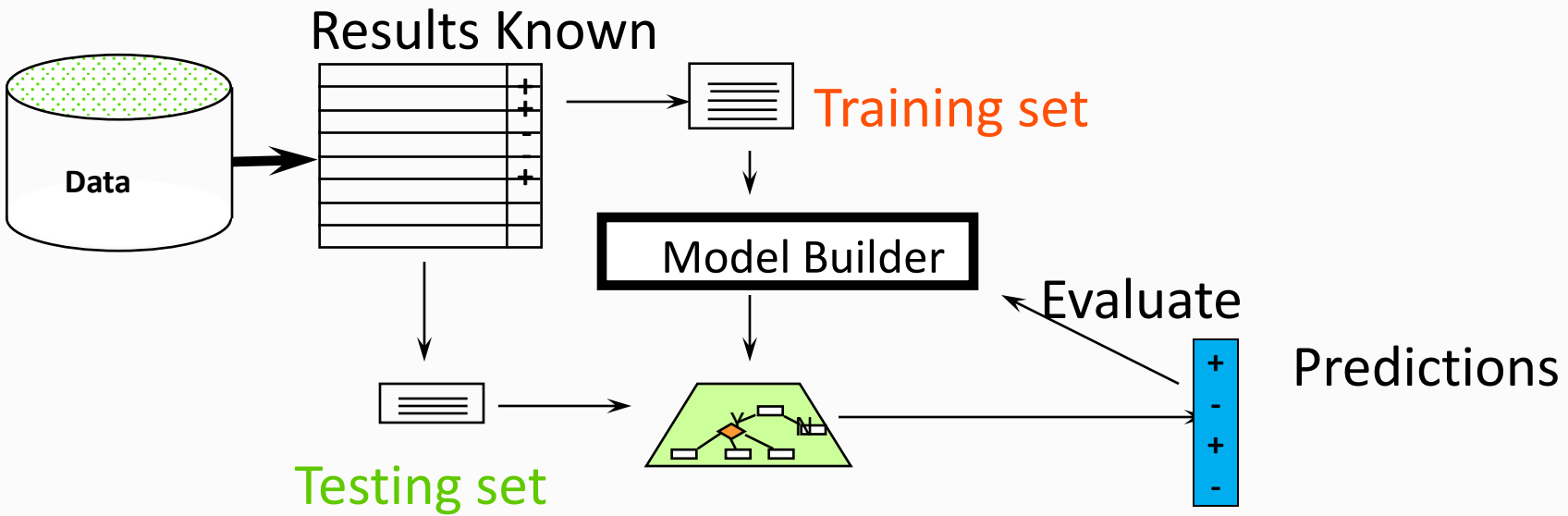
Classification Step 1: Split data into train and test sets



Classification Step 2: Build a model on a training set



Classification Step 3: Evaluate on test set



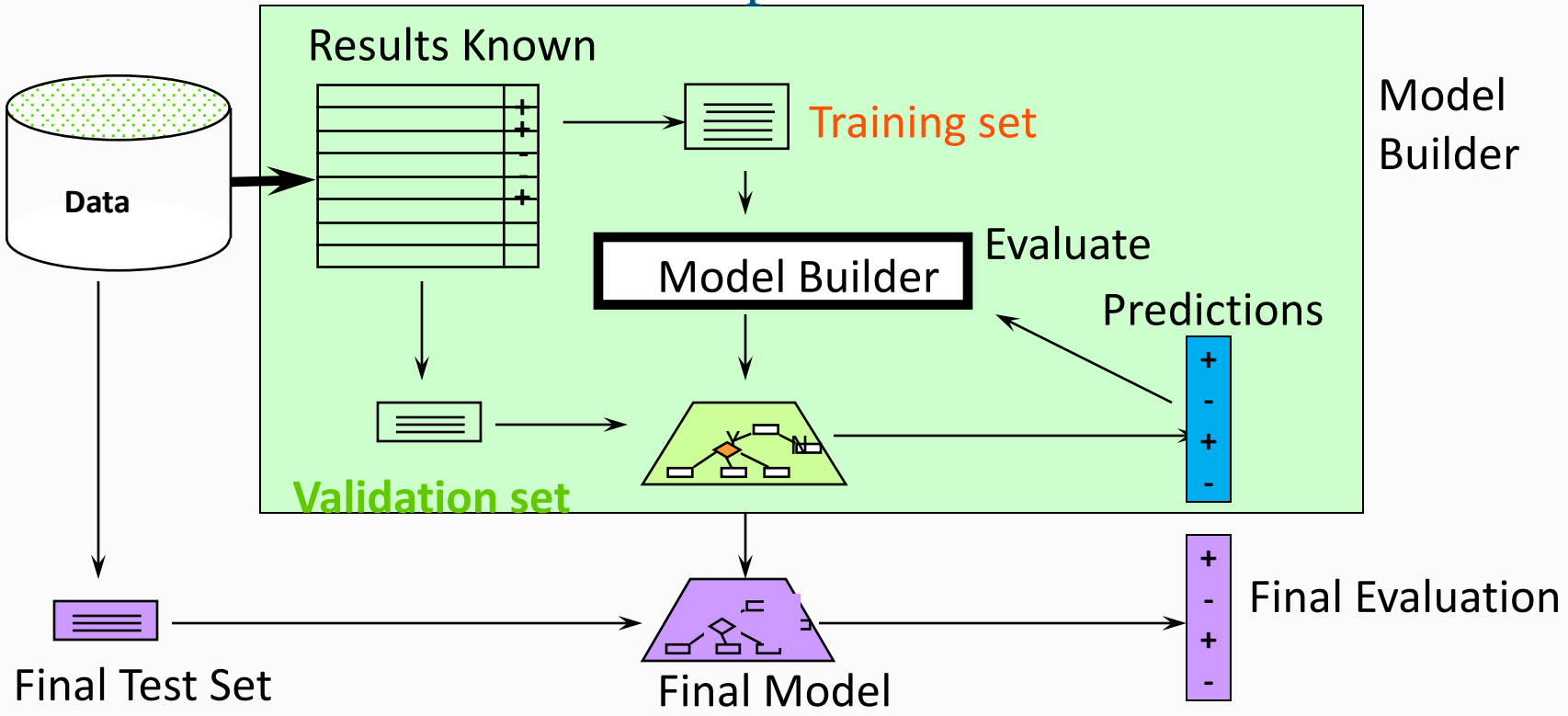
A note on parameter tuning

- It is important that the test data is not used *in any way* to create the classifier
- Some learning schemes operate in two stages:
  - Stage 1: builds the basic structure
  - Stage 2: optimizes parameter settings
- The test data can't be used for parameter tuning!
- Proper procedure uses three sets: **training data, validation data, and test data**
  - Validation data is used to optimize parameters

Making the most of the data

- Once evaluation is complete, *all the data* can be used to build the final classifier
- Generally, the larger the training data the better the classifier (but returns diminish)
- The larger the test data the more accurate the error estimate

Classification: Train, Validation, Test split



Evaluation on "small" data

- The *holdout* method reserves a certain amount for testing and uses the remainder for training
  - Usually: one third for testing, the rest for training
- For "unbalanced" datasets, samples might not be representative
  - Few or none instances of some classes
- *Stratified sample: advanced version of balancing the data*
  - Make sure that each class is represented with approximately equal proportions in both subsets

Evaluation on "small" data

What if we have a small data set?

- The chosen 2/3 for training may not be representative.
- The chosen 1/3 for testing may not be representative.

Repeated holdout method

repeated holdout method

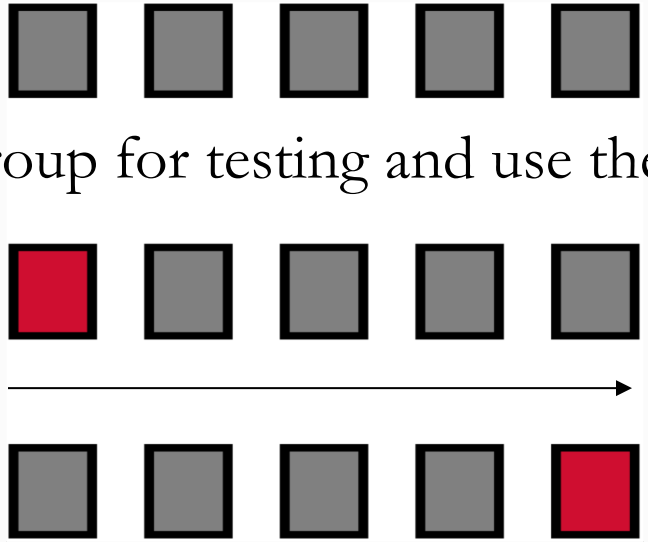
- Holdout estimate can be made more reliable by repeating the process with different subsamples
  - In each iteration, a certain proportion is randomly selected for training (possibly with stratification)
  - The error rates on the different iterations are averaged to yield an overall error rate
- Still not optimum: the different test sets overlap.
  - Can we prevent overlapping?

Cross-validation

- *Cross-validation* avoids overlapping test sets
  - First step: data is split into *k* subsets of equal size
  - Second step: each subset in turn is used for testing and the remainder for training
- This is called *k-fold cross-validation*
- Often the subsets are stratified before the cross-validation is performed
- The error estimates are averaged to yield an overall error estimate

Cross-validation example:

Break up data into groups of the same size



Hold aside one group for testing and use the rest to build model

Test

Repeat

More on cross-validation

- Standard method for evaluation: stratified ten-fold cross-validation
- Why ten? Extensive experiments have shown that this is the best choice to get an accurate estimate
- Stratification reduces the estimate's variance
- Even better: repeated stratified cross-validation
  - E.g. ten-fold cross-validation is repeated ten times and results are averaged (reduces the variance)

Leave-One-Out cross-validation

- Leave-One-Out:
  a particular form of cross-validation:
  - Set number of folds to number of training instances
  - I.e., for *n* training instances, build classifier *n* times
- Makes best use of the data
- Involves no random subsampling
- Very computationally expensive
  - (exception: NN)

Summary of Evaluation Methods

- Use Train, Test, Validation sets for "LARGE" data
- Balance "un-balanced" data
- Use Cross-validation for small data
- Don't use test data for parameter tuning - use separate validation data

- Most Important: Avoid Overfitting

Model Evaluation

Metrics for Performance Evaluation

- How to evaluate the performance of a model?

Methods for Performance Evaluation

- How to obtain reliable estimates?

Methods for Model Comparison

- How to compare the relative performance among competing models?

ROC (Receiver Operating Characteristic)

Developed in 1950s for signal detection theory to analyze noisy signals

- Characterize the trade-off between positive hits and false alarms

ROC curve plots TP (on the y-axis) against FP (on the x-axis)

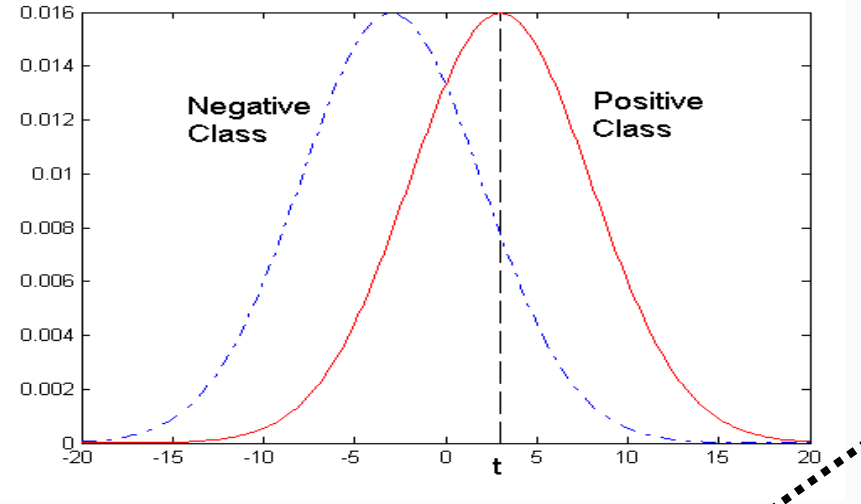
Performance of each classifier represented as a point on the ROC curve

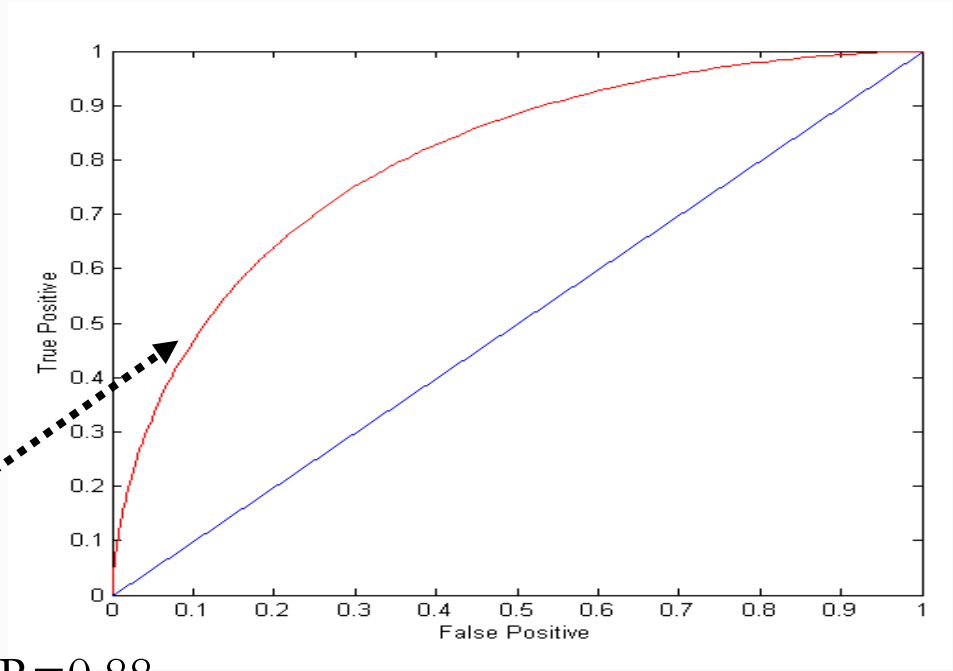
- changing the threshold of algorithm, sample distribution or cost matrix changes the location of the point

ROC Curve

- 1-dimensional data set containing 2 classes (positive and negative)

- any points located at x > t is classified as positive





At threshold t:

TPR=0.5, FNR=0.5, FPR=0.12, TNR=0.88

ROC Curve

(TP,FP):

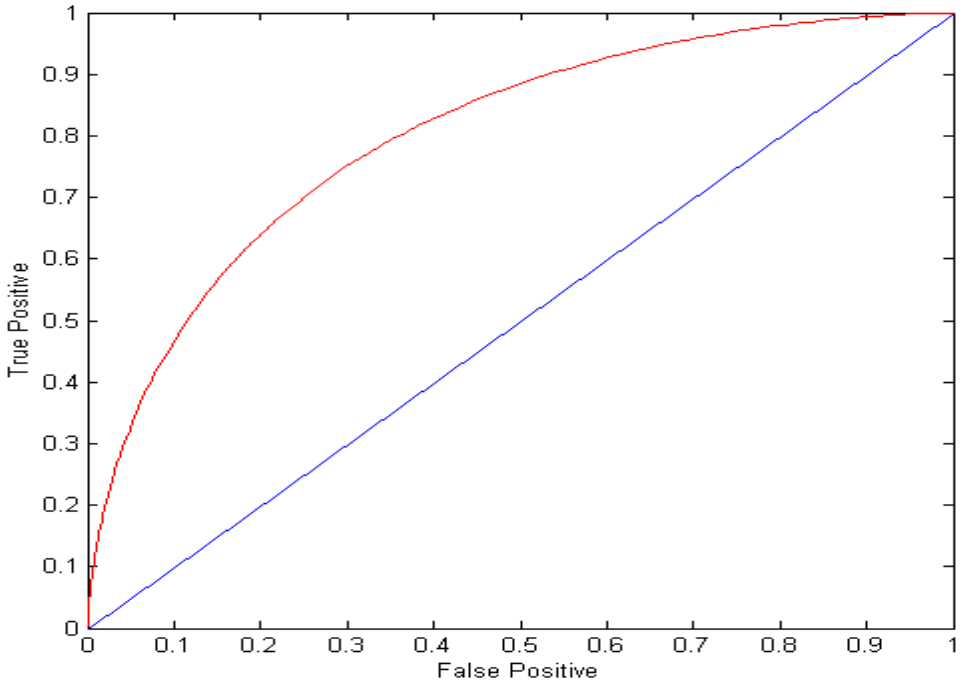
(0,0): declare everything to be negative class

(1,1): declare everything to be positive class

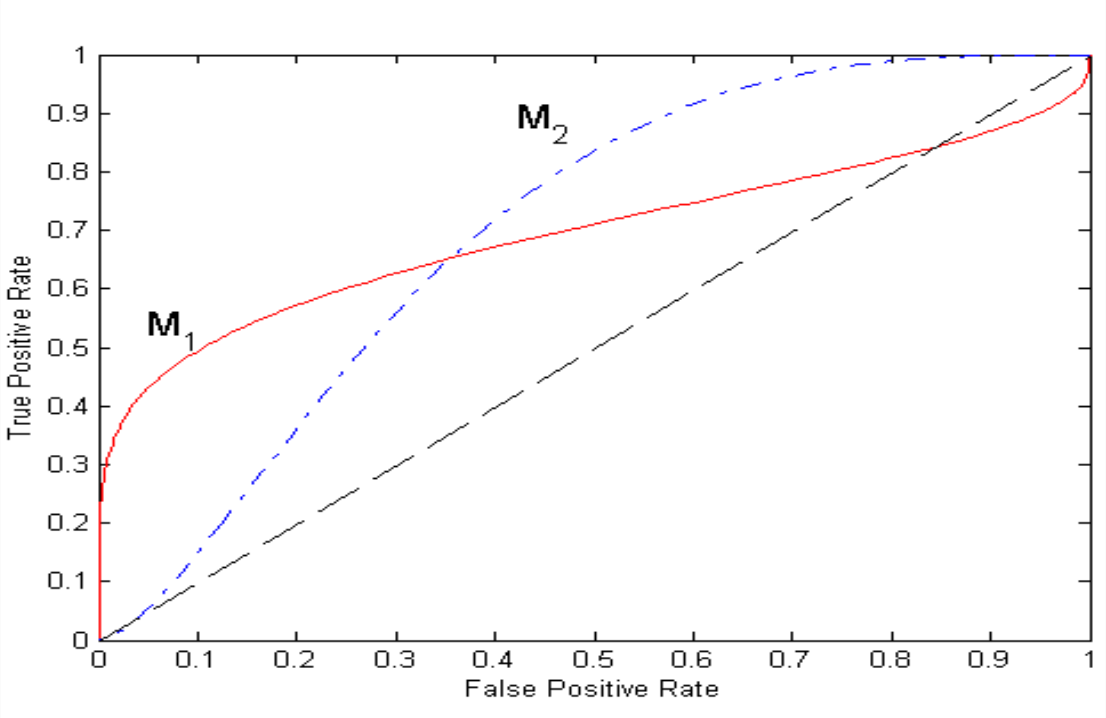
(1,0): ideal

Diagonal line:

- Random guessing
- Below diagonal line:
  - prediction is opposite of the true class



Using ROC for Model Comparison



- No model consistently outperform the other
  - M_1 is better for small FPR
  - M_2 is better for large FPR
- Area Under the ROC curve
  - Ideal:
    - Area = 1
  - Random guess:
    - Area = 0.5

How to Construct an ROC curve

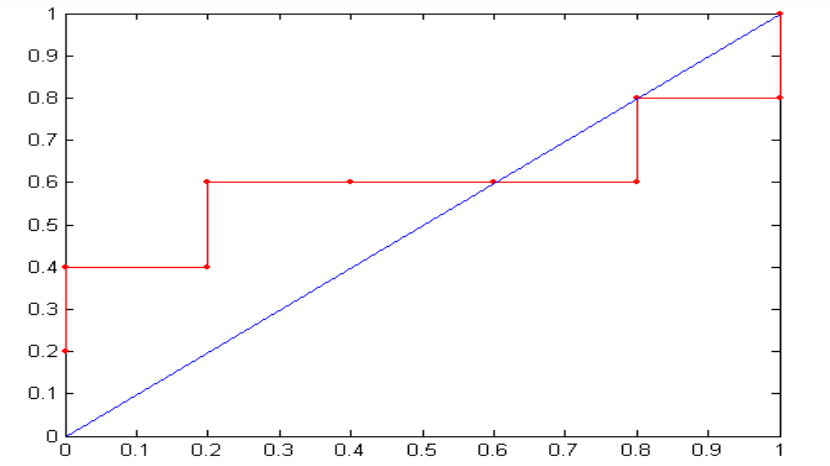
Instance	P(+|A)	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	+
5	0.85	-
6	0.85	-
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

- Use classifier that produces posterior probability for each test instance $P(+ \mid A)$
- Sort the instances according to $P(+ \mid A)$ in decreasing order
- Apply threshold at each unique value of $P(+ \mid A)$
- Count the number of TP, FP, TN, FN at each threshold
- TP rate, TPR = TP/(TP+FN)
- FP rate, FPR = FP/(FP + TN)

How to construct an ROC curve

	Class	+	-	+	-	-	-	+	-	+	+	
Threshold >=		0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
	TP	5	4	4	3	3	3	3	2	2	1	0
	FP	5	5	4	4	3	2	1	1	0	0	0
	TN	0	0	1	1	2	3	4	4	5	5	5
	FN	0	1	1	2	2	2	2	3	3	4	5
→	TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
→	FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

ROC Curve:



Test of Significance

Given two models:

- Model M1: accuracy = 85%, tested on 30 instances
- Model M2: accuracy = 75%, tested on 5000 instances

Can we say M1 is better than M2?

- How much confidence can we place on accuracy of M1 and M2?
- Can the difference in performance measure be explained as a result of random fluctuations in the test set?

Confidence Interval for Accuracy

- Prediction can be regarded as a Bernoulli trial
  - A Bernoulli trial has 2 possible outcomes
  - Possible outcomes for prediction: correct or wrong
  - Collection of Bernoulli trials has a Binomial distribution:
    - $x \sim Bin(N, p)$ x: number of correct predictions
    - e.g: Toss a fair coin 50 times, how many heads would turn up?
      Expected number of heads = $N \times p = 50 \times 0.5 = 25$

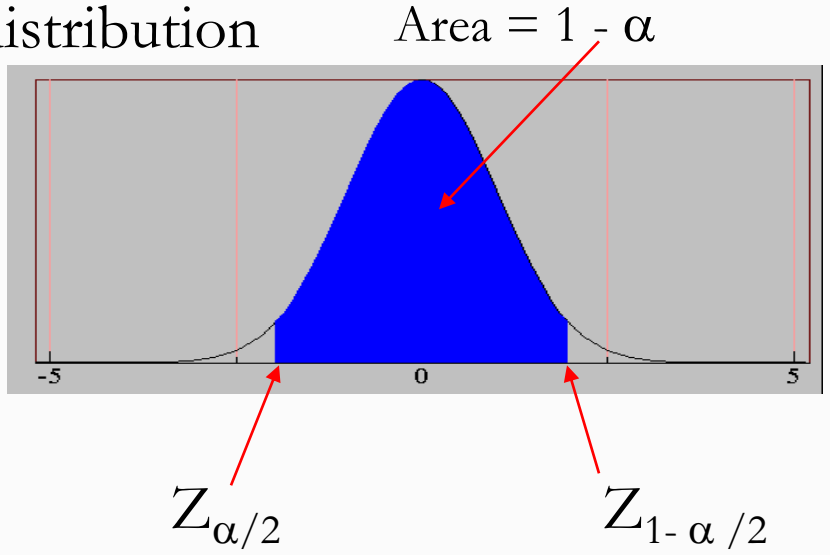
- x is given by a binomial distribution with mean Np and variance Np(1-p)
- acc is given by a binomial distribution with mean p and variance p(1-p)/N
- Given x (# of correct predictions) or equivalently, acc=x/N, and N (# of test instances), Can we predict p (true accuracy of model)?

Confidence Interval for Accuracy

For large test sets (N > 30),

- acc can be approximated by a normal distribution with mean p and variance p(1-p)/N

$$P(Z_{\alpha/2} < \frac{acc - p}{\sqrt{p(1-p)/N}} < Z_{1-\alpha/2}) = 1 - \alpha$$



Confidence Interval for p:

$$p = \frac{2 \times N \times acc + Z^2_{\alpha/2} \pm \sqrt{Z^2_{\alpha/2} + 4 \times N \times acc - 4 \times N \times acc^2}}{2(N + Z^2_{\alpha/2})}$$

Confidence Interval for Accuracy

Consider a model that produces an accuracy of 80% when evaluated on 100 test instances:

- N=100, acc = 0.8
- Let 1-α = 0.95 (95% confidence)
- From probability table, $Z_{\alpha/2}$=1.96

N	50	100	500	1000	5000
p(lower)	0.670	0.711	0.763	0.774	0.789
p(upper)	0.888	0.866	0.833	0.824	0.811

1-α	Z
0.99	2.58
0.98	2.33
0.95	1.96
0.90	1.65

Comparing Performance of 2 Models

Given two models, say M1 and M2, which is better?

- M1 is tested on D1 (size=n1), found error rate = e_1
- M2 is tested on D2 (size=n2), found error rate = e_2
- Assume D1 and D2 are independent
- If n1 and n2 are sufficiently large, then

$$e_1 \sim N(\mu_1, {\sigma_1}^2)$$
$$e_2 \sim N(\mu_2, {\sigma_2}^2)$$

- Approximate:

$$\hat{\sigma}_i^{\,2} = \frac{e_i(1-e_i)}{n_i}$$

Comparing Performance of 2 Models

To test if performance difference is statistically significant: d = e1 – e2

- $d \sim N(d_t, \sigma_t^2)$ where d_t is the true difference

- Since D1 and D2 are independent, their variance adds up:

$$\sigma_t^2 = \sigma_1^2 + \sigma_2^2 \cong \hat{\sigma}_1^2 + \hat{\sigma}_2^2$$
$$= \frac{e1(1-e1)}{n1} + \frac{e2(1-e2)}{n2}$$

- At (1-α) confidence level, $d_t = d \pm Z_{\alpha/2}\hat{\sigma}_t$

An Illustrative Example

Given: M1: n1 = 30, e1 = 0.15

M2: n2 = 5000, e2 = 0.25

d = |e2 – e1| = 0.1 (2-sided test)

$$\hat{\sigma}_d^2 = \frac{0.15(1-0.15)}{30} + \frac{0.25(1-0.25)}{5000} = 0.0043$$

At 95% confidence level, $Z_{\alpha/2}=1.96$

$$d_t = 0.100 \pm 1.96 \times \sqrt{0.0043} = 0.100 \pm 0.128$$

=> Interval contains 0 => difference may not be statistically significant

Comparing Performance of 2 Algorithms

Each learning algorithm may produce k models:

- L1 may produce M11 , M12, …, M1k
- L2 may produce M21 , M22, …, M2k

If models are generated on the same test sets D1,D2, …, Dk (e.g., via cross-validation)

- For each set: compute $d_j = e_{1j} - e_{2j}$
- d_j has mean d_t and variance σ_t^2
- Estimate:

$$\hat{\sigma}_t^2 = \frac{\sum_{j=1}^{k} (d_j - \bar{d})^2}{k(k-1)}$$

$$d_t = d \pm t_{1-\alpha,k-1} \hat{\sigma}_t$$