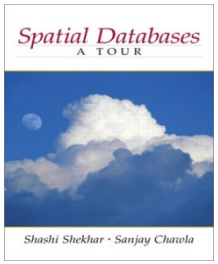
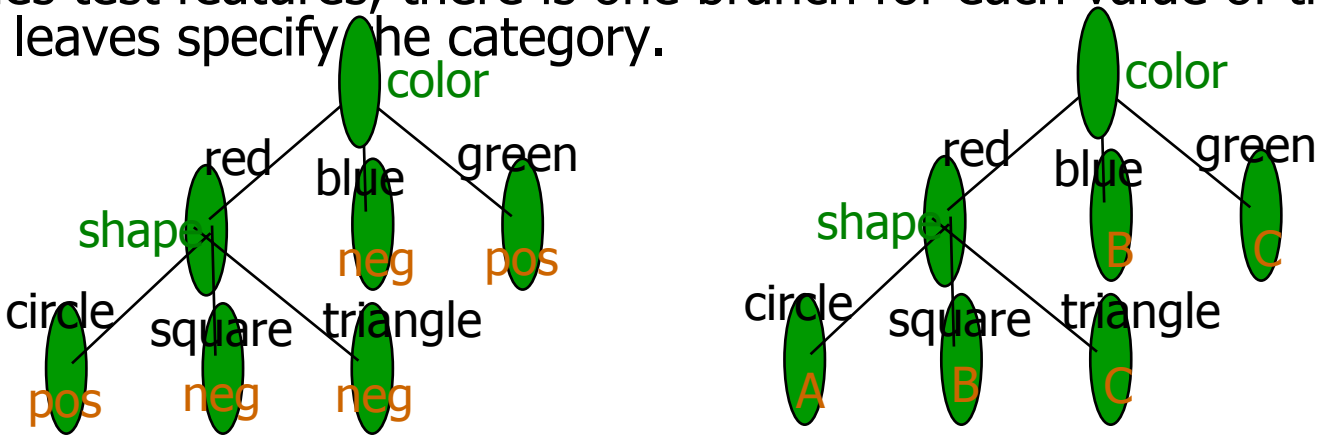


Spatial Data Mining

Overview of Classification Techniques

Decision Trees

- Tree-based classifiers for instances represented as feature-vectors. Nodes test features, there is one branch for each value of the feature, and leaves specify the category.



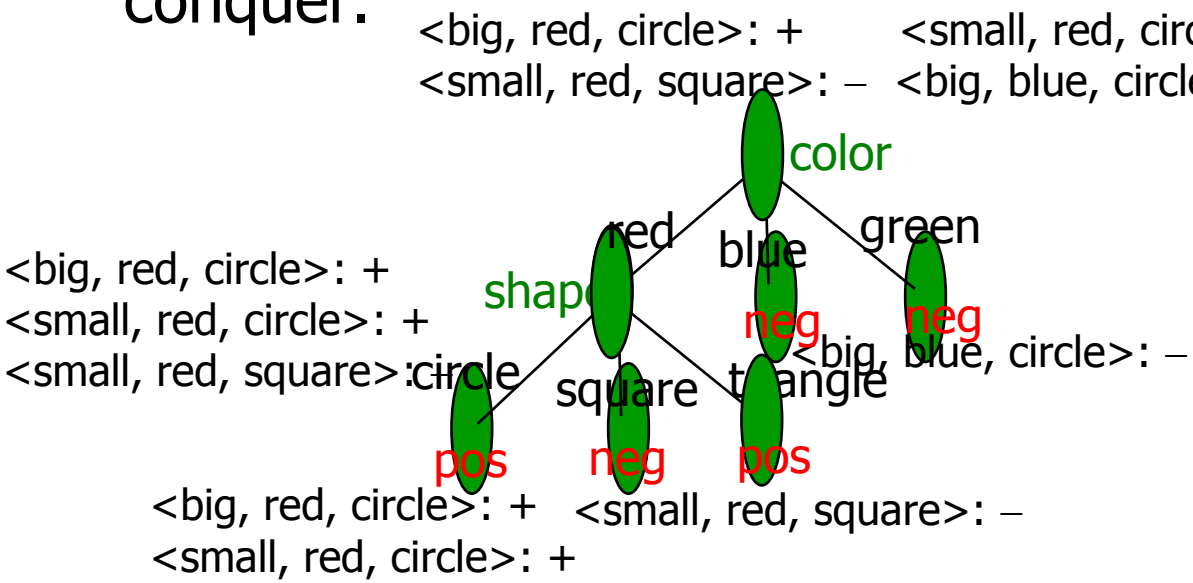
- Can represent arbitrary conjunction and disjunction. Can represent any classification function over discrete feature vectors.
- Can be rewritten as a set of rules, i.e. disjunctive normal form (DNF).
  - red $\wedge$ circle $\rightarrow$ pos
  - red $\wedge$ circle $\rightarrow$ A

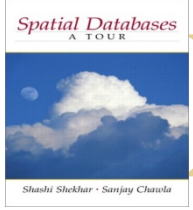
    blue $\rightarrow$ B; red $\wedge$ square $\rightarrow$ B

    green $\rightarrow$ C; red $\wedge$ triangle $\rightarrow$ C

Top-Down Decision Tree Induction

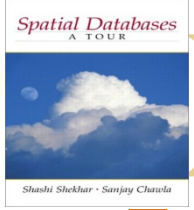
- Recursively build a tree top-down by divide and conquer.





Picking a Good Split Feature

- Goal is to have the resulting tree be as small as possible, per Occam's razor.
- Finding a minimal decision tree (nodes, leaves, or depth) is an NP-hard optimization problem.
- Top-down divide-and-conquer method does a greedy search for a simple tree but does not guarantee to find the smallest.
  - General lesson in ML: "Greed is good."
- Want to pick a feature that creates subsets of examples that are relatively "pure" in a single class so they are "closer" to being leaf nodes.
- There are a variety of heuristics for picking a good test, a popular one is based on information gain that originated with the ID3 system of Quinlan (1979).



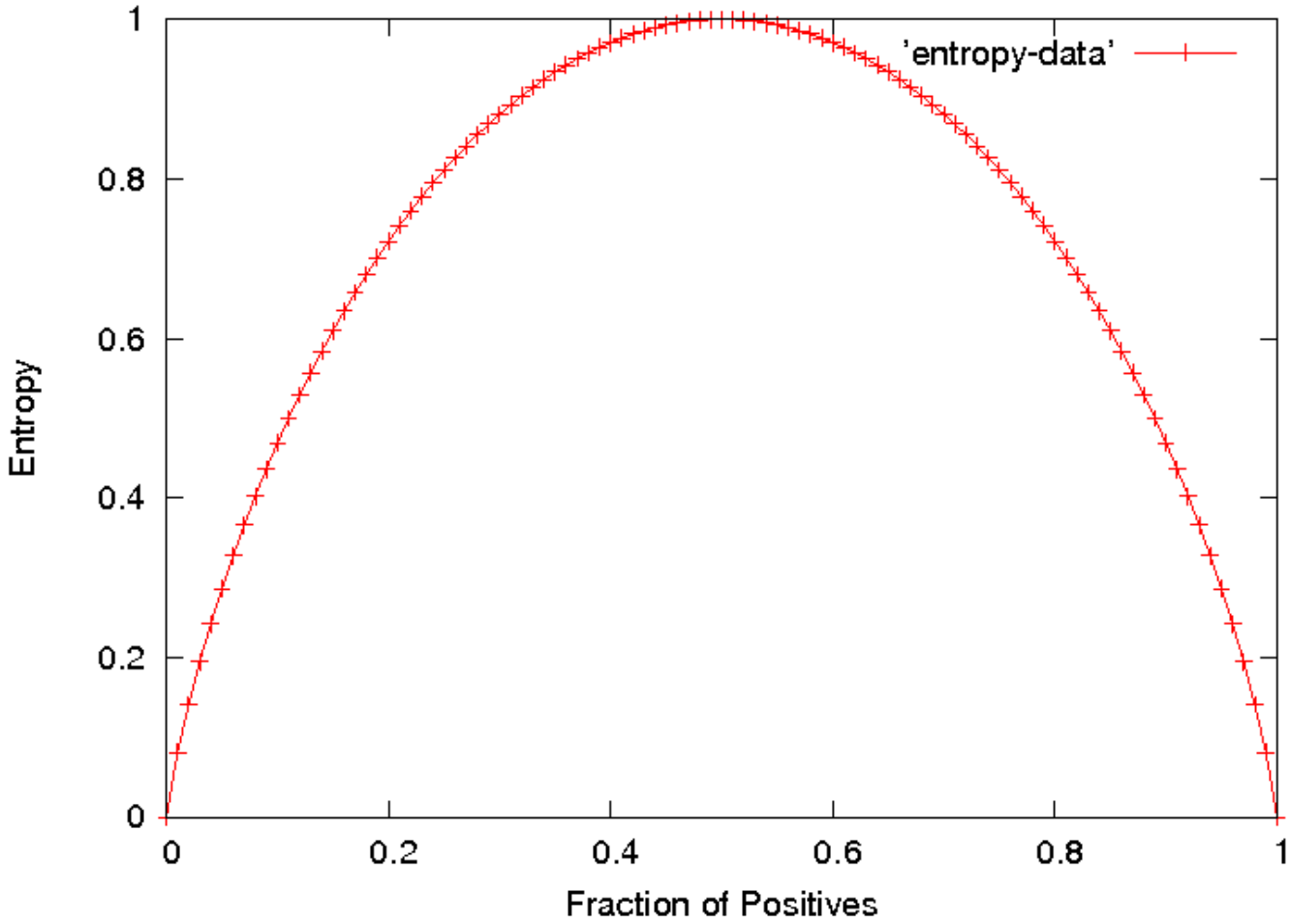
Entropy

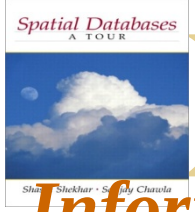
- Entropy (disorder, impurity) of a set of examples, S, relative to a binary classification is:

$$Entropy(S) = -p_1 \log_2(p_1) - p_0 \log_2(p_0)$$

  where p_1 is the fraction of positive examples in S and p_0 is the fraction of negatives.
- If all examples are in one category, entropy is zero (we define $0 \cdot \log(0)=0$)
- If examples are equally mixed ($p_1=p_0=0.5$), entropy is a maximum of 1.
- Entropy can be viewed as the number of bits required on average to encode the class of an example in *S* where data compression (e.g. Huffman coding) is used to give shorter codes to more likely cases.
- For multi-class problems with c categories, entropy generalizes to: $Entropy(S) = \sum_{i=1}^{c} -p_i \log_2(p_i)$

Entropy Plot for Binary Classification





Information Gain

- The information gain of a feature *F* is the expected reduction in entropy resulting from splitting on this feature.

$$Gain(S,F) = Entropy(S) - \sum_{v \in Values(F)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where S_v is the subset of S having value v for feature F.

- Entropy of each resulting subset weighted by its relative size.
- Example:
  - <big, red, circle>: + <small, red, circle>: +
  - <small, red, square>: – <big, blue, circle>: –

2+, 2 –: E=1

size

big: 1+,1– E=1 small: 1+,1– E=1

Gain=1–(0.5·1 + 0.5·1) = 0

2+, 2 – : E=1

color

red: 2+,1– E=0.918 blue: 0+,1– E=0

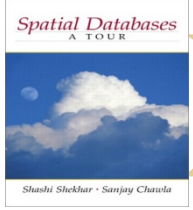
Gain=1–(0.75·0.918 + 0.25·0) = 0.311

2+, 2 – : E=1

shape

circle: 2+,1– E=0.918 square: 0+,1– E=0

Gain=1–(0.75·0.918 + 0.25·0) = 0.311

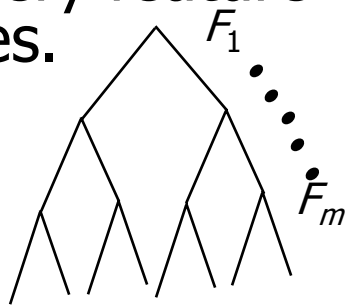


History of Decision-Tree Research

- Hunt and colleagues use exhaustive search decision-tree methods (CLS) to model human concept learning in the 1960's.
- In the late 70's, Quinlan developed ID3 with the information gain heuristic to learn expert systems from examples.
- Simulataneously, Breiman and Friedman and colleagues develop CART (Classification and Regression Trees), similar to ID3.
- In the 1980's a variety of improvements are introduced to handle noise, continuous features, missing features, and improved splitting criteria. Various expert-system development tools results.
- Quinlan's updated decision-tree package (C4.5) released in 1993.
- Weka includes Java version of C4.5 called J48.

Computational Complexity

- Worst case builds a complete tree where every path test every feature. Assume *n* examples and *m* features.



Maximum of *n* examples spread across all nodes at each of the *m* levels

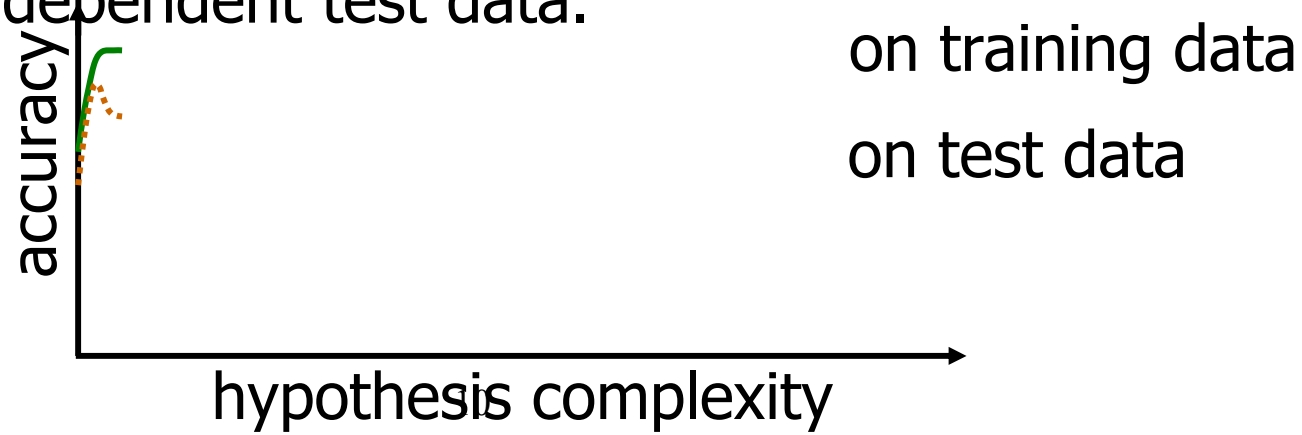
- At each level, *i*, in the tree, must examine the remaining *m*– *i* features for each instance at the level to calculate info gains.

$$\sum_{i=1}^{m} i \cdot n = O(nm^2)$$

- However, learned tree is rarely complete (number of leaves is ≤ *n*). In practice, complexity is linear in both number of features (*m*) and number of training examples (*n*).

Overfitting

- Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization to unseen data.
  - There may be noise in the training data that the tree is erroneously fitting.
  - The algorithm may be making poor decisions towards the leaves of the tree that are based on very little data and may not reflect reliable trends.
- A hypothesis, *h*, is said to overfit the training data is there exists another hypothesis which, *h′*, such that *h* has less error than *h′* on the training data but greater error on independent test data.

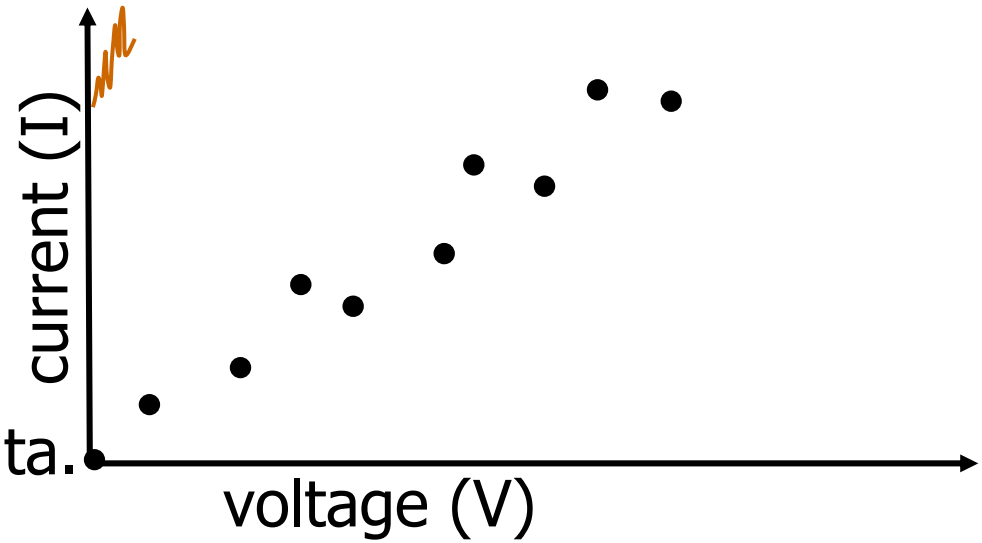


Overfitting Example

Testing Ohms Law: V = IR (I = (1/R)V)

Experimentally measure 10 points

Fit a curve to the resulting data.

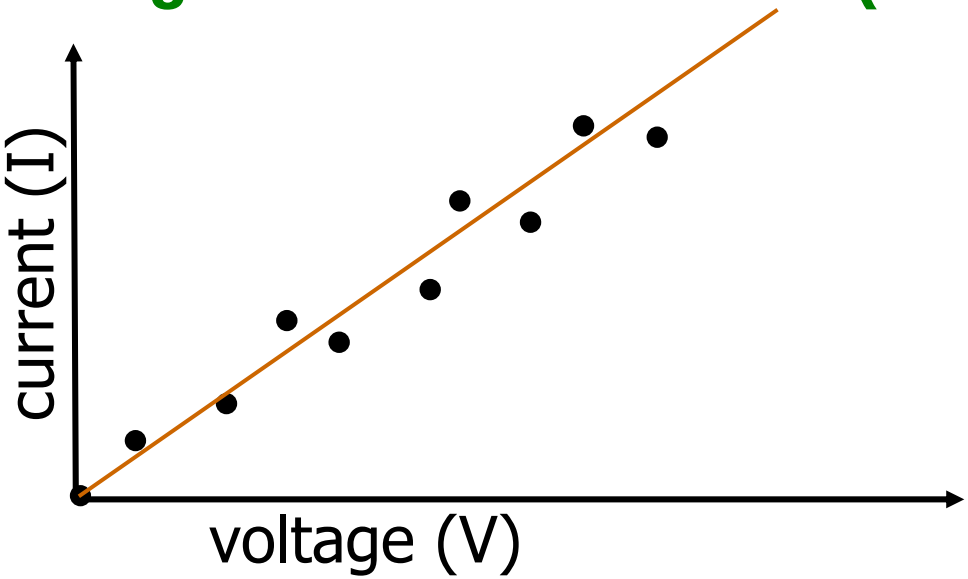


Perfect fit to training data with an 9^{th} degree polynomial
(can fit *n* points exactly with an *n*-1 degree polynomial)

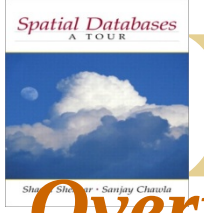
Ohm was wrong, we have found a more accurate function!

Overfitting Example

Testing Ohms Law: V = IR (I = (1/R)V)



Better generalization with a linear function that fits training data less accurately.



Overfitting Prevention (Pruning) Methods

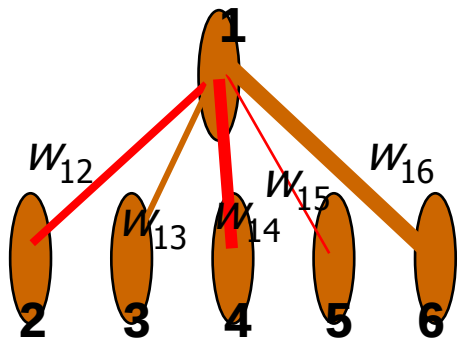
- Two basic approaches for decision trees
  - Prepruning: Stop growing tree as some point during top-down construction when there is no longer sufficient data to make reliable decisions.
  - Postpruning: Grow the full tree, then remove subtrees that do not have sufficient evidence.
- Label leaf resulting from pruning with the majority class of the remaining data, or a class probability distribution.
- Method for determining which subtrees to prune:
  - Cross-validation: Reserve some training data as a hold-out set (***validation set***, ***tuning set***) to evaluate utility of subtrees.
  - Statistical test: Use a statistical test on the training data to determine if any observed regularity can be dismisses as likely due to random chance.
  - Minimum description length (MDL): Determine if the additional complexity of the hypothesis is less complex than just explicitly remembering any exceptions resulting from pruning.

Neural Network Learning

- Learning approach based on modeling adaptation in biological neural systems.
- Perceptron: Initial algorithm for learning simple neural networks (single layer) developed in the 1950's.
- Backpropagation: More complex algorithm for learning multi-layer neural networks developed in the 1980's.

Artificial Neuron Model

- Model network as a graph with cells as nodes and synaptic connections as weighted edges from node *i* to node *j*, w_{ji}



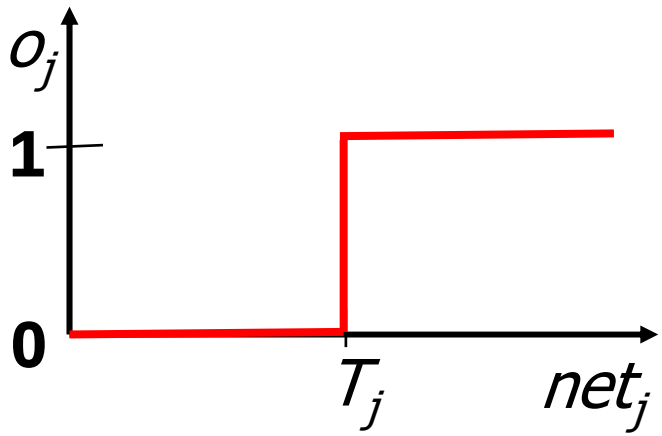
- Model net input to cell as

$$net_j = \sum_i w_{ji} o_i$$

- Cell output is:

$$o_j = \begin{cases} 0 \text{ if } net_j < T_j \\ 1 \text{ if } net_i \geq T_j \end{cases}$$

(T_j is threshold for unit *j*)



Perceptron Learning Algorithm

- Iteratively update weights until convergence.

Initialize weights to random values
Until outputs of all training examples are correct
    For each training pair, *E*, do:
        Compute current output o_j for *E* given its inputs
        Compare current output to target value, t_j, for *E*
        Update synaptic weights and threshold using learning rule

- Each execution of the outer loop is typically called an *epoch*.

Perceptron Learning Rule

- Update weights by:

$$w_{ji} = w_{ji} + \eta(t_j - o_j)o_i$$

where η is the "learning rate"

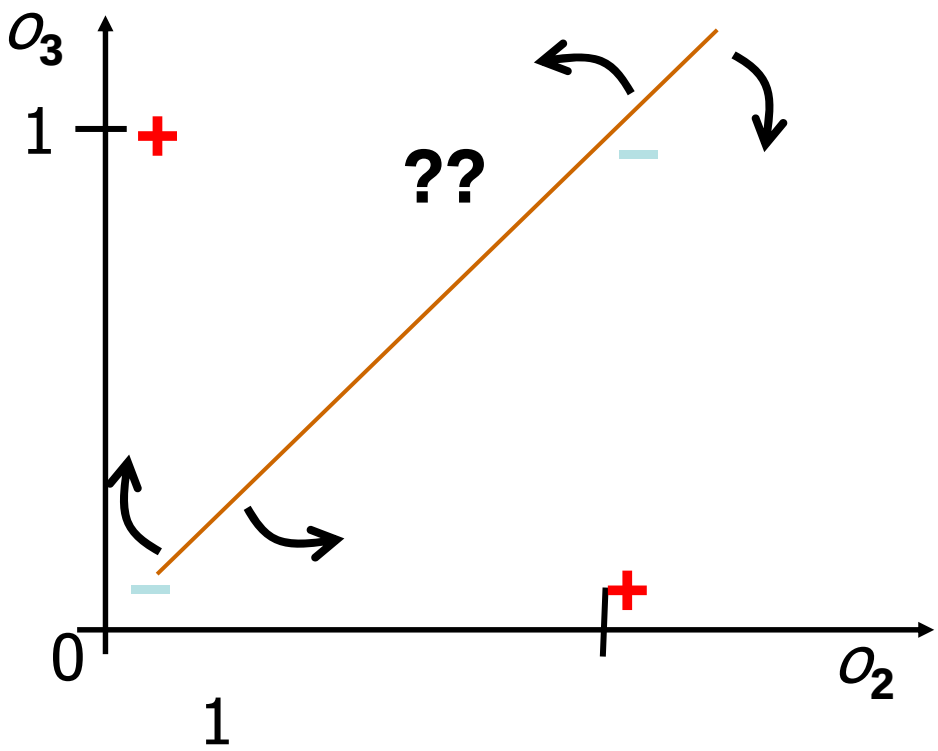
t_j is the teacher specified output for unit j.

- Equivalent to rules:
  - If output is correct do nothing.
  - If output is high, lower weights on active inputs
  - If output is low, increase weights on active inputs
- Also adjust threshold to compensate:

$$T_j = T_j - \eta(t_j - o_j)$$

Concept Perceptron Cannot Learn

- Cannot learn exclusive-or, or parity function in general.

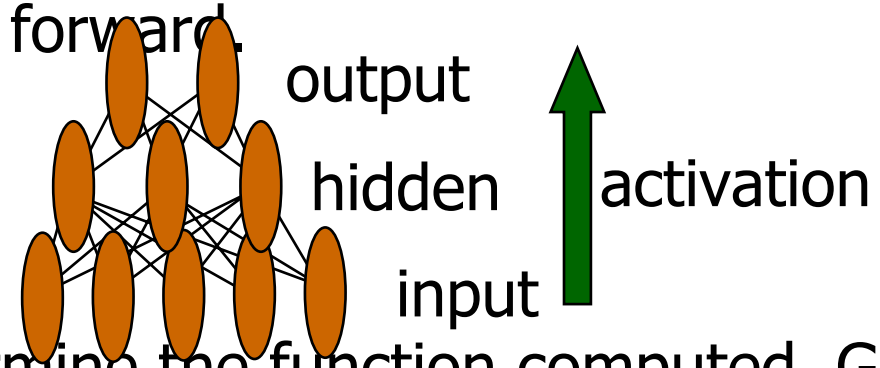


Perceptron Limits

- System obviously cannot learn concepts it cannot represent.
- Minksy and Papert (1969) wrote a book analyzing the perceptron and demonstrating many functions it could not learn.
- These results discouraged further research on neural nets; and symbolic AI became the dominate paradigm.

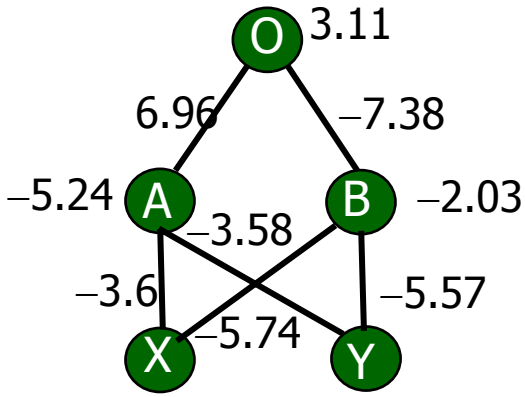
Multi-Layer Networks

- Multi-layer networks can represent arbitrary functions, but an effective learning algorithm for such networks was thought to be difficult.
- A typical multi-layer network consists of an input, hidden and output layer, each fully connected to the next, with activation feeding forward.



- The weights determine the function computed. Given an arbitrary number of hidden units, any boolean function can be computed with a single hidden layer.

Sample Learned XOR Network



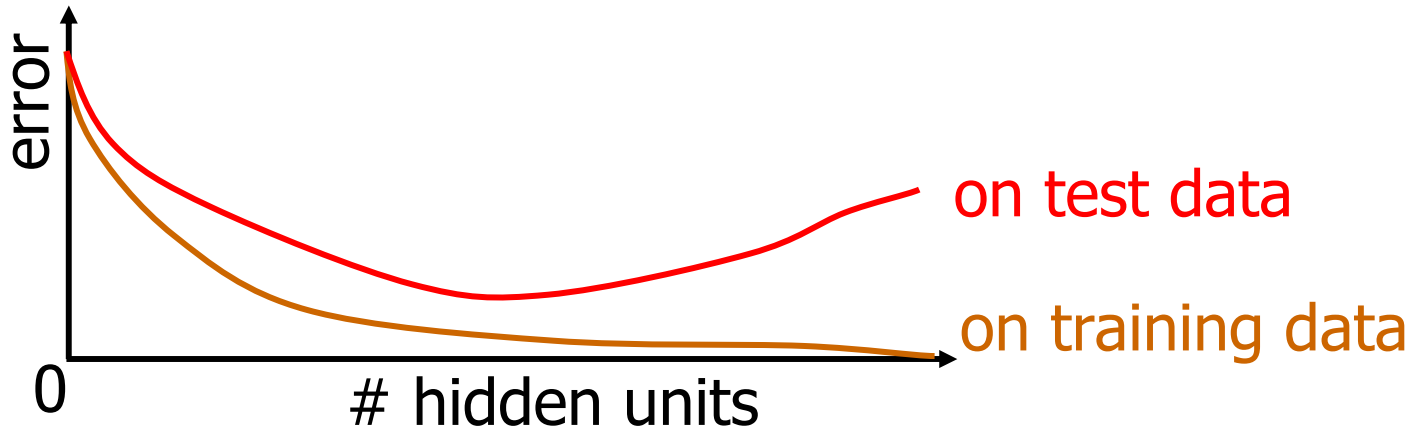
Hidden Unit A represents: $\neg(X \wedge Y)$
Hidden Unit B represents: $\neg(X \vee Y)$
Output O represents: $A \wedge \neg B = \neg(X \wedge Y) \wedge (X \vee Y)$
$= X \oplus Y$

Comments on Training Algorithm

- Not guaranteed to converge to zero training error, may converge to local optima or oscillate indefinitely.
- However, in practice, does converge to low error for many large networks on real data.
- Many epochs (thousands) may be required, hours or days of training for large networks.
- To avoid local-minima problems, run several trials starting with different random weights (*random restarts*).
  - Take results of trial with lowest training set error.
  - Build a committee of results from multiple trials (possibly weighting votes by training set accuracy).

Determining the Best Number of Hidden Units

- Too few hidden units prevents the network from adequately fitting the data.
- Too many hidden units can result in over-fitting.



- Use internal cross-validation to empirically determine an optimal number of hidden units.

Successful Applications

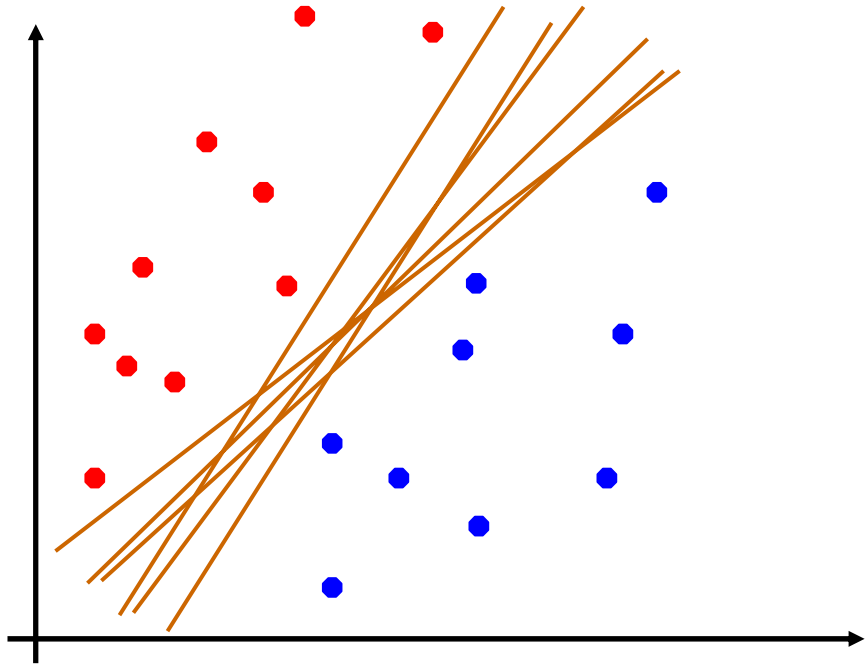
- Text to Speech (NetTalk)
- Fraud detection
- Financial Applications
  - HNC (eventually bought by Fair Isaac)
- Chemical Plant Control
  - Pavillion Technologies
- Automated Vehicles
- Game Playing
  - Neurogammon
- Handwriting recognition

Issues in Neural Nets

- More efficient training methods:
  - Quickprop
  - Conjugate gradient (exploits 2nd derivative)
- Learning the proper network architecture:
  - Grow network until able to fit data
    - Cascade Correlation
    - Upstart
  - Shrink large network until unable to fit data
    - Optimal Brain Damage
- Recurrent networks that use feedback and can learn finite state machines with "backpropagation through time."

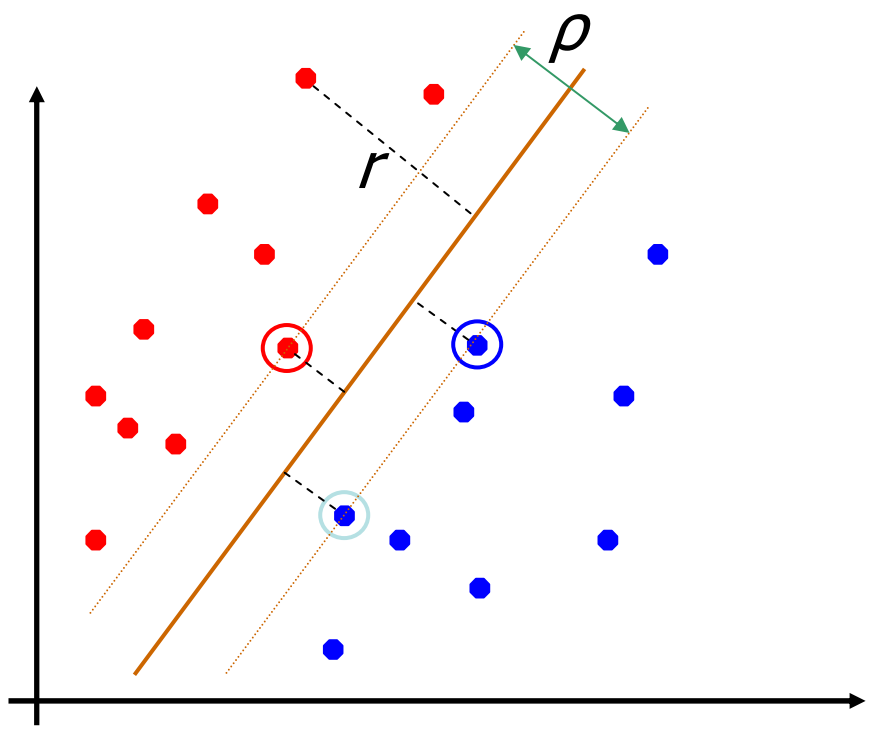
Linear Separators

- Which of the linear separators is optimal?



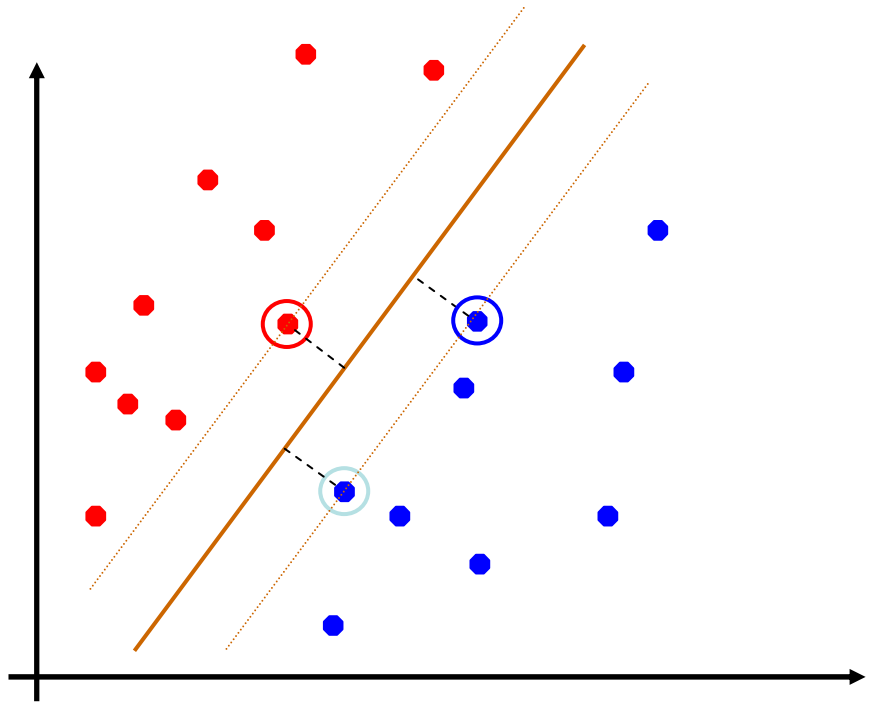
Classification Margin

- Distance from example $\mathbf{x}_i$ to the separator is $r = \frac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|}$
- Examples closest to the hyperplane are ***support vectors***.
- ***Margin*** ρ of the separator is the distance between support vectors.



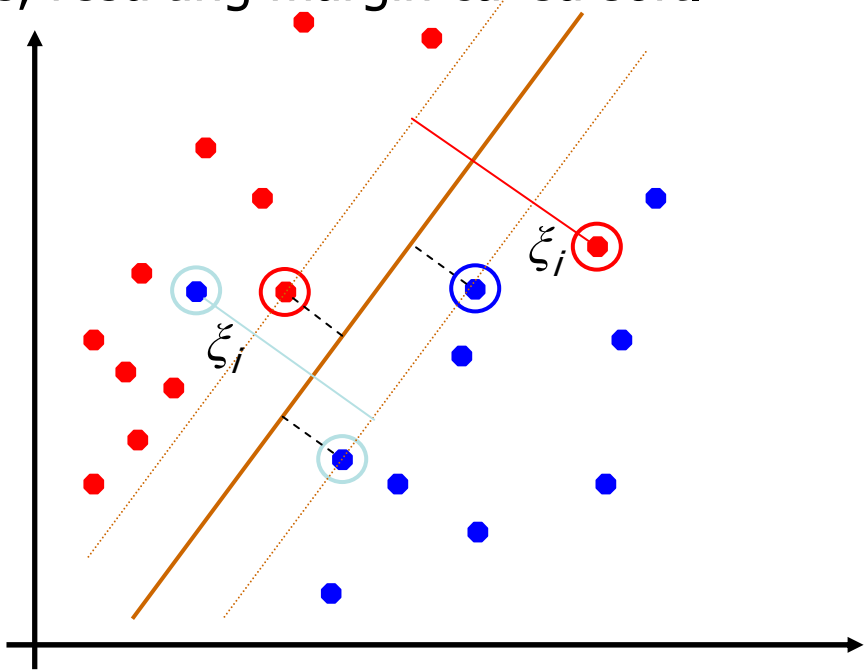
Maximum Margin Classification

- Maximizing the margin is good according to intuition and PAC theory.
- Implies that only support vectors matter; other training examples are ignorable.



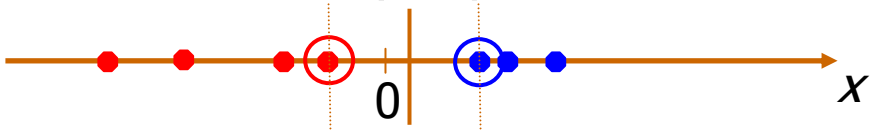
Soft Margin Classification

- What if the training set is not linearly separable?
- *Slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft*.



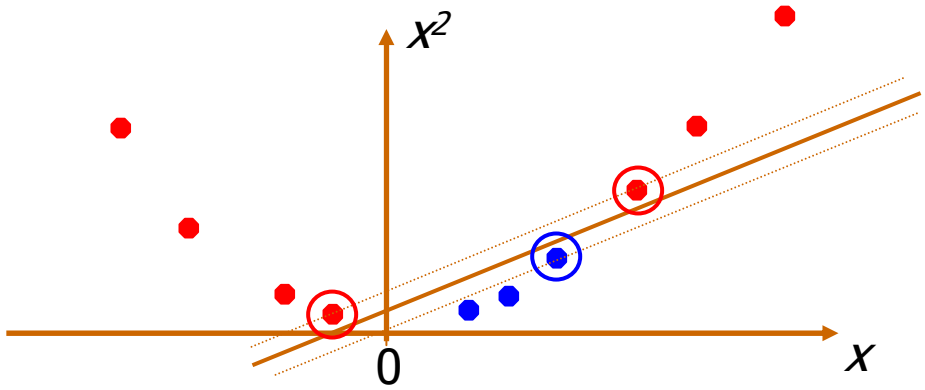
Non-linear SVMs

- Datasets that are linearly separable with some noise work out great:



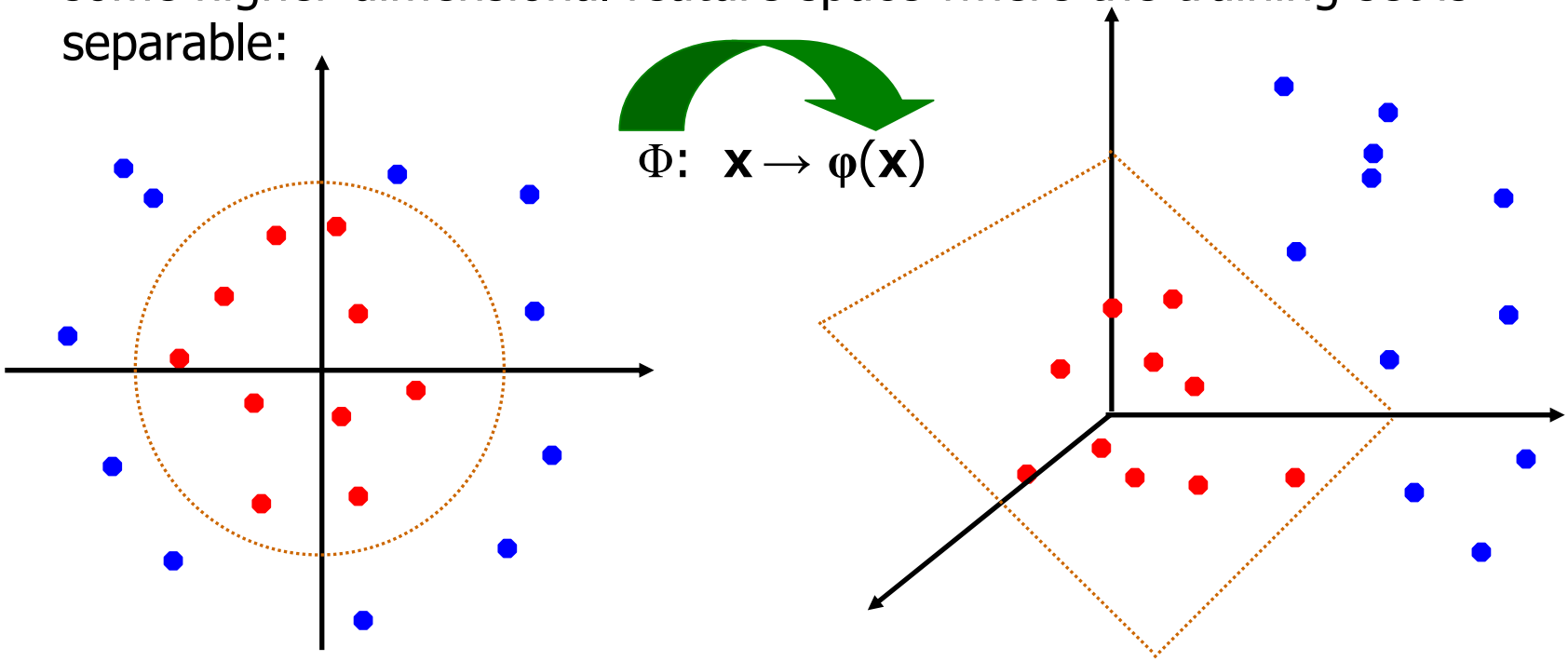
- But what are we going to do if the dataset is just too hard?

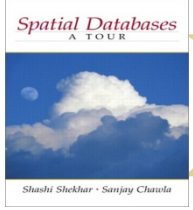
- How about… mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:





SVM applications

- SVMs were originally proposed by Boser, Guyon and Vapnik in 1992 and gained increasing popularity in late 1990s.
- SVMs are currently among the best performers for a number of classification tasks ranging from text to genomic data.
- SVMs can be applied to complex data types beyond feature vectors (e.g. graphs, sequences, relational data) by designing kernel functions for such data.
- SVM techniques have been extended to a number of tasks such as regression [Vapnik *et al.* '97], principal component analysis [Schölkopf *et al.* '99], etc.
- Most popular optimization algorithms for SVMs use *decomposition* to hill-climb over a subset of α_iś at a time, e.g. SMO [Platt '99] and [Joachims '99]
- Tuning SVMs remains a black art: selecting a specific kernel and parameters is usually done in a try-and-see manner.

Instance-Based Learning

- Unlike other learning algorithms, does not involve construction of an explicit abstract generalization but classifies new instances based on direct comparison and similarity to known training instances.
- Training can be very easy, just memorizing training instances.
- Testing can be very expensive, requiring detailed comparison to all past training instances.
- Also known as:
  - Case-based
  - Exemplar-based
  - Nearest Neighbor
  - Memory-based
  - Lazy Learning

Similarity/Distance Metrics

- Instance-based methods assume a function for determining the similarity or distance between any two instances.
- For continuous feature vectors, Euclidian distance is the generic choice:

$$d(x_i, x_j) = \sqrt{\sum_{p=1}^{n} (a_p(x_i) - a_p(x_j))^2}$$

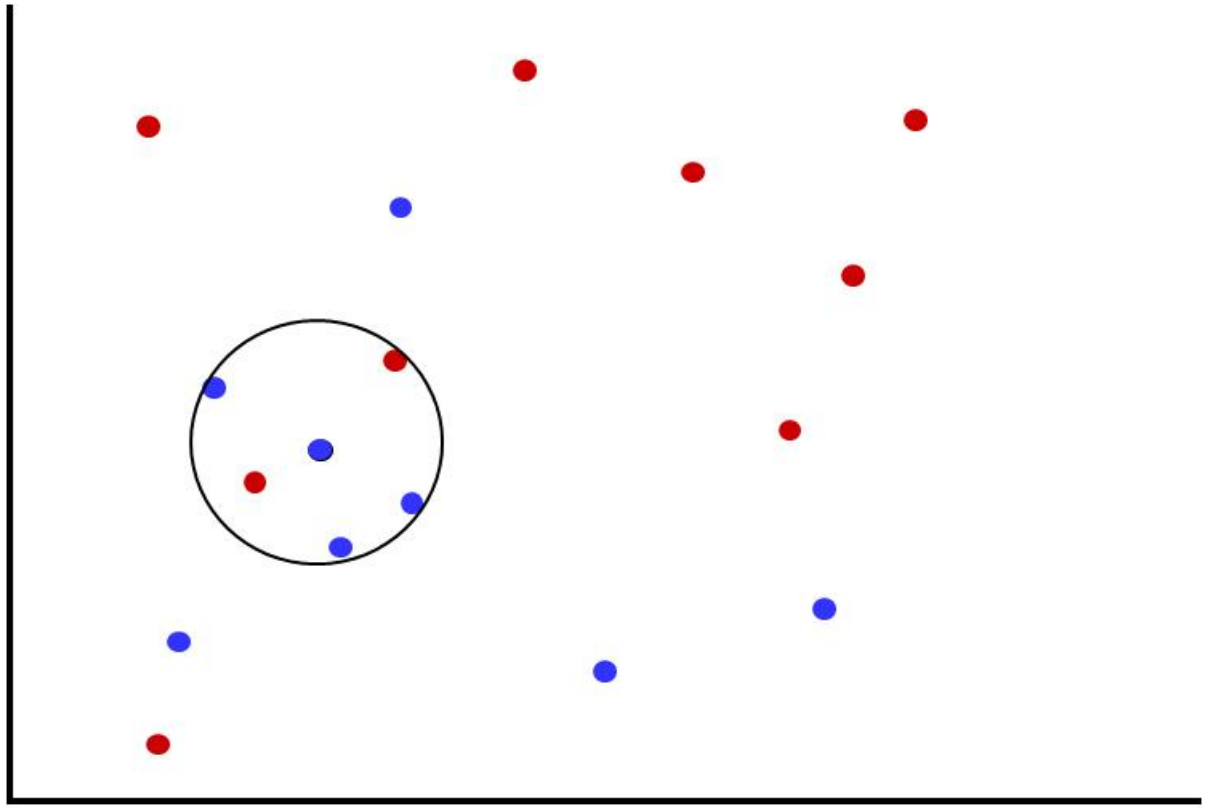
Where $a_p(x)$ is the value of the pth feature of instance x.

- For discrete features, assume distance between two values is 0 if they are the same and 1 if they are different (e.g. Hamming distance for bit vectors).
- To compensate for difference in units across features, scale all continuous values to the interval [0,1].

Other Distance Metrics

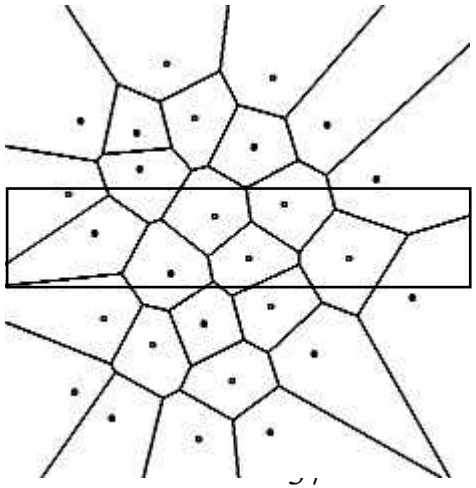
- Mahalanobis distance
  - Scale-invariant metric that normalizes for variance.
- Cosine Similarity
  - Cosine of the angle between the two vectors.
  - Used in text and other high-dimensional data.
- Pearson correlation
  - Standard statistical correlation coefficient.
  - Used for bioinformatics data.
- Edit distance
  - Used to measure distance between unbounded length strings.
  - Used in text and bioinformatics.

k-Nearest Neighbor Example



Implicit Classification Function

- Although it is not necessary to explicitly calculate it, the learned classification rule is based on regions of the feature space closest to each training example.
- For 1-nearest neighbor with Euclidian distance, the **Voronoi diagram** gives the complex polyhedra segmenting the space into the regions closest to each point.

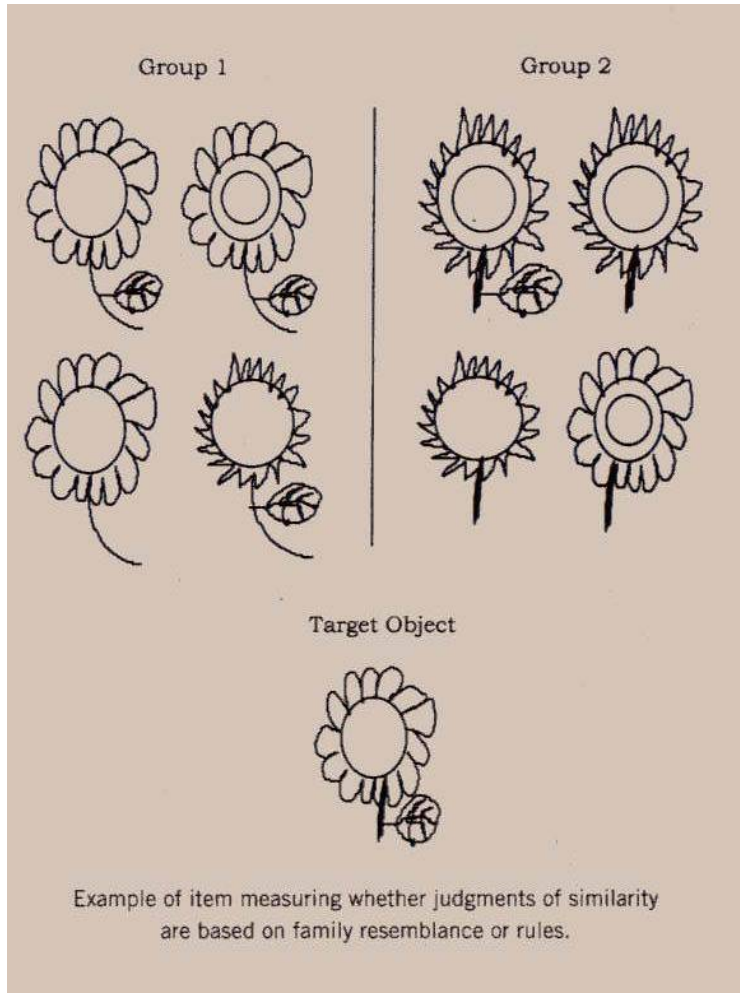


Efficient Indexing

- Linear search to find the nearest neighbors is not efficient for large training sets.
- Indexing structures can be built to speed testing.
- For Euclidian distance, a **kd-tree** can be built that reduces the expected time to find the nearest neighbor to O(log *n*) in the number of training examples.
  - Nodes branch on threshold tests on individual features and leaves terminate at nearest neighbors.
- Other indexing structures possible for other metrics or string data.
  - Inverted index for text retrieval.

Rules and Instances in Human Learning Biases

- Psychological experiments show that people from different cultures exhibit distinct categorization biases.
- "Western" subjects favor simple rules (straight stem) and classify the target object in group 2.
- "Asian" subjects favor global similarity and classify the target object in group 1.



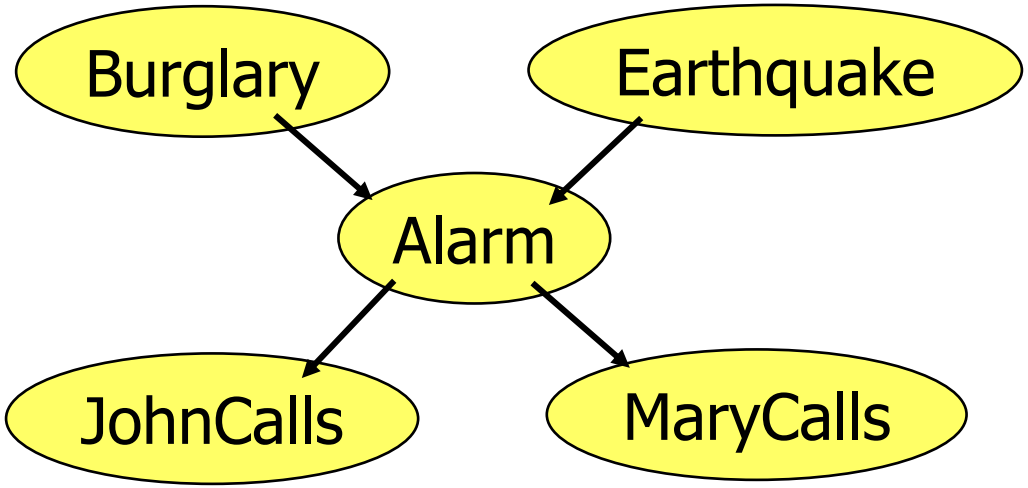
Example of item measuring whether judgments of similarity are based on family resemblance or rules.

Other Issues

- Can reduce storage of training instances to a small set of representative examples.
  - Support vectors in an SVM are somewhat analogous.
- Can be used for more complex relational or graph data.
  - Similarity computation is complex since it involves some sort of graph isomorphism.
- Can be used in problems other than classification.
  - Case-based planning
  - Case-based reasoning in law and business.

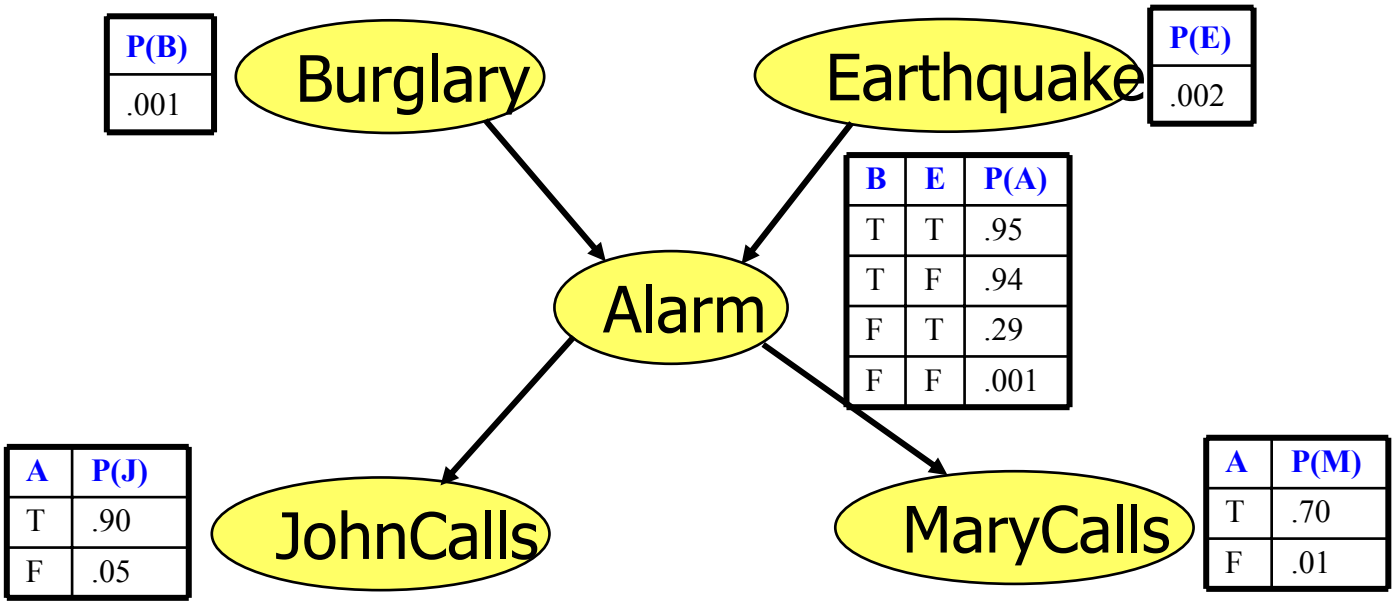
Bayesian Networks

- Directed Acyclic Graph (DAG)
  - Nodes are random variables
  - Edges indicate causal influences



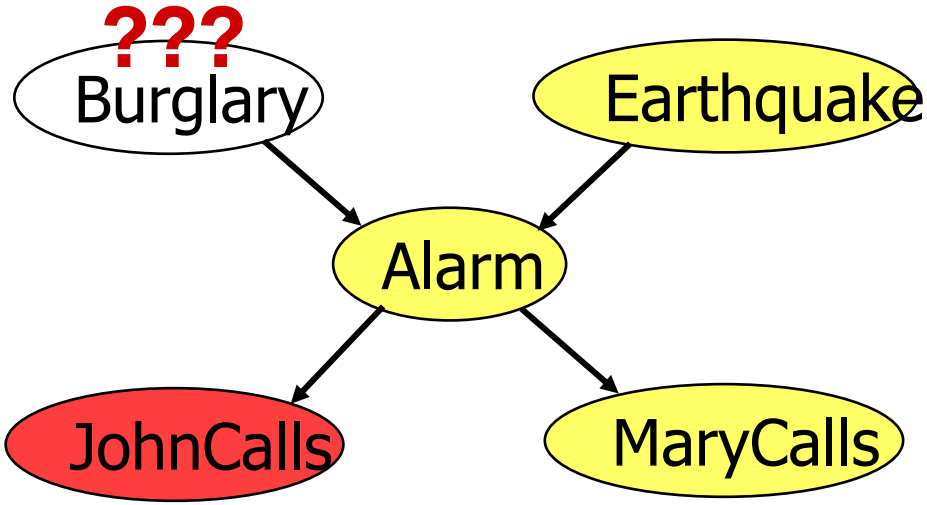
Conditional Probability Tables

- Each node has a **conditional probability table** (**CPT**) that gives the probability of each of its values given every possible combination of values for its parents (conditioning case).
  - Roots (sources) of the DAG that have no parents are given prior probabilities.



Bayes Net Inference

- Given known values for some **evidence variables**, determine the posterior probability of some **query variables**.
- Example: Given that John calls, what is the probability that there is a Burglary?

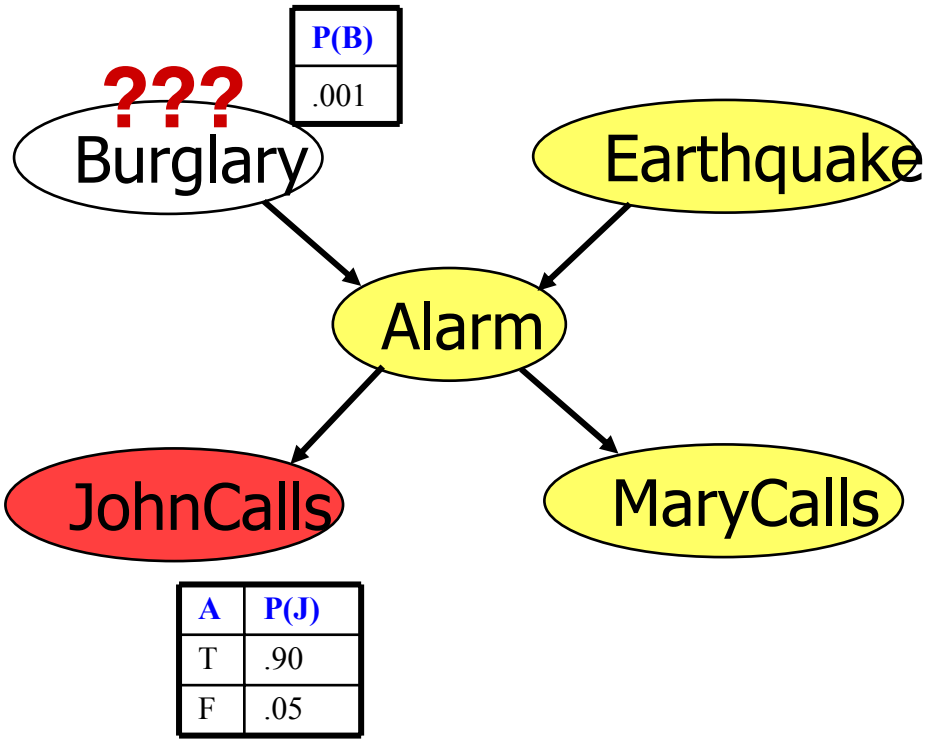


John calls 90% of the time there is an Alarm and the Alarm detects 94% of Burglaries so people generally think it should be fairly high.

However, this ignores the prior probability of John calling.

Bayes Net Inference

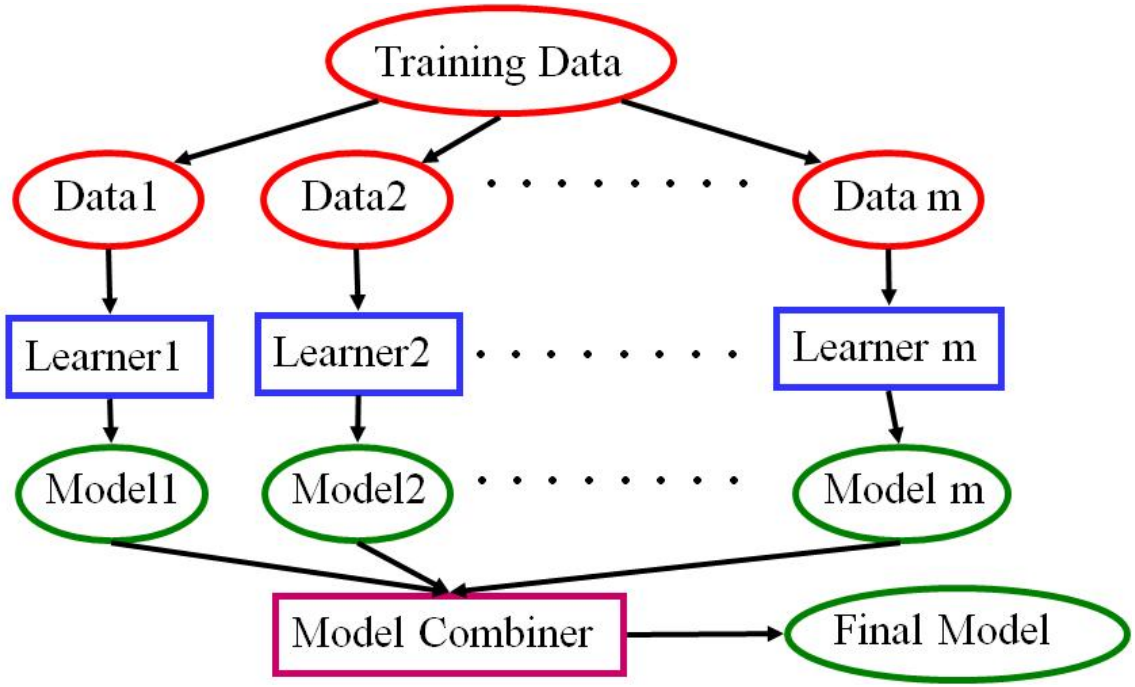
- Example: Given that John calls, what is the probability that there is a Burglary?



John also calls 5% of the time when there
is no Alarm. So over 1,000 days we expect 1 Burglary and John will probably call. However, he will also call with a false report 50 times on average. So the call is about 50 times more likely a false report:
P(Burglary | JohnCalls) ≈ 0.02

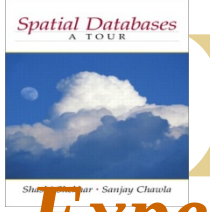
Learning Ensembles

- Learn multiple alternative definitions of a concept using different training data or different learning algorithms.
- Combine decisions of multiple definitions, e.g. using weighted voting.



Value of Ensembles

- When combing multiple ***independent*** and ***diverse*** decisions each of which is at least more accurate than random guessing, random errors cancel each other out, correct decisions are reinforced.
- Human ensembles are demonstrably better
  - How many jelly beans in the jar?: Individual estimates vs. group average.
  - Who Wants to be a Millionaire: Expert friend vs. audience vote.



Experimental Results on Ensembles (Freund & Schapire, 1996; Quinlan, 1996)

- Ensembles have been used to improve generalization accuracy on a wide variety of problems.
- On average, Boosting provides a larger increase in accuracy than Bagging.
- Boosting on rare occasions can degrade accuracy.
- Bagging more consistently provides a modest improvement.
- Boosting is particularly subject to over-fitting when there is significant noise in the training data.

K-Fold Cross Validation Comments

- Every example gets used as a test example once and as a training example k–1 times.
- All test sets are independent; however, training sets overlap significantly.
- Measures accuracy of hypothesis generated for $[(k–1)/k]\cdot|D|$ training examples.
- Standard method is 10-fold.
- If k is low, not sufficient number of train/test trials; if k is high, test set is small and test variance is high and run time is increased.
- If k=|D|, method is called ***leave-one-out*** cross validation.