CSE 592
Applications of Artificial Intelligence

Neural Networks & Data Mining

Henry Kautz
Winter 2003

Neural Networks

Preview

- Perceptrons
- Gradient descent
- Multilayer networks
- Backpropagation

Connectionist Models

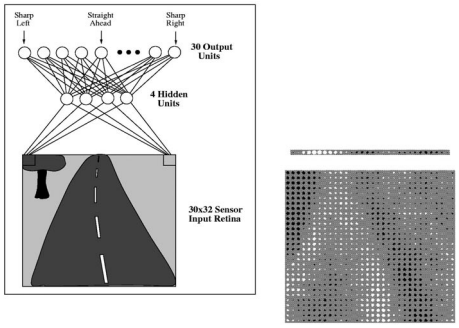
Consider humans:

- Neuron switching time $\sim$ .001 second
- Number of neurons $\sim 10^{10}$
- Connections per neuron $\sim 10^{4-5}$
- Scene recognition time $\sim$ .1 second
- 100 inference steps doesn't seem like enough

$\Rightarrow$ Much parallel computation

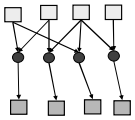
Properties of neural nets:

- Many neuron-like threshold switching units
- Many weighted interconnections among units
- Highly parallel, distributed process
- Emphasis on tuning weights automatically



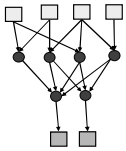
Kinds of Networks

- Feed-forward
- Single layer
- Multi-layer
- Recurrent



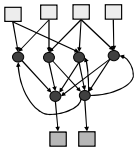
Kinds of Networks

- Feed-forward
- Single layer
- Multi-layer
- Recurrent

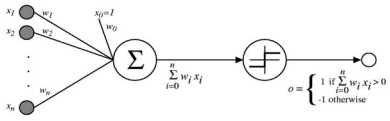


Kinds of Networks

- Feed-forward
- Single layer
- Multi-layer
- Recurrent



Perceptron

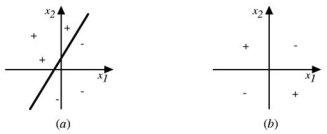


$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Sometimes we'll use simpler vector notation:

$$o(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Decision Surface of a Perceptron



Represents some useful functions

- What weights represent $g(x_1, x_2) = AND(x_1, x_2)$?

But some functions not representable

- All not linearly separable
- Therefore, we'll want networks of these...

Perceptron Training Rule

$$w_i \leftarrow w_i + \Delta w_i$$

where

$$\Delta w_i = \eta (t - o) x_i$$

Where:

- $t = c(\vec{x})$ is target value
- o is perceptron output
- η is small constant (e.g., 0.1) called *learning rate*

Basic Idea:

Use error between target and actual output to adjust weights

Perceptron Training Rule

Can prove it will converge if

- Training data is linearly separable
- η sufficiently small

Gradient Descent

To understand, consider simpler *linear unit*, where

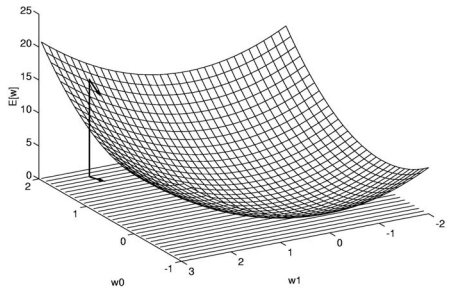
$$o = w_0 + w_1 x_1 + \cdots + w_n x_n$$

Let's learn w_i's that minimize the squared error

$$E[\vec{w}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

Where D is set of training examples

Gradient Descent



Gradient Descent

Gradient:

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n} \right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

I.e.:

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

In other words: take a step the steepest downhill direction

Gradient Descent

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_d (t_d - o_d)^2 \\
&= \frac{1}{2} \sum_d \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\
&= \frac{1}{2} \sum_d 2 (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\
&= \sum_d (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \vec{w} \cdot \vec{x_d}) \\
\frac{\partial E}{\partial w_i} &= \sum_d (t_d - o_d)(-x_{i,d})
\end{aligned}$$

Multiply by η and you get the training rule!

Gradient Descent

GRADIENT-DESCENT($training\_examples, \eta$)

Initialize each w_i to some small random value

Until the termination condition is met, Do

- Initialize each Δw_i to zero.
- For each $\langle \vec{x}, t \rangle$ in $training\_examples$, Do
  - Input instance $\vec{x}$ to unit and compute output o
  - For each linear unit weight w_i, Do

$$\Delta w_i \leftarrow \Delta w_i + \eta(t - o)x_i$$

- For each linear unit weight w_i, Do

$$w_i \leftarrow w_i + \Delta w_i$$

Summary

Perceptron training rule guaranteed to succeed if

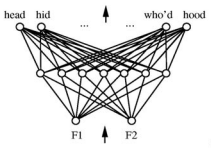
- Training examples are linearly separable
- Sufficiently small learning rate η

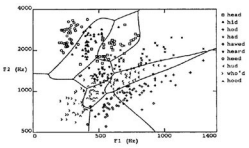
Linear unit training rule uses gradient descent

- Guaranteed to converge to hypothesis with minimum squared error
- Given sufficiently small learning rate η
- Even when training data contains noise
- Even when training data not separable by H

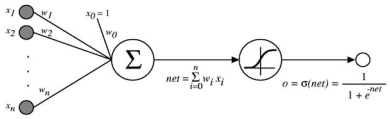
Demos

Multilayer Networks of Sigmoid Units





Sigmoid Unit



$\sigma(x)$ is the sigmoid function

$$\frac{1}{1 + e^{-x}}$$

Nice property: $\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$

Training Rule

- Single sigmoid unit (a "soft" perceptron)

  $\Delta w_i = \eta \delta x_i$

  where the error term $\delta = o(1-o)(t-o)$

  Deriviative of the sigmoid gives this part

- Multi-Layered network
  - Compute Δ values for output units, using observed outputs
  - For each layer from output back:
    - Propagate the Δ values back to previous layer
    - Update incoming weights

Backpropagation Algorithm

Initialize all weights to small random numbers
Until convergence, Do
  For each training example, Do
    1. Input it to network and compute network outputs
    2. For each output unit k
$$\delta_k \leftarrow o_k(1-o_k)(t_k-o_k)$$
    3. For each hidden unit h
$$\delta_h \leftarrow o_h(1-o_h)\sum_{k\in outputs} w_{h,k}\delta_k$$
    4. Update each network weight $w_{i,j}$
$$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$$
       where $\Delta w_{i,j} = \eta \delta_j x_{i,j}$

Backpropagation Algorithm

Initialize all weights to small random numbers
Until convergence, Do
  For each training example, Do
    1. Input it to network and compute network outputs
    2. For each output unit k
$$\delta_k \leftarrow o_k(1-o_k)(t_k-o_k)$$
    3. For each hidden unit h
$$\delta_h \leftarrow o_h(1-o_h)\sum_{k\in outputs} w_{h,k}\delta_k$$
    4. Update each network weight $w_{i,j}$
$$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$$
       where $\Delta w_{i,j} = \eta \delta_j x_{i,j}$

Derivative of output

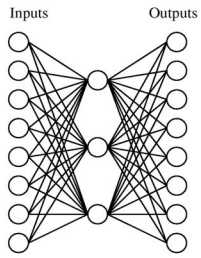
Weighted error

More on Backpropagation

- Gradient descent over entire *network* weight vector
- Easily generalized to arbitrary directed graphs
- Will find a local, not necessarily global error minimum
  - In practice, often works well (can run multiple times)

- Minimizes error over *training* examples
  - Will it generalize well to subsequent examples?
- Training can take thousands of iterations → slow!
- Using network after training is very fast

Learning Hidden Layer Representations



A target function:

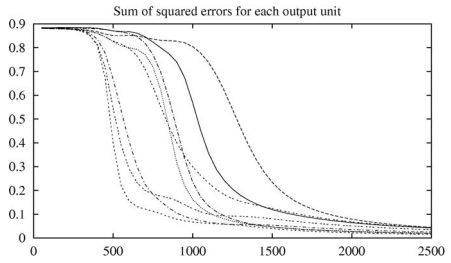
Input		Output
10000000	→	10000000
01000000	→	01000000
00100000	→	00100000
00010000	→	00010000
00001000	→	00001000
00000100	→	00000100
00000010	→	00000010
00000001	→	00000001

Can this be learned?

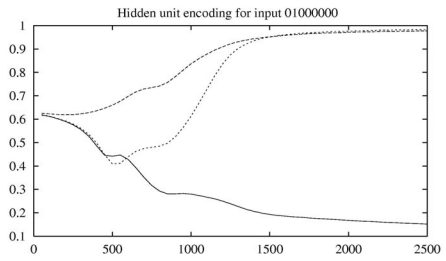
Learned hidden layer representation:

Input		Hidden Values				Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.01	.11	.88	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.22	.99	.99	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

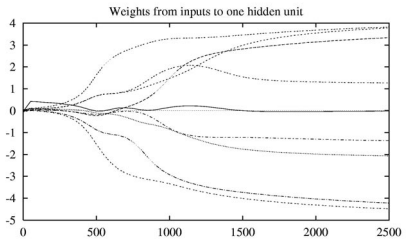
Training



Training



Training



Convergence of Backpropagation

Gradient descent to some local minimum

- Perhaps not global minimum...
- Add momentum
- Stochastic gradient descent
- Train multiple nets with different inital weights

Nature of convergence

- Initialize weights near zero
- Therefore, initial networks near-linear
- Increasingly non-linear functions possible as training progresses

Expressiveness of Neural Nets

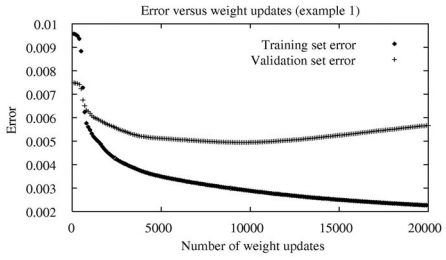
Boolean functions:

- Every Boolean function can be represented by network with single hidden layer
- But might require exponential (in number of inputs) hidden units

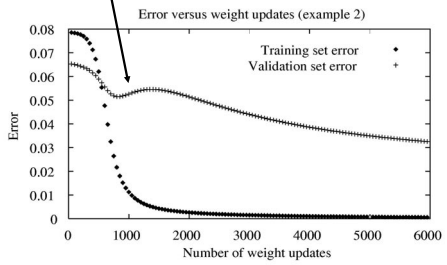
Continuous functions:

- Every bounded continuous function can be approximated with arbitrarily small error, by network with one hidden layer
- Any function can be approximated to arbitrary accuracy by a network with two hidden layers

Overfitting in Neural Nets



Be careful not to stop too soon!



Overfitting Avoidance

Penalize large weights:

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2 + \gamma \sum_{i,j} w_{ji}^2$$

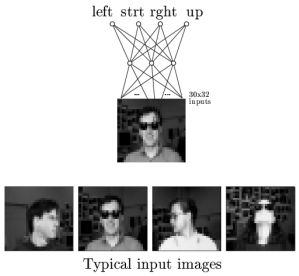
Train on target slopes as well as values:

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} \left[(t_{kd} - o_{kd})^2 + \mu \sum_{j \in inputs} \left(\frac{\partial t_{kd}}{\partial x_d^j} - \frac{\partial o_{kd}}{\partial x_d^j} \right)^2 \right]$$

Weight sharing

Early stopping

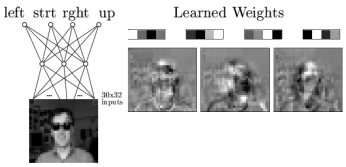
Neural Nets for Face Recognition



Typical input images

90% accurate learning head pose, and recognizing 1-of-20 faces

Learned Hidden Unit Weights



Typical input images

Neural Networks: Summary

- Perceptrons
- Gradient descent
- Multilayer networks
- Backpropagation

Break!

Data Mining

Data Mining

- What is the difference between machine learning and data mining?

Data Mining

- What is the difference between machine learning and data mining?
  - Scale – DM is ML in the large
  - Focus – DM is more interested in finding "interesting" patterns than in learning to classify data

Data Mining

- What is the difference between machine learning and data mining?
  - Scale – DM is ML in the large
  - Focus – DM is more interested in finding "interesting" patterns than in learning to classify data
  - Marketing!

Data Mining:
Association Rules

Mining Association Rules in Large Databases

- Introduction to association rule mining
- Mining single-dimensional Boolean association rules from transactional databases
- Mining multilevel association rules from transactional databases
- Mining multidimensional association rules from transactional databases and data warehouse
- Constraint-based association mining
- Summary

What Is Association Rule Mining?

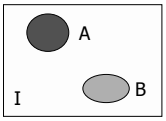
- Association rule mining:
  - Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.
- Applications:
  - Basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering, classification, etc.
- Examples:
  - Rule form: "Body → Head [support, confidence]".
  - buys(x, "diapers") → buys(x, "beers") [0.5%, 60%]
  - major(x, "CS") ^ takes(x, "DB") → grade(x, "A") [1%, 75%]

Association Rules: Basic Concepts

- Given: (1) database of transactions, (2) each transaction is a list of items (purchased by a customer in a visit)
- Find: all rules that correlate the presence of one set of items with that of another set of items
  - E.g., *98% of people who purchase tires and auto accessories also get automotive services done*
- Applications
  - *? ⇒ Maintenance Agreement* (What the store should do to boost Maintenance Agreement sales)
  - *Home Electronics ⇒ ?* (What other products should the store stocks up?)
  - Attached mailing in direct marketing

Association Rules: Definitions

- Set of *items*: $I = \{i_1, i_2, \ldots, i_m\}$
- Set of *transactions*: $D = \{d_1, d_2, \ldots, d_n\}$
  Each $d_i \subset I$
- An *association rule*: $A \Rightarrow B$
  where $A \subset I$, $B \subset I$, $A \cap B = \varnothing$



- Means that to some extent A implies B.
- Need to measure how strong the implication is.

Association Rules: Definitions II

- The probability of a set A:

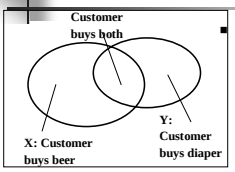
$$P(A) = \frac{\sum_i C(A, d_i)}{|D|} \quad \text{Where:} \quad C(X,Y) = \begin{cases} 1 \text{ if } X \subseteq Y \\ 0 \text{ else} \end{cases}$$

- *k-itemset*: tuple of items, or sets of items:
- Example: {A,B} is a 2-itemset
- The probability of {A,B} is the probability of the set $A \cup B$, that is the fraction of transactions that contain both A and B. Not the same as $P(A \cap B)$.

Association Rules: Definitions III

- *Support* of a rule $A \Rightarrow B$ is the probability of the itemset {A,B}. This gives an idea of how often the rule is relevant.
  - $support(A \Rightarrow B) = P(\{A,B\})$
- *Confidence* of a rule $A \Rightarrow B$ is the conditional probability of B given A. This gives a measure of how accurate the rule is.
  - $confidence(A \Rightarrow B) = P(B|A)$
    $= support(\{A,B\}) / support(A)$

Rule Measures: Support and Confidence



Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

- Find all the rules $X \Rightarrow Y$ given thresholds for minimum confidence and minimum support.
  - support, *s*, probability that a transaction contains {X, Y}
  - confidence, *c*, conditional probability that a transaction having X also contains *Y*

With *minimum support 50%, and minimum confidence 50%, we have*

- $A \Rightarrow C$ (50%, 66.6%)
- $C \Rightarrow A$ (50%, 100%)

Association Rule Mining: A Road Map

- Boolean vs. quantitative associations (Based on the types of values handled)
  - buys(x, "SQLServer") ^ buys(x, "DMBook") → buys(x, "DBMiner") [0.2%, 60%]
  - age(x, "30..39") ^ income(x, "42..48K") → buys(x, "PC") [1%, 75%]
- Single dimension vs. multiple dimensional associations (see ex. Above)
- Single level vs. multiple-level analysis
  - What brands of beers are associated with what brands of diapers?
- Various extensions and analysis
  - Correlation, causality analysis
    - Association does not necessarily imply correlation or causality
  - Maxpatterns and closed itemsets
  - Constraints enforced
    - E.g., small sales (sum < 100) trigger big buys (sum > 1,000)?

Mining Association Rules in Large Databases

- Association rule mining
- Mining single-dimensional Boolean association rules from transactional databases
- Mining multilevel association rules from transactional databases
- Mining multidimensional association rules from transactional databases and data warehouse
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Mining Association Rules—An Example

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Min. support 50%
Min. confidence 50%

Frequent Itemset	Support
{A}	75%
{B}	50%
{C}	50%
{A,C}	50%

For rule $A \Rightarrow C$:

support = support($\{A, C\}$) = 50%

confidence = support($\{A, C\}$)/support($\{A\}$) = 66.6%

The Apriori principle:

Any subset of a frequent itemset must be frequent

Mining Frequent Itemsets: the Key Step

- Find the *frequent itemsets*: the sets of items that have at least a given minimum support
  - A subset of a frequent itemset must also be a frequent itemset
    - i.e., if $\{A, B\}$ is a frequent itemset, both $\{A\}$ and $\{B\}$ should be a frequent itemset
  - Iteratively find frequent itemsets with cardinality from 1 to *k* (*k*-itemset)
- Use the frequent itemsets to generate association rules.

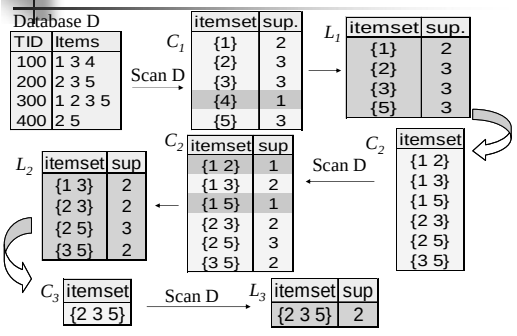
The Apriori Algorithm

- Join Step: C_k is generated by joining $L_{k-1}$with itself
- Prune Step: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset
- Pseudo-code:

C_k: Candidate itemset of size k
L_k: frequent itemset of size k

L_1 = {frequent items}
for (k = 1; L_k !=∅; k++) **do begin**
  C_{k+1} = candidates generated from L_k
  for each transaction t in database do
    increment the count of all candidates in C_{k+1} that are contained in t
  L_{k+1} = candidates in C_{k+1} with min_support
  end
return $\cup_k L_k$;

The Apriori Algorithm — Example



How to do Generate Candidates?

- Suppose the items in L_{k-1} are listed in an order
- Step 1: self-joining L_{k-1}

  insert into C_k

  select $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$

  from L_{k-1} p, L_{k-1} q

  where $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
- Step 2: pruning

  forall ***itemsets c in*** C_k do

  forall ***(k-1)-subsets s of c*** do

  if *(s is not in* L_{k-1}*)* **then delete** *c* **from** C_k

Example of Generating Candidates

- L_3={*abc, abd, acd, ace, bcd*}
- Self-joining: L_3*L_3
  - *abcd* from *abc* and *abd*
  - *acde* from *acd* and *ace*
- Pruning:
  - *acde* is removed because *ade* is not in L_3
- C_4={*abcd*}

Methods to Improve Apriori's Efficiency

- Hash-based itemset counting: A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
- Transaction reduction: A transaction that does not contain any frequent k-itemset is useless in subsequent scans
- Partitioning: Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
- Sampling: mining on a subset of given data, lower support threshold + a method to determine the completeness
- Dynamic itemset counting: add new candidate itemsets only when all of their subsets are estimated to be frequent

Is Apriori Fast Enough? — Performance Bottlenecks

- The core of the Apriori algorithm:
  - Use frequent ($k-1$)-itemsets to generate candidate frequent *k*-itemsets
  - Use database scan and pattern matching to collect counts for the candidate itemsets
- The bottleneck of *Apriori*: candidate generation
  - Huge candidate sets:
    - 10^4 frequent 1-itemset will generate 10^7 candidate 2-itemsets
    - To discover a frequent pattern of size 100, e.g., {a_1, a_2, …, a_{100}}, one needs to generate $2^{100} \approx 10^{30}$ candidates.
  - Multiple scans of database:
    - Needs ($n+1$) scans, n is the length of the longest pattern

Mining Frequent Patterns Without Candidate Generation

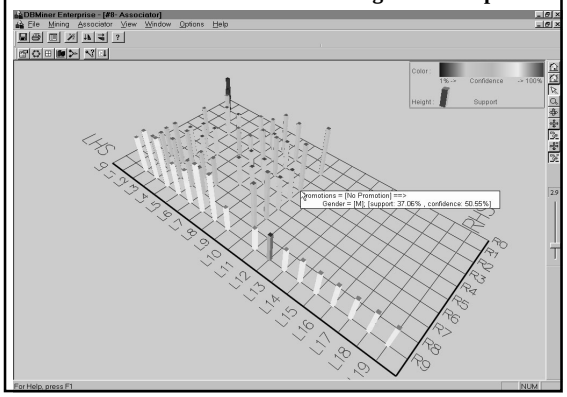
- Compress a large database into a compact, Frequent-Pattern tree (FP-tree) structure
  - highly condensed, but complete for frequent pattern mining
  - avoid costly database scans
- Develop an efficient, FP-tree-based frequent pattern mining method
  - A divide-and-conquer methodology: decompose mining tasks into smaller ones
  - Avoid candidate generation: sub-database test only!

Presentation of Association Rules (Table Form)

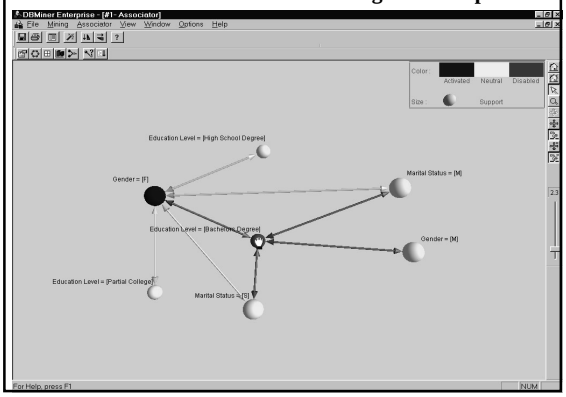
	Body	Implies	Head	Supp (%)	Conf (%)	F	G	H	I
1	cost(x) = '0.00~1000.00'	==>	revenue(x) = '0.00~500.00'	28.45	40.4				
2	cost(x) = '0.00~1000.00'	==>	revenue(x) = '500.00~1000.00'	28.46	39.95				
3	cost(x) = '0.00~1000.00'	==>	order_qty(x) = '0.00~100.00'	59.17	84.04				
4	cost(x) = '0.00~1000.00'	==>	revenue(x) = '1000.00~1500.00'	10.45	14.84				
5	cost(x) = '0.00~1000.00'	==>	region(x) = 'United States'	22.56	32.04				
6	cost(x) = '1000.00~2000.00'	==>	order_qty(x) = '0.00~100.00'	12.91	69.34				
7	order_qty(x) = '0.00~100.00'	==>	revenue(x) = '0.00~500.00'	28.45	34.54				
8	order_qty(x) = '0.00~100.00'	==>	cost(x) = '1000.00~2000.00'	12.91	15.67				
9	order_qty(x) = '0.00~100.00'	==>	region(x) = 'United States'	25.9	31.45				
10	order_qty(x) = '0.00~100.00'	==>	cost(x) = '0.00~1000.00'	59.17	71.86				
11	order_qty(x) = '0.00~100.00'	==>	product_line(x) = 'Tents'	13.52	16.42				
12	order_qty(x) = '0.00~100.00'	==>	revenue(x) = '500.00~1000.00'	19.67	23.86				
13	product_line(x) = 'Tents'	==>	order_qty(x) = '0.00~100.00'	13.52	98.72				
14	region(x) = 'United States'	==>	order_qty(x) = '0.00~100.00'	25.9	81.94				
15	region(x) = 'United States'	==>	cost(x) = '0.00~1000.00'	22.56	71.39				
16	revenue(x) = '0.00~500.00'	==>	cost(x) = '0.00~1000.00'	28.45	100				
17	revenue(x) = '0.00~500.00'	==>	order_qty(x) = '0.00~100.00'	28.45	100				
18	revenue(x) = '1000.00~1500.00'	==>	cost(x) = '0.00~1000.00'	10.45	95.75				
19	revenue(x) = '500.00~1000.00'	==>	cost(x) = '0.00~1000.00'	20.46	100				
20	revenue(x) = '500.00~1000.00'	==>	order_qty(x) = '0.00~100.00'	19.67	96.14				
21									
22									
23	cost(x) = '0.00~1000.00'	==>	revenue(x) = '0.00~500.00' AND order_qty(x) = '0.00~100.00'	28.45	40.4				
24	cost(x) = '0.00~1000.00'	==>	revenue(x) = '0.00~500.00' AND order_qty(x) = '0.00~100.00'	28.45	40.4				
25	cost(x) = '0.00~1000.00'	==>	revenue(x) = '500.00~1000.00' AND order_qty(x) = '0.00~100.00'	19.67	27.93				
26	cost(x) = '0.00~1000.00'	==>	revenue(x) = '500.00~1000.00' AND order_qty(x) = '0.00~100.00'	19.67	27.93				
27	cost(x) = '0.00~1000.00' AND order_qty(x) = '0.00~100.00'	==>	revenue(x) = '500.00~1000.00'	19.67	33.23				

Sheet1

Visualization of Association Rule Using Plane Graph



Visualization of Association Rule Using Rule Graph

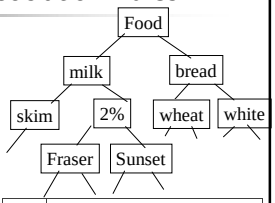


Mining Association Rules in Large Databases

- Association rule mining
- Mining single-dimensional Boolean association rules from transactional databases
- Mining multilevel association rules from transactional databases
- Mining multidimensional association rules from transactional databases and data warehouse
- From association mining to correlation analysis
- Constraint-based association mining
- Summary

Multiple-Level Association Rules

- Items often form hierarchy.
- Items at the lower level are expected to have lower support.
- Rules regarding itemsets at appropriate levels could be quite useful.
- Transaction database can be encoded based on dimensions and levels
- We can explore shared multi-level mining



TID	Items
T1	{111, 121, 211, 221}
T2	{111, 211, 222, 323}
T3	{112, 122, 221, 411}
T4	{111, 121}
T5	{111, 122, 211, 221, 413}

Mining Multi-Level Associations

- A top_down, progressive deepening approach:
  - First find high-level strong rules:
    milk → bread [20%, 60%].
  - Then find their lower-level "weaker" rules:
    2% milk → wheat bread [6%, 50%].
- Variations at mining multiple-level association rules.
  - Level-crossed association rules:
    2% *milk* → *Wonder wheat bread*
  - Association rules with multiple, alternative hierarchies:
    2% *milk* → *Wonder bread*

Mining Association Rules in Large Databases

- Association rule mining
- Mining single-dimensional Boolean association rules from transactional databases
- Mining multilevel association rules from transactional databases
- Mining multidimensional association rules from transactional databases and data warehouse
- Constraint-based association mining
- Summary

Multi-Dimensional Association: Concepts

- Single-dimensional rules:
  buys(X, "milk") ⇒ buys(X, "bread")
- Multi-dimensional rules: ❍ 2 dimensions or predicates
  - Inter-dimension association rules (*no repeated predicates*)
    age(X,"19-25") ∧ occupation(X,"student") ⇒ buys(X,"coke")
  - hybrid-dimension association rules (*repeated predicates*)
    age(X,"19-25") ∧ buys(X, "popcorn") ⇒ buys(X, "coke")
- Categorical Attributes
  - finite number of possible values, no ordering among values
- Quantitative Attributes
  - numeric, implicit ordering among values

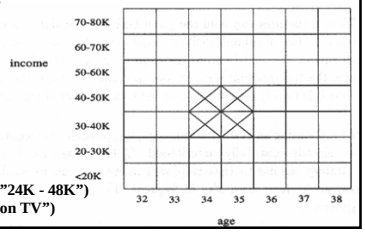
Techniques for Mining MD Associations

- Search for frequent *k*-predicate set:
  - Example: {age, occupation, buys} is a 3-predicate set.
  - Techniques can be categorized by how age are treated.

1. Using static discretization of quantitative attributes
   - Quantitative attributes are statically discretized by using predefined concept hierarchies.
2. Quantitative association rules
   - Quantitative attributes are dynamically discretized into "bins" based on the distribution of the data.

Quantitative Association Rules

- Numeric attributes are *dynamically* discretized
  - Such that the confidence or compactness of the rules mined is maximized.
- 2-D quantitative association rules: $A_{quan1} \wedge A_{quan2} \Rightarrow A_{cat}$
- Cluster "adjacent" association rules to form general rules using a 2-D grid.
- Example:



age(X,"30-34") ∧ income(X,"24K - 48K") ⇒ buys(X,"high resolution TV")

Mining Association Rules in Large Databases

- Association rule mining
- Mining single-dimensional Boolean association rules from transactional databases
- Mining multilevel association rules from transactional databases
- Mining multidimensional association rules from transactional databases and data warehouse
- Constraint-based association mining
- Summary

Mining Association Rules in Large Databases

- Association rule mining
- Mining single-dimensional Boolean association rules from transactional databases
- Mining multilevel association rules from transactional databases
- Mining multidimensional association rules from transactional databases and data warehouse
- Constraint-based association mining
- Summary

Constraint-Based Mining

- Interactive, exploratory mining giga-bytes of data?
  - Could it be real? — Making good use of constraints!
- What kinds of constraints can be used in mining?
  - Knowledge type constraint: classification, association, etc.
  - Data constraint: SQL-like queries
    - Find product pairs sold together in Vancouver in Dec.'98.
  - Dimension/level constraints:
    - in relevance to region, price, brand, customer category.
  - Rule constraints
    - small sales (price < $10) triggers big sales (sum > $200).
  - Interestingness constraints:
    - strong rules (min_support ≥ 3%, min_confidence ≥ 60%).

Rule Constraints in Association Mining

- Two kind of rule constraints:
  - Rule form constraints: meta-rule guided mining.
    - P(x, y) ^ Q(x, w) → takes(x, "database systems").
  - Rule (content) constraint: constraint-based query optimization (Ng, et al., SIGMOD'98).
    - sum(LHS) < 100 ^ min(LHS) > 20 ^ count(LHS) > 3 ^ sum(RHS) > 1000
- 1-variable vs. 2-variable constraints (Lakshmanan, et al. SIGMOD'99):
  - 1-var: A constraint confining only one side (L/R) of the rule, e.g., as shown above.
  - 2-var: A constraint confining both sides (L and R).
    - sum(LHS) < min(RHS) ^ max(RHS) < 5* sum(LHS)

Constrained Association Query Optimization Problem

- Given a CAQ = { (S_1, S_2) / C }, the algorithm should be :
  - sound: It only finds frequent sets that satisfy the given constraints C
  - complete: All frequent sets satisfy the given constraints C are found
- A naïve solution:
  - Apply Apriori for finding all frequent sets, and then to test them for constraint satisfaction one by one.
- More advanced approach:
  - Comprehensive analysis of the properties of constraints and try to push them as deeply as possible inside the frequent set computation.

Summary

- Association rules offer an efficient way to mine interesting probabilities about data in very large databases.
- Can be dangerous when misinterpreted as signs of statistically significant causality.
- The basic Apriori algorithm and it's extensions allow the user to gather a good deal of information without too many passes through data.

Data Mining:
Clustering

Preview

- Introduction
- Partitioning methods
- Hierarchical methods
- Model-based methods
- Density-based methods

What is Clustering?

- Cluster: a collection of data objects
  - Similar to one another within the same cluster
  - Dissimilar to the objects in other clusters
- Cluster analysis
  - Grouping a set of data objects into clusters
- Clustering is unsupervised classification: no predefined classes
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

Examples of Clustering Applications

- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- Land use: Identification of areas of similar land use in an earth observation database
- Insurance: Identifying groups of motor insurance policy holders with a high average claim cost
- Urban planning: Identifying groups of houses according to their house type, value, and geographical location
- Seismology: Observed earth quake epicenters should be clustered along continent faults

What Is a Good Clustering?

- A good clustering method will produce clusters with
  - High intra-class similarity
  - Low inter-class similarity
- Precise definition of clustering quality is difficult
  - Application-dependent
  - Ultimately subjective

Requirements for Clustering in Data Mining

- Scalability
- Ability to deal with different types of attributes
- Discovery of clusters with arbitrary shape
- Minimal domain knowledge required to determine input parameters
- Ability to deal with noise and outliers
- Insensitivity to order of input records
- Robustness wrt high dimensionality
- Incorporation of user-specified constraints
- Interpretability and usability

Similarity and Dissimilarity Between Objects

- Properties of a metric $d(i,j)$:
  - $d(i,j) \geq 0$
  - $d(i,i) = 0$
  - $d(i,j) = d(j,i)$
  - $d(i,j) \leq d(i,k) + d(k,j)$

Major Clustering Approaches

- Partitioning: Construct various partitions and then evaluate them by some criterion
- Hierarchical: Create a hierarchical decomposition of the set of objects using some criterion
- Model-based: Hypothesize a model for each cluster and find best fit of models to data
- Density-based: Guided by connectivity and density functions

Partitioning Algorithms

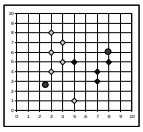
- Partitioning method: Construct a partition of a database ***D*** of ***n*** objects into a set of ***k*** clusters
- Given a *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: *k-means* and *k-medoids* algorithms
  - *k-means* (MacQueen, 1967): Each cluster is represented by the center of the cluster
  - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw, 1987): Each cluster is represented by one of the objects in the cluster

K-Means Clustering

- Given *k*, the *k-means* algorithm consists of four steps:
  - Select initial centroids at random.
  - Assign each object to the cluster with the nearest centroid.
  - Compute each centroid as the mean of the objects assigned to it.
  - Repeat previous 2 steps until no change.

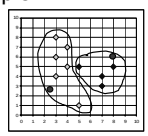
K-Means Clustering (contd.)

- Example



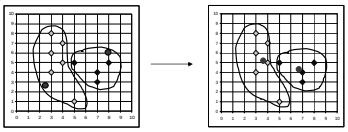
K-Means Clustering (contd.)

- Example



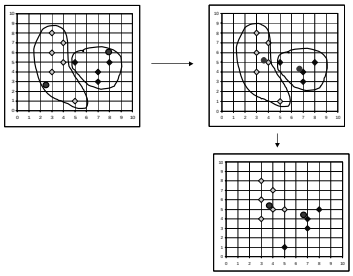
K-Means Clustering (contd.)

- Example



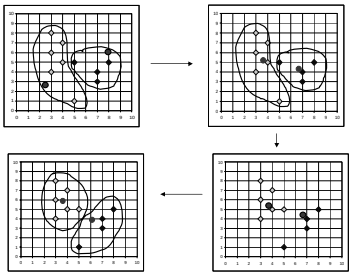
K-Means Clustering (contd.)

- Example



K-Means Clustering (contd.)

- Example

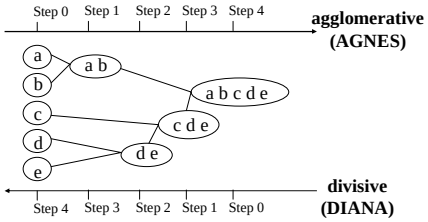


Comments on the *K-Means* Method

- Strengths
  - *Relatively efficient*: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, $k, t << n$.
  - Often terminates at a *local optimum*. The *global optimum* may be found using techniques such as *simulated annealing* and *genetic algorithms*
- Weaknesses
  - Applicable only when *mean* is defined (what about categorical data?)
  - Need to specify *k,* the *number* of clusters, in advance
  - Trouble with noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

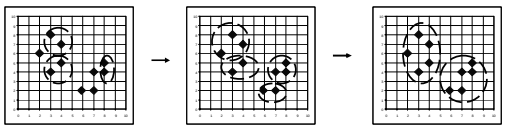
Hierarchical Clustering

- Use distance matrix as clustering criteria. This method does not require the number of clusters ***k*** as an input, but needs a termination condition



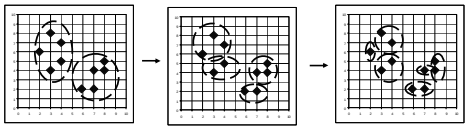
AGNES (Agglomerative Nesting)

- Produces tree of clusters (nodes)
- Initially: each object is a cluster (leaf)
- Recursively merges nodes that have the least dissimilarity
- Criteria: min distance, max distance, avg distance, center distance
- Eventually all nodes belong to the same cluster (root)



DIANA (Divisive Analysis)

- Inverse order of AGNES
- Start with root cluster containing all objects
- Recursively divide into subclusters
- Eventually each cluster contains a single object



Other Hierarchical Clustering Methods

- Major weakness of agglomerative clustering methods
  - Do not scale well: time complexity of at least $O(n^2)$, where n is the number of total objects
  - Can never undo what was done previously
- Integration of hierarchical with distance-based clustering
  - BIRCH: uses CF-tree and incrementally adjusts the quality of sub-clusters
  - CURE: selects well-scattered points from the cluster and then shrinks them towards the center of the cluster by a specified fraction

Model-Based Clustering

- Basic idea: Clustering as probability estimation
- One model for each cluster
- *Generative* model:
  - Probability of selecting a cluster
  - Probability of generating an object in cluster
- Find max. likelihood or MAP model
- Missing information: Cluster membership
  - Use EM algorithm
- Quality of clustering: Likelihood of test objects

http://ic.arc.nasa.gov/ic/projects/bayes-group/autoclass/

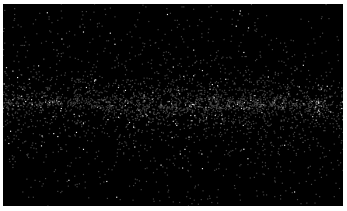
AutoClass

An unsupervised Bayesian classification system that seeks a maximum posterior probability classification.

Key features:

- determines the number of classes automatically;
- can use mixed discrete and real valued data;
- can handle missing values – uses EM (Expectation Maximization)
- processing time is roughly linear in the amount of the data;
- cases have probabilistic class membership;
- allows correlation between attributes within a class;
- generates reports describing the classes found; and
- predicts "test" case class memberships from a "training" classification

From subtle differences between their infrared spectra, two subgroups of stars were distinguished, where previously no difference was suspected.



The difference is confirmed by looking at their positions on this map of the galaxy.

Clustering: Summary

- Introduction
- Partitioning methods
- Hierarchical methods
- Model-based methods

- Next week: Making Decisions
  - From utility theory to reinforcement learning
- Finish assignments!
- Start (or keep rolling on project) –
  - Today's status report in my mail ASAP (next week at the latest)