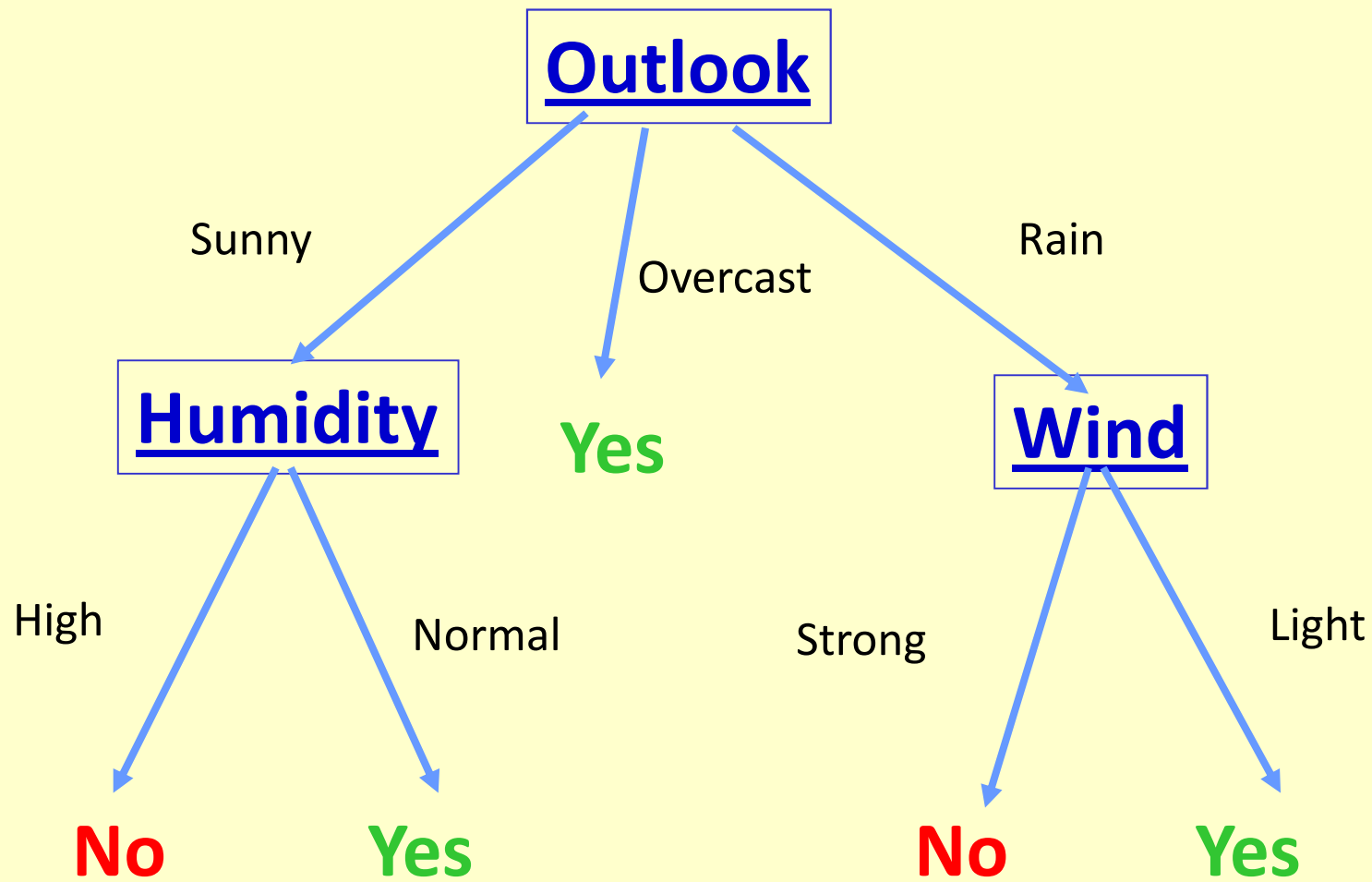


Prediction

Prediction

- What is prediction
- Simple methods for prediction
- **Classification by decision tree induction**
- **Classification and Regression evaluation**



Decision tree induction

- Decision tree **generation** consists of **two phases**
 - **Tree construction**
 - At start, all the training examples are at the root
 - **Partition** examples **recursively** based on selected attributes
 - **Tree pruning**
 - Identify and remove branches that **reflect noise** or outliers
- **Prefer simplest** tree (Occam's razor)
 - The simplest tree captures the most generalization and hopefully represents the most essential relationships

Dataset subsets

- **Training set** – used in model construction
- **Test set** – used in model validation
- **Pruning set** – used in model construction
 - (30% of training set)
- Train/test (70% / 30%)

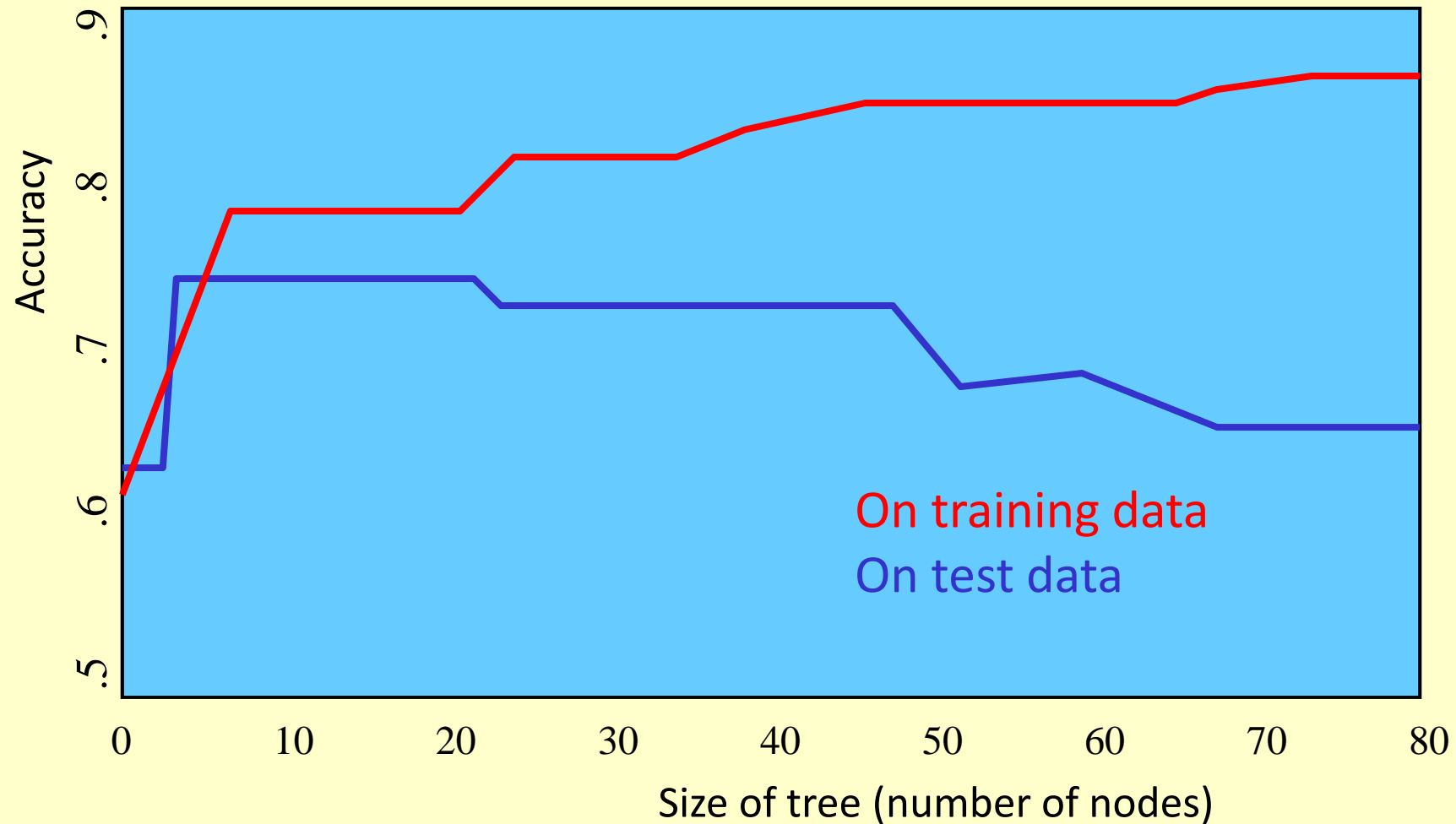
PRUNING TO AVOID OVERFITTING

Avoid Overfitting in Classification

- Ideal goal of classification:
 - Find the **simplest** decision tree that **fits the data** and **generalizes to unseen data**
 - Intractable in general
- The **generated tree may overfit** the training data
 - Too many branches, some may reflect anomalies due to noise, outliers or too little training data
 - erroneous attribute values
 - erroneous classification
 - too sparse training examples
 - insufficient set of attributes
 - Result in **poor accuracy** for unseen samples

Overfitting and accuracy

- Typical relation between tree size and accuracy:



Pruning to avoid overfitting

- **Prepruning:** Stop growing the tree when there is not enough data to make reliable decisions or when the examples are acceptably homogenous
 - Do not split if there is not enough data in the node to have a good prediction
 - Do not split a node if this would result in the goodness measure falling below a threshold (e.g. InfoGain)
 - Difficult to choose an appropriate threshold
- **Postpruning:** Grow the full tree, then remove nodes for which there is not sufficient evidence
 - Replace a split (subtree) with a leaf if the *predicted* validation error is no worse than the more complex tree (use \neq dataset)
- Prepruning easier, but **postpruning works better**
- Prepruning - hard to know when to stop

Prepruning

- Based on statistical significance test
 - Stop growing the tree when there is no *statistically significant* association between any attribute and the class at a particular node
- Most popular test: *chi-squared test*
- ID3 used chi-squared test in addition to information gain
 - Only statistically significant attributes were allowed to be selected by information gain procedure

chi-squared test

- Allows to compare the cells frequencies with the frequencies that would be obtained if the variables were independent

Civil status	Yes	No
Married		
Single		
Widowed		
Divorced		
Total		

- In this case, compare the overall frequency of yes and no with the frequencies in the attribute branches. If the difference is small the attribute does not add enough value to the decision

	A	B	C	D	E
1	Contingency Tables-Actual	Voting preferences			Total
2	Gender	Republican	Democrat	Independent	
3	Male	200	150	50	400
4	Female	250	300	50	600
5	Total	450	450	100	1000
6	Expected Values Data Table				
7	Condition	Republican	Democrat	Independent	
8	Male	180	180	40	
9	Female	270	270	60	0.0003

Methods for postpruning

- **Reduced-error pruning**
 - Split data into training & validation sets
 - Build full decision tree from training set
 - For every non-leaf node N
 - Prune subtree rooted by N, replace with majority class. Test accuracy of pruned tree on validation set, that is, check if the pruned tree performs no worse than the original over the validation set
 - Greedily remove the subtree that results in greatest improvement in accuracy on validation set
- **Sub-tree raising** (more complex)
 - An entire sub-tree is raised to replace another sub-tree.

Estimating error rates

- Prune only if it reduces the estimated error
- Error on the training data is NOT a useful estimator
Q: Why it would result in very little pruning?
- Use hold-out set for pruning (“reduced-error pruning”)
- C4.5’s method
 - Derive confidence interval from training data
 - Use a heuristic limit, derived from this, for pruning
 - Standard Bernoulli-process-based method (Binomial distribution)
 - Shaky statistical assumptions (based on training data)

Estimating error rates

- How to decide if we should replace a node by a leaf?
 - Use an independent **test set to estimate the error** (reduced error pruning) -> less data to train the tree
 - Make estimates of the error based on the training data (C4.5)
 - The majority class is chosen to represent the node
 - Count the number of errors, $\#e/N = \text{error rate}$
 - Establish a **confidence interval and use the upper limit**, pessimistic estimate of the error rate.
 - Compare such estimate with the combined estimate of the error estimates for the leaves
 - If the first is smaller replace the node by a leaf.

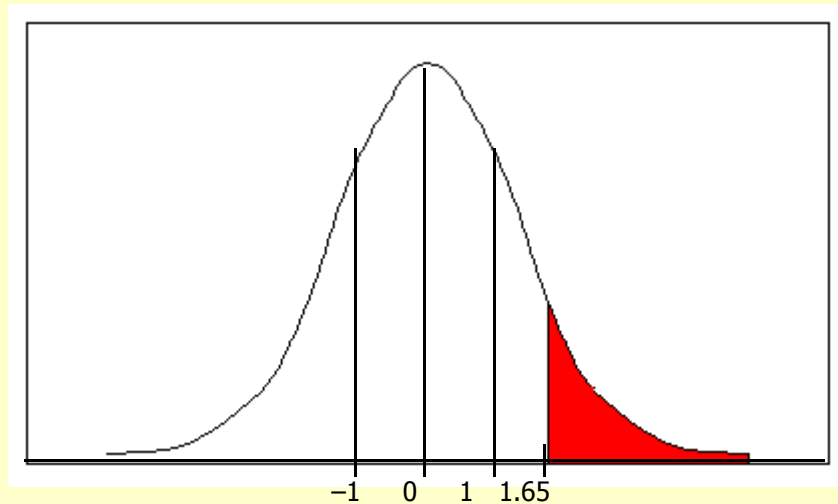
*Mean and variance

- Mean and variance for a Bernoulli trial:
 $p, p(1-p)$
- Expected success rate $f=S/N$
- Mean and variance for $f : p, p(1-p)/N$
- For large enough N , f follows a Normal distribution
- $c\%$ confidence interval $[-z \leq X \leq z]$ for random variable with 0 mean is given by:
- With a symmetric distribution: $\Pr[-z \leq X \leq z] = c$

$$\Pr[-z \leq X \leq z] = 1 - 2 \times \Pr[X \geq z]$$

*Confidence limits

- Confidence limits for the normal distribution with 0 mean and a variance of 1:



Pr[X ≥ z]	z
0.1%	3.09
0.5%	2.58
1%	2.33
5%	1.65
10%	1.28
20%	0.84
25%	0.69
40%	0.25

- Thus:

$$\Pr[-1.65 \leq X \leq 1.65] = 90\%$$

- To use this we have to reduce our random variable f to have 0 mean and unit variance

*Transforming f

- Transformed value for f : $\frac{f - p}{\sqrt{p(1-p)/N}}$
(i.e. subtract the mean and divide by the *standard deviation*)
- Resulting equation: $\Pr\left[-z \leq \frac{f - p}{\sqrt{p(1-p)/N}} \leq z\right] = c$
- Solving for p : $p = \left(f + \frac{z^2}{2N} \pm z \sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}} \right) / \left(1 + \frac{z^2}{N} \right)$

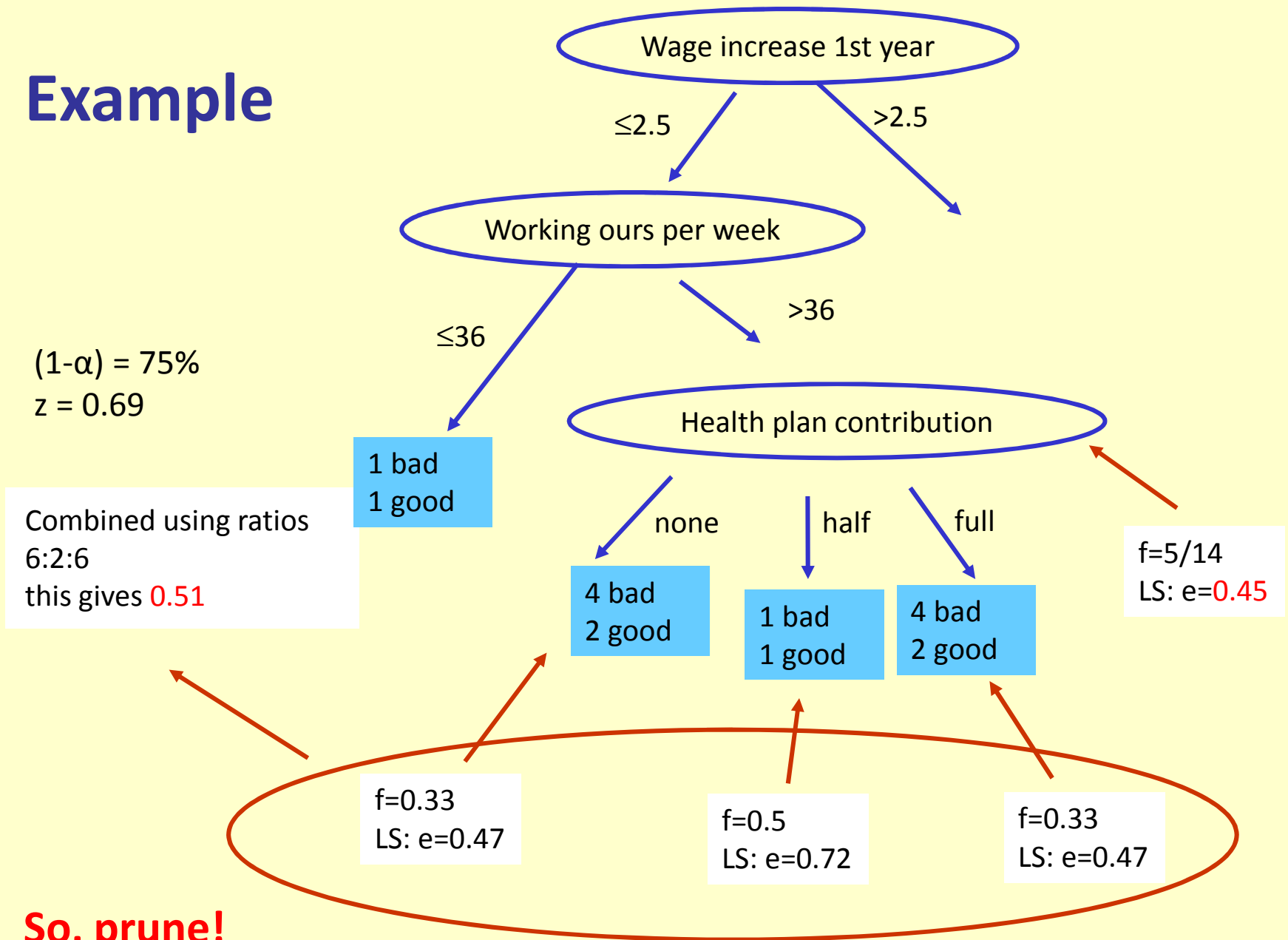
C4.5's method

- Error estimate for subtree is weighted sum of error estimates for all its leaves
- Error estimate for a node (upper bound):

$$e = p = \left(f + \frac{z^2}{2N} + z \sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}} \right) / \left(1 + \frac{z^2}{N} \right)$$

- If $c = 25\%$ then $z = 0.69$ (from normal distribution)
- f is the error rate on the training data
- N is the number of instances covered by the leaf

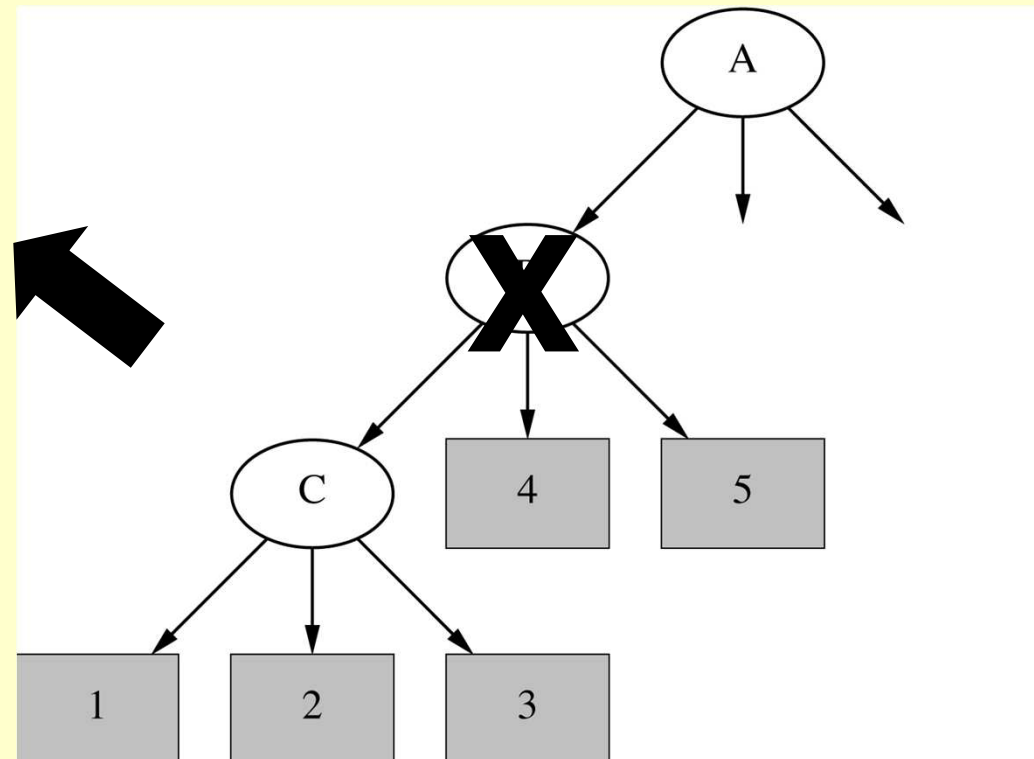
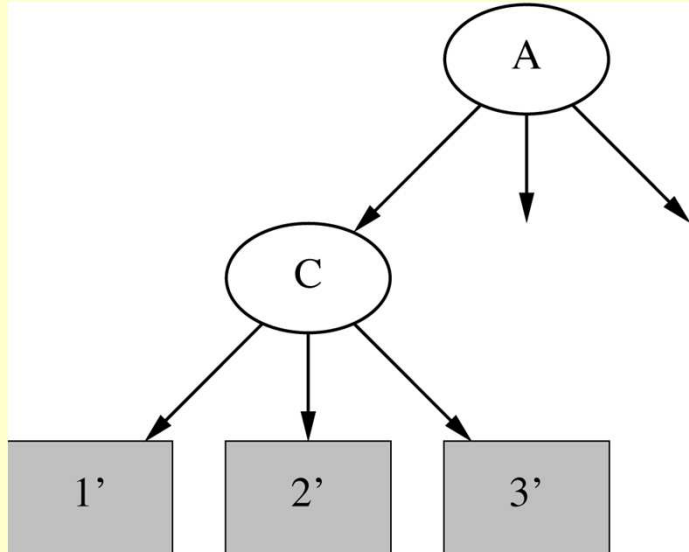
Example



*Subtree raising

- Delete node
- Redistribute instances
- Slower than subtree replacement

(Worthwhile?)



FROM TREES TO RULES

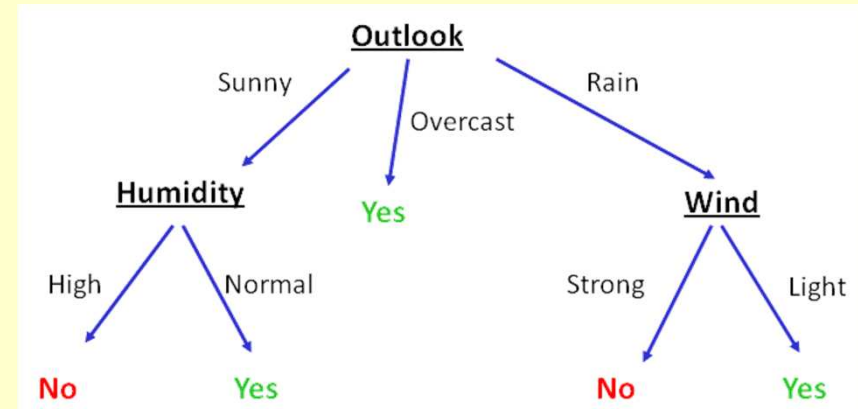
Extracting classification rules from trees

- Simple way:
 - Represent the knowledge in the form of **IF-THEN** rules
 - **One rule** is created **for each path** from the root to a leaf
 - Each attribute-value pair along a path forms a conjunction
 - The **leaf node** holds the **class prediction**
 - Rules are **easier** for humans **to understand**

Extracting classification rules from trees

If-then rules

- **IF** Outlook=Sunny \cap Humidity=Normal **THEN** PlayTennis=Yes
- **IF** Outlook=Overcast **THEN** PlayTennis=Yes
- **IF** Outlook=Rain \cap Wind=Weak **THEN** PlayTennis=Yes
- **IF** Outlook=Sunny \cap Humidity=High **THEN** PlayTennis=No
- **IF** Outlook=Rain \cap Wind=Strong **THEN** PlayTennis=No



Is Saturday morning OK for playing tennis?

- Outlook=Sunny, Temperature=Hot, Humidity=High, Wind=Strong
- PlayTennis = No, because Outlook=Sunny \cap Humidity=High

From trees to rules

- C4.5rules: greedily prune conditions from each rule if this reduces its estimated error
 - Can produce duplicate rules
 - Check for this at the end
- Then
 - look at each class in turn
 - consider the rules for that class
 - find a “good” subset (guided by MDL)
- Then rank the subsets to avoid conflicts
- Finally, remove rules (greedily) if this decreases error on the training data

C4.5rules: choices and options

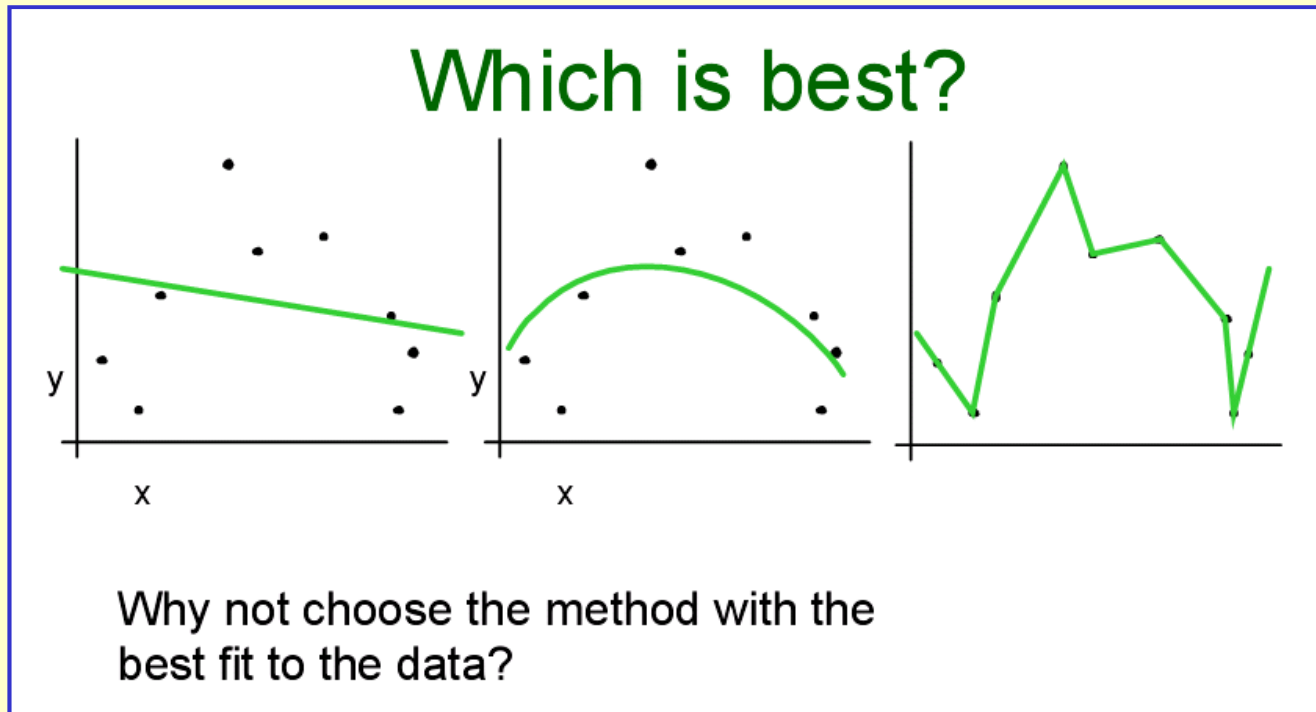
- C4.5rules slow for large and noisy datasets
- Commercial version C5.0rules uses a different technique
 - Much faster and a bit more accurate
- C4.5 has two parameters
 - Confidence value (default 25%):
lower values incur heavier pruning
 - Minimum number of instances in the two most popular branches
(default 2)

EVALUATING A DECISION TREE

Classification Accuracy

- **How predictive** is the model we learned?
- Error on the **training data is *not* a good indicator** of performance on future data
 - **Q: Why?**
 - A: Because new data will probably not be **exactly** the same as the training data!
- **Overfitting** – fitting the training data too precisely - usually leads to poor results on new data

Overfitting

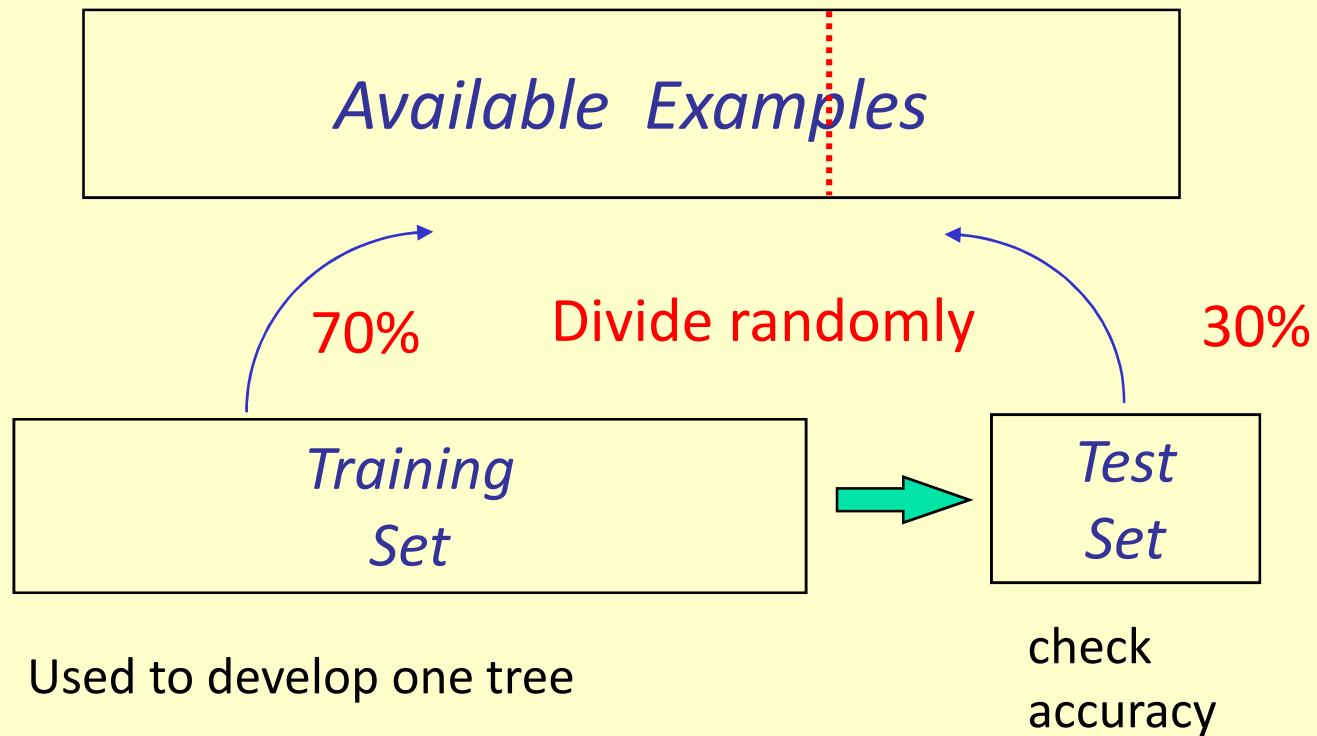


How well is the model going to predict future data?

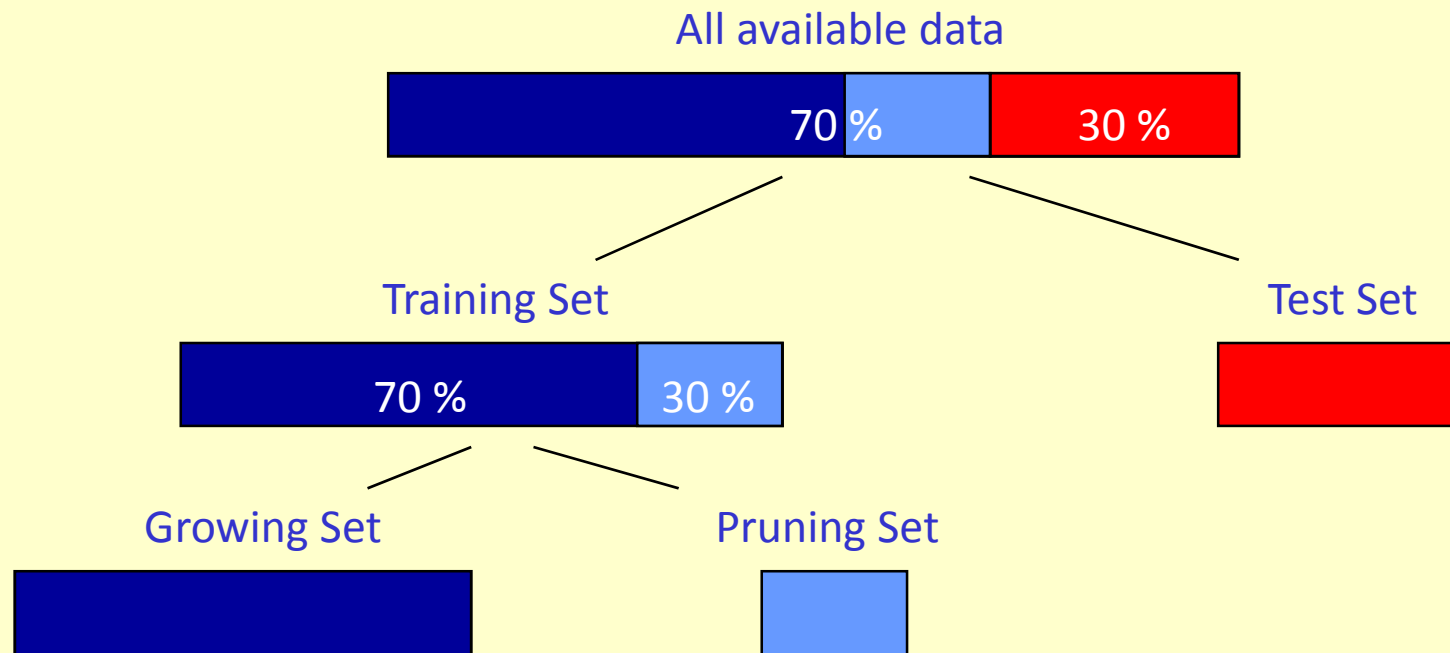
Evaluation on “LARGE” data

- If many (thousands) of examples are available, including several hundred examples from each class, then a simple evaluation is sufficient
 - Randomly split data into training and test sets
 - (usually 2/3 for train and 1/3 for test)
- Build a classifier using the train set and evaluate it using the test set.

evaluation - usual procedure



Typical proportions



Problem with using "Pruning Set": less data for "Growing Set"

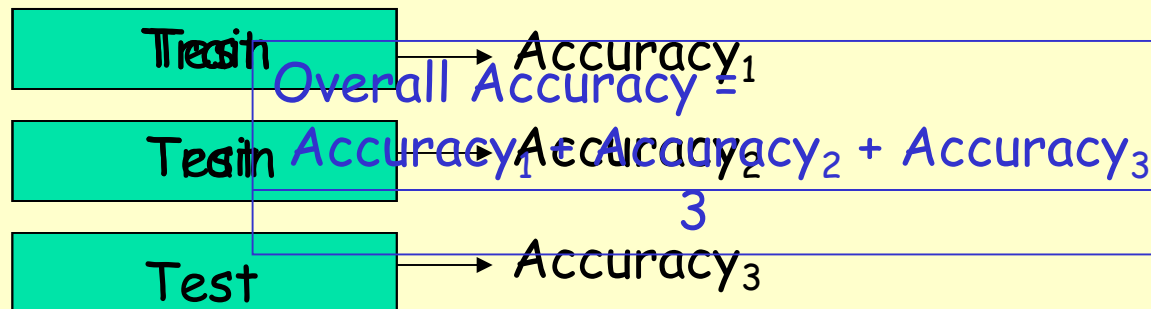
Evaluation on “small” data

- *Cross-validation*
 - **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

Tree evaluation - cross validation

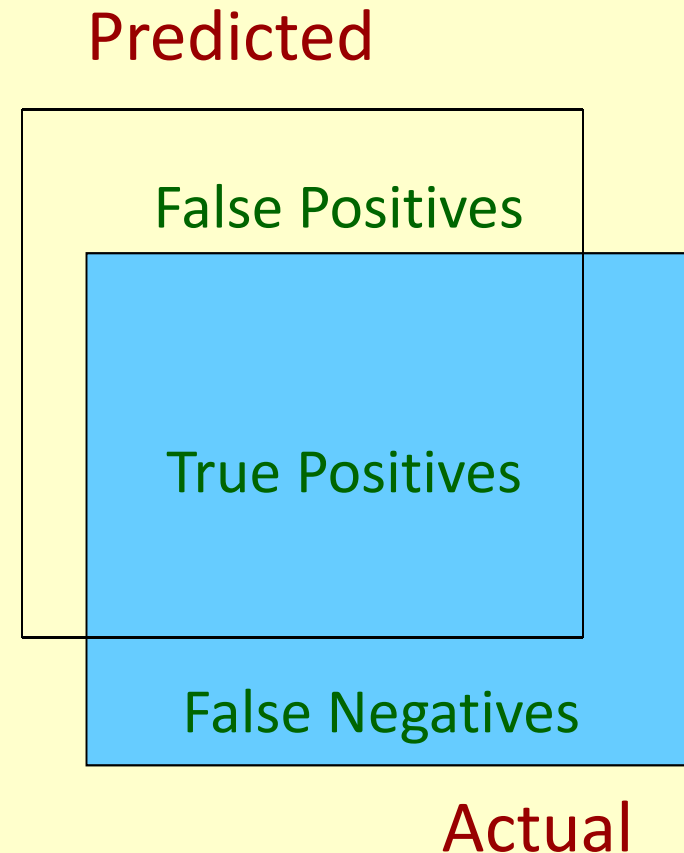
- Method for training and testing on the same set of data
 - Useful when training data is limited
1. Divide the training set into N partitions (usually 10)
 2. Do N experiments: each partition is used once as the validation set, and the other N-1 partitions are used as the training set.

The best model is chosen



Evaluation of classification systems

- **Training Set:** examples with class values for learning.
- **Test Set:** examples with class values for evaluating.
- **Evaluation:** Hypotheses are used to infer classification of examples in the test set; inferred classification is compared to known classification.
- **Accuracy:** percentage of examples in the test set that are classified correctly.



Types of possible outcomes

- Spam example
- Two types of errors:
 - **False positive**: classify a good email as spam
 - **False negative**: classify a spam email as good
- Two types of good decisions:
 - **True positive**: classify a spam email as spam
 - **True negative**: classify a good email as good

Confusion matrix

		Actual Class	
		Y	N
Predicted class	Y	A: True +	B: False +
	N	C: False -	D: True -

Entries are counts of correct classifications and counts of errors

Evaluating Classification

- Which goal we have:
 - minimize the number of classification errors
 - minimize the total cost of misclassifications
- In some cases FN and FP have different associated costs
 - spam vs. non spam
 - medical diagnosis
- We can define a cost matrix in order to associate a cost with each type of result. This way we can replace the success rate by the corresponding average cost.

Example Misclassification Costs

Diagnosis of Appendicitis

- Cost Matrix: $C(i,j)$ = cost of predicting class i when the true class is j

Predicted State of Patient	True State of Patient	
	Positive	Negative
Positive	1	1
Negative	100	0

Estimating Expected Misclassification Cost

- Let M be the *confusion matrix* for a classifier: $M(i,j)$ is the number of test examples that are predicted to be in class i when their true class is j

Predicted State of Patient	True State of Patient	
	Positive	Negative
Positive	1	1
Negative	100	0

Predicted Class	True Class	
	Positive	Negative
Positive	40	16
Negative	8	36

$$\text{Cost} = 1 * 40 + 1 * 16 + 100 * 8 + 0 * 36$$

Reduce the 4 numbers to two rates

- True Positive Rate = TP = $(\#TP)/(\#P)$
- False Positive Rate = FP = $(\#FP)/(\#N)$

True	Predicted	
	pos	neg
pos	40	60
neg	30	70

Classifier 1

TP = 0.4

FP = 0.3

True	Predicted	
	pos	neg
pos	70	30
neg	50	50

Classifier 2

TP = 0.7

FP = 0.5

True	Predicted	
	pos	neg
pos	60	40
neg	20	80

Classifier 3

TP = 0.6

FP = 0.2

Direct Marketing Paradigm

- Find most likely prospects to contact
- Not everybody needs to be contacted
- Number of targets is usually much smaller than number of prospects

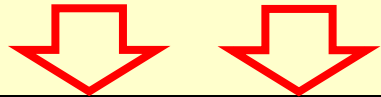
- Typical Applications
 - retailers, catalogues, direct mail (and e-mail)
 - customer acquisition, cross-sell, attrition prediction
 - ...

Direct Marketing Evaluation

- Accuracy on the entire dataset is not the right measure
- Approach
 - develop a target model
 - score all prospects and rank them by decreasing score
 - select top P% of prospects for action
- How to decide what is the best selection?

Model-Sorted List

Use a model to assign score to each customer
Sort customers by decreasing score
Expect more targets (hits) near the top of the list



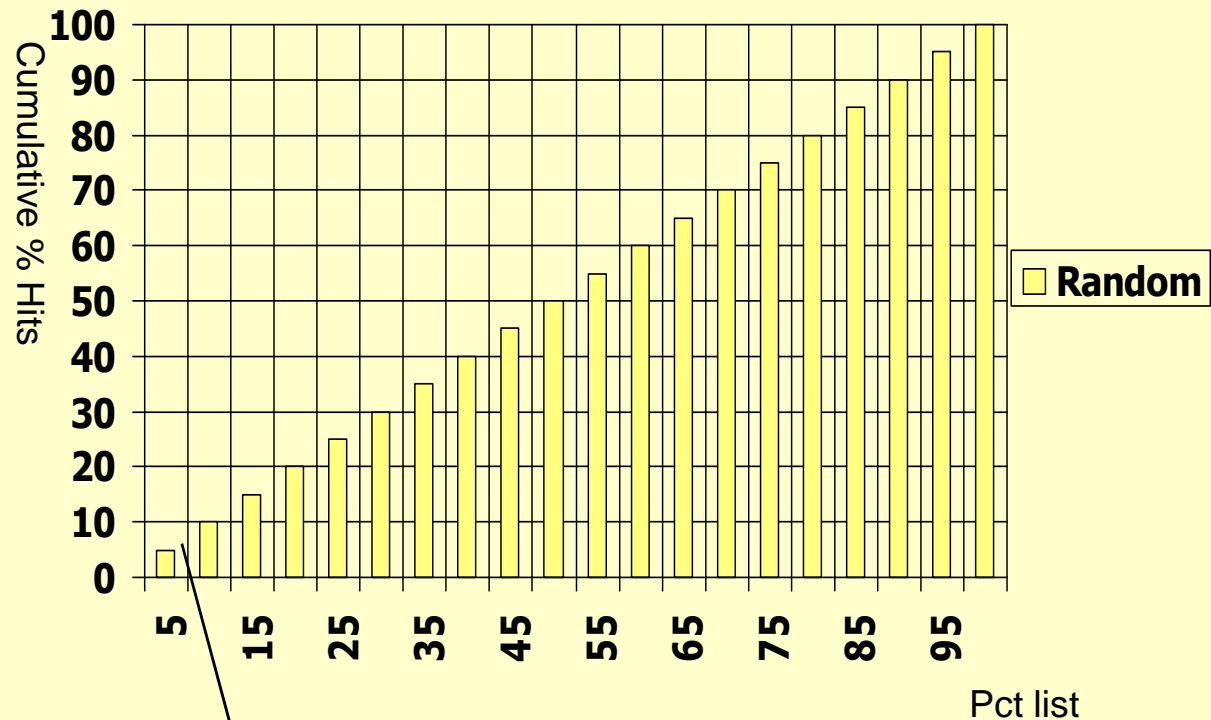
No	Score	Target	CustID	Age	
1	0.97	Y	1746	...	
2	0.95	N	1024	...	
3	0.94	Y	2478	...	
4	0.93	Y	3820	...	
5	0.92	N	4897	...	
...	
99	0.11	N	2734	...	
100	0.06	N	2422		

3 hits in top 5% of the list

If there are 15 targets overall,
then top 5 has $3/15=20\%$ of
targets

CPH (Cumulative Percentage Hits)

Definition:
CPH(P,M)
= % of all targets
in the first P%
of the list scored
by model M
CPH frequently
called Gains



5% of random list have 5% of targets

Q: What is expected value for CPH(P,Random) ?

A: Expected value for CPH(P,Random) = P

CPH: Random List vs Model-ranked list

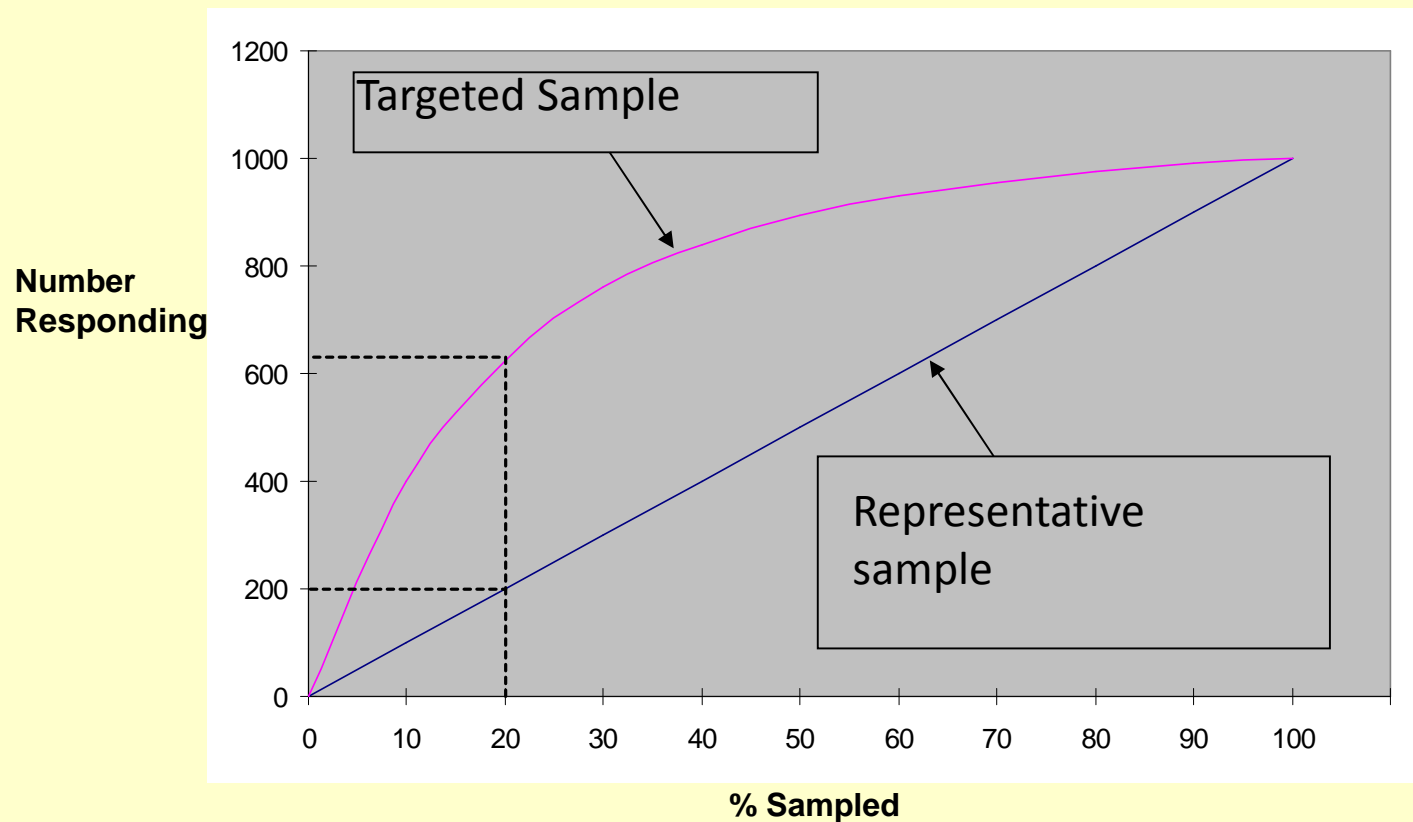


5% of random list have 5% of targets,

but 5% of model ranked list have 21% of targets $CPH(5\%, \text{model})=21\%$.

Comparing models by measuring lift

Absolute number of true positives, instead of a percentage



Targeted vs. mass mailing

Steps in Building a Lift Chart

- 1. First, produce a ranking of the data, using your learned model (classifier, etc):
 - Rank 1 means most likely in + class,
 - Rank n means least likely in + class
- 2. For each ranked data instance, label with Ground Truth label:
 - This gives a list like
 - Rank 1, +
 - Rank 2, -,
 - Rank 3, +,
 - Etc.
- 3. Count the number of true positives (TP) from Rank 1 onwards
 - Rank 1, +, TP = 1
 - Rank 2, -, TP = 1
 - Rank 3, +, TP=2,
 - Etc.
- 4. Plot # of TP against % of data in ranked order (if you have 10 data instances, then each instance is 10% of the data):
 - 10%, TP=1
 - 20%, TP=1
 - 30%, TP=2,
 - ...
- This gives a lift chart.

Generating a lift chart

- Instances are sorted according to their predicted probability of being a true positive:

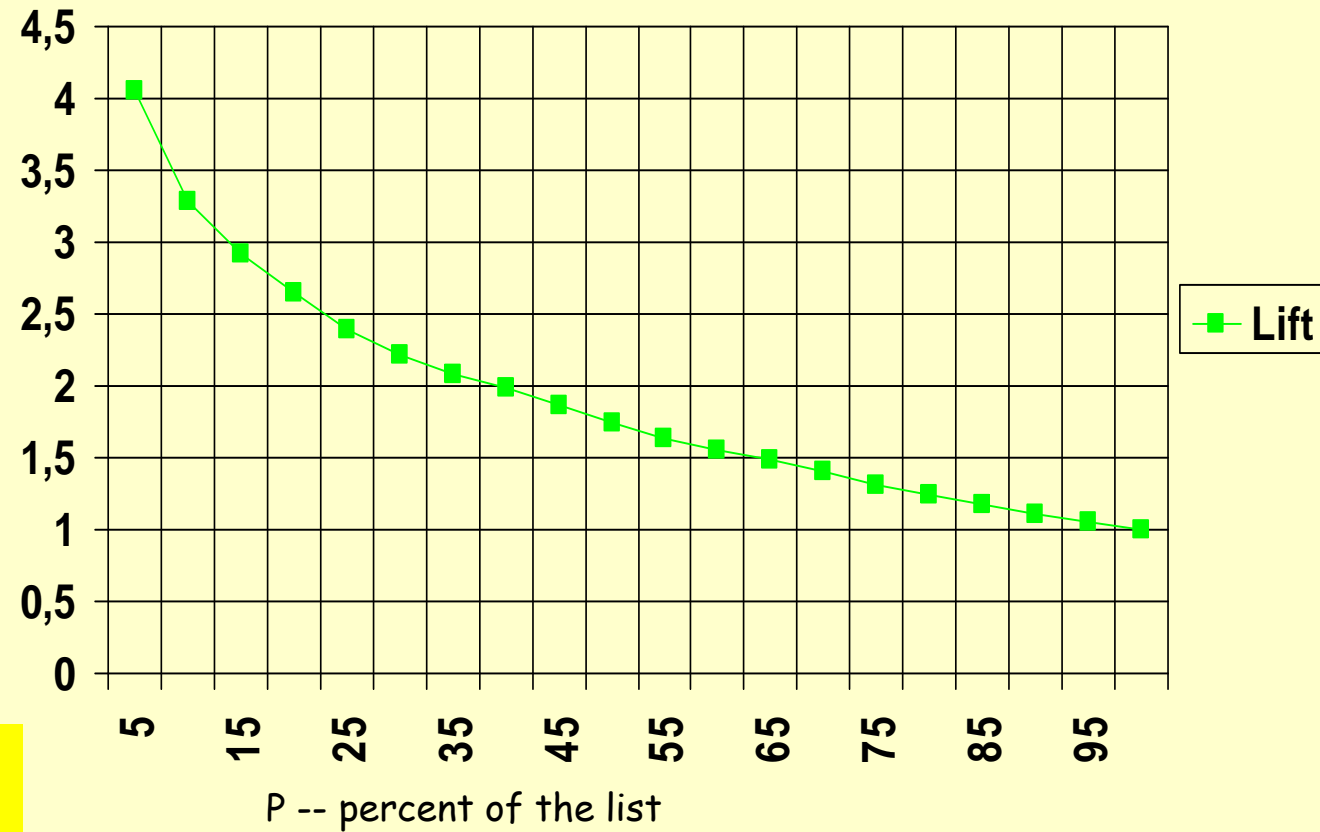
Rank	Predicted probability	Actual class
1	0.95	Yes
2	0.93	Yes
3	0.93	No
4	0.88	Yes
...

- In lift chart, x axis is sample size and y axis is number of true positives

Lift

$$\text{Lift}(P, M) = \text{CPH}(P, M) / P$$

Lift (at 5%)
= 21% / 5%
= 4.2
better
than random



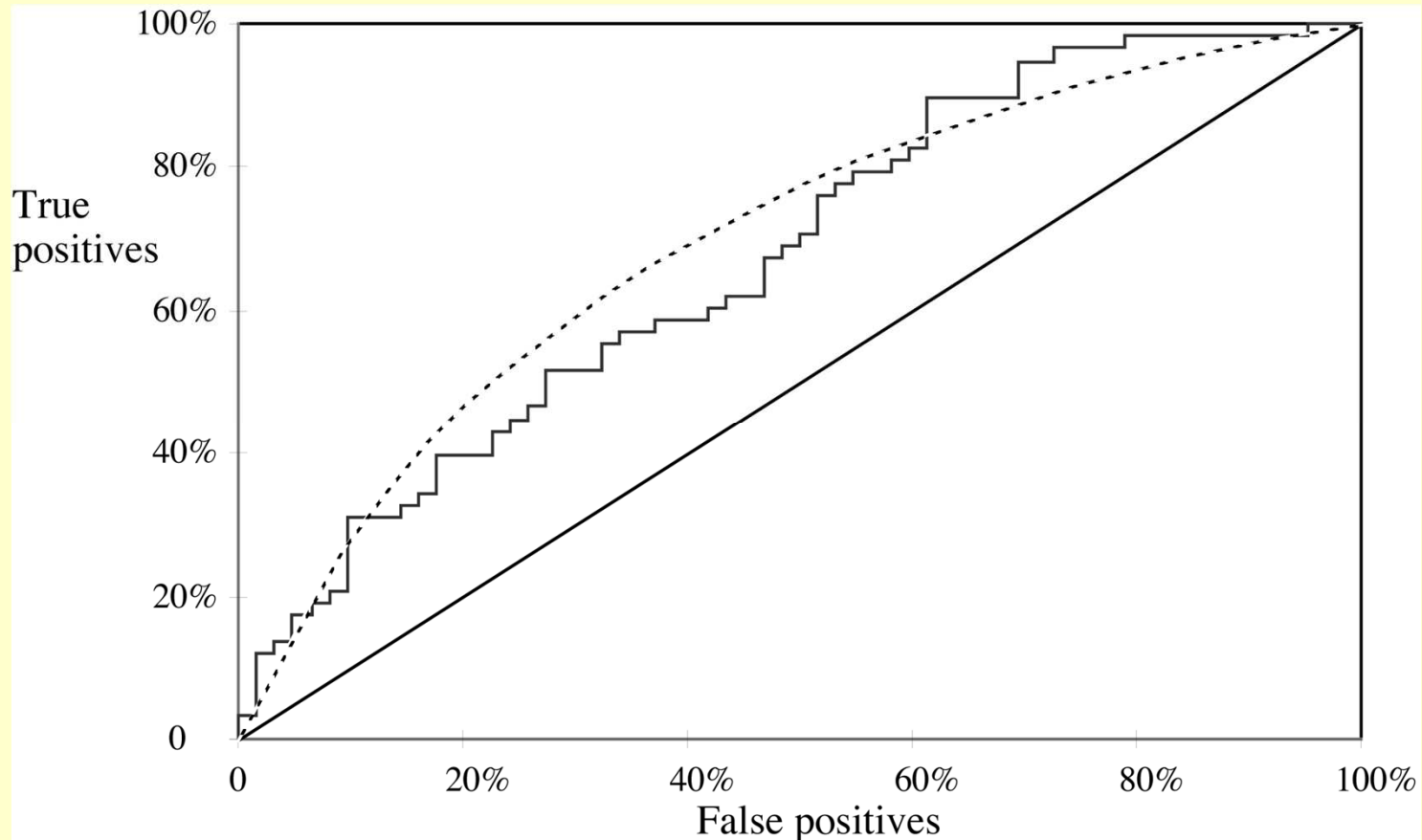
Note: Some (including Witten & Eibe) use "Lift" for what we call CPH.

*ROC curves

- *ROC curves* are similar to CPH (gains) charts
 - Stands for “receiver operating characteristic”
 - Used in signal detection to show tradeoffs between hit rate and false alarm rate over noisy channel
- Differences from Lift chart
 - y axis shows percentage of true positives in sample
rather than absolute number
 - x axis shows percentage of false positives in sample
rather than sample size

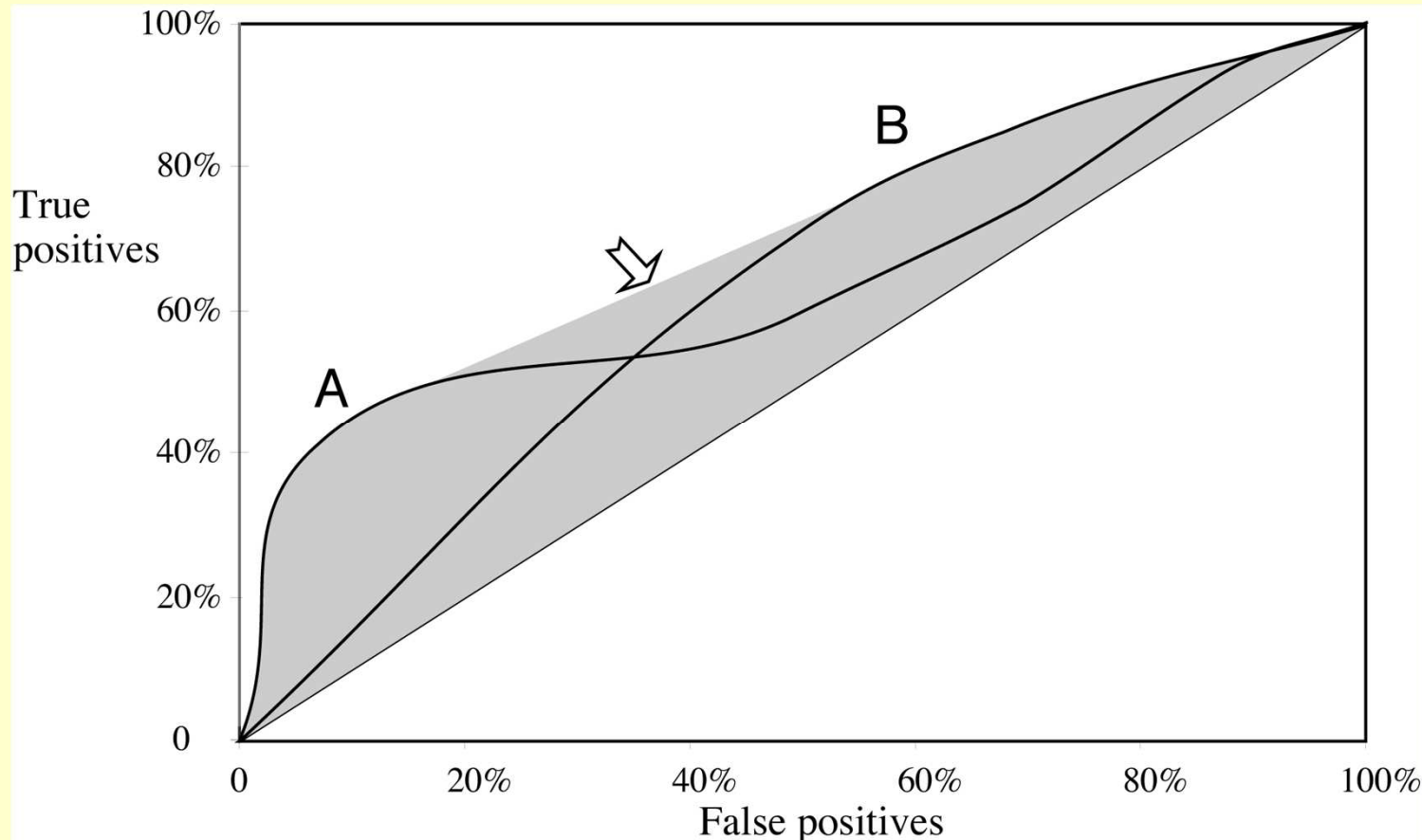
To understand ROC curves go to -> <http://www.anaesthetist.com/mnm/stats/roc/>

*A sample ROC curve



- Jagged curve—one set of test data
- Smooth curve—use cross-validation

*ROC curves for two schemes



- For a small, focused sample, use method A
- For a larger one, use method B
- In between, choose between A and B with appropriate probabilities

Evaluating numeric prediction

- Same strategies: independent test set, cross-validation, significance tests, etc.
- Difference: error measures
- Actual target values: $a_1 a_2 \dots a_n$
- Predicted target values: $p_1 p_2 \dots p_n$
- Most popular measure: *mean-squared error*

$$\frac{(p_1 - a_1)^2 + \dots + (p_n - a_n)^2}{n}$$

- Easy to manipulate mathematically

Other measures for numeric prediction

- The *root mean-squared error* :

$$\sqrt{\frac{(p_1 - a_1)^2 + \dots + (p_n - a_n)^2}{n}}$$

- The *mean absolute error* is less sensitive to outliers than the mean-squared error:

$$\frac{|p_1 - a_1| + \dots + |p_n - a_n|}{n}$$

- Sometimes *relative* error values are more appropriate (e.g. 10% for an error of 50 when predicting 500)

Different Costs

- In practice, true positive and false negative errors often incur different costs
- Examples:
 - Medical diagnostic tests: does X have leukaemia?
 - Loan decisions: approve mortgage for X?
 - Web mining: will X click on this link?
 - Promotional mailing: will X buy the product?
 - ...

Cost Sensitive Learning

- There are two types of errors

		Predicted class	
		Yes	No
Actual class	Yes	TP: True positive	FN: False negative
	No	FP: False positive	TN: True negative

- Machine Learning methods usually minimize FP+FN
- Direct marketing maximizes TP

Cost-sensitive learning

- Most learning schemes do not perform cost-sensitive learning
 - They generate the same classifier no matter what costs are assigned to the different classes
 - Example: standard decision tree learner
- Simple methods for cost-sensitive learning:
 - Re-sampling of instances according to costs
 - Weighting of instances according to costs
- Some schemes are inherently cost-sensitive, e.g. naïve Bayes

Summary

- Classification is an extensively studied problem (mainly in statistics, machine learning & neural networks)
- Classification is probably one of the most widely used data mining techniques with a lot of extensions
- Knowing how to evaluate different classifiers is essential for the process of building a model that is adequate for a given problem

References

- Jiawei Han and Micheline Kamber, “Data Mining: Concepts and Techniques”, 2000
- Ian H. Witten, Eibe Frank, “Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations”, 1999
- Tom M. Mitchell, “Machine Learning”, 1997
- J. Shafer, R. Agrawal, and M. Mehta. “SPRINT: A scalable parallel classifier for data mining”. In VLDB'96, pp. 544-555,
- J. Gehrke, R. Ramakrishnan, V. Ganti. “RainForest: A framework for fast decision tree construction of large datasets.” In VLDB'98, pp. 416-427
- Robert Holt “Cost-Sensitive Classifier Evaluation” (ppt slides)
- James Guszczka, “The Basics of Model Validation”, CAS Predictive Modeling Seminar, September, 2005



Thank you !!!