Review: Association Learning

- Algorithms for association learning:
  - take a set of training examples
  - discover associations among attributes
    - example: products that people tend to purchase together
  - Unlike classification learning, association learning doesn't single out a single attribute for special treatment.
  - the resulting model may determine the value of any of the attributes – or even of combinations of attributes
Association Rules

- The learned associations are usually expressed as rules known as association rules. Examples:
  
  if PurchaseDiapers = Yes  
  then PurchaseBeer = Yes  

  if PurchaseMilk = Yes and PurchaseJuice = Yes  
  then PurchaseEggs = Yes and PurchaseCheese = Yes  

- The test or tests in the if clause of a rule is known as the precondition of the rule.

- The assignment in the then clause of a rule is known as the conclusion of the rule.

- General format:
  
  if <precondition>  
  then <conclusion>

The Converse of a Rule

- The converse of a rule is obtained by swapping the precondition and conclusion.

  - example: here’s one rule:
    
    if PurchaseDiapers = Yes  
    then PurchaseBeer = Yes  

    its converse is:
    
    if PurchaseBeer = Yes  
    then PurchaseDiapers = Yes  

  - The converse of a rule is not necessarily true.

  - example: this rule is true:
    
    if name = 'Perry Sullivan'  
    then yearBorn = 2000  

    its converse is not:
    
    if yearBorn = 2000  
    then name = 'Perry Sullivan'
Example Problem: Credit-Card Promotions

- We'll use these training examples, which omit the Age attribute:

<table>
<thead>
<tr>
<th>Sex</th>
<th>Income Range</th>
<th>Credit Card Insurance</th>
<th>Life Insurance Promotion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Female</td>
<td>30–40K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Male</td>
<td>30–40K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>50–60K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>20–30K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Male</td>
<td>30–40K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>20–30K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Male</td>
<td>30–40K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Female</td>
<td>30–40K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>20–30K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>50–60K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Female</td>
<td>20–30K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Possible association rules include:
  
  if Sex = Male and IncomeRange = 40-50K then CreditCardIns = No and LifeIns = No
  
  if CreditCardIns = Yes and LifeIns = Yes then Sex = Male

Metric #1: Support

- The support of a rule is the number of training examples containing the attribute values found in both the rule's precondition and its conclusion.
  
  i.e., the number of examples that the rule gets right
  
  these examples "support" the rule

- This metric can also be expressed as a percentage of the total number of training examples.
Metric #1: Support (cont.)

if Sex = Male and IncomeRange = 40-50K then CreditCardIns = No and LifeIns = No
• support = 3 instances (or 20% of the total training set)

if CreditCardIns = Yes and LifeIns = Yes then Sex = Male
• support = 2 instances (or 13.3% of the total training set)

<table>
<thead>
<tr>
<th>Sex</th>
<th>Income Range</th>
<th>Credit Card Insurance</th>
<th>Life Insurance</th>
<th>Promotion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Female</td>
<td>30–40K</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Male</td>
<td>30–40K</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Female</td>
<td>50–60K</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Female</td>
<td>20–30K</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Male</td>
<td>30–40K</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Male</td>
<td>20–30K</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Female</td>
<td>30–40K</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Female</td>
<td>40–50K</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Male</td>
<td>20–30K</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Female</td>
<td>50–60K</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Female</td>
<td>20–30K</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

Metric #2: Confidence

• The confidence of a rule provides a measure of a rule's accuracy – of how well it predicts the values in the conclusion.

• It answers the question: if the precondition of the rule holds, how likely is it that the conclusion also holds?

• Here's the formula:

\[
\text{confidence} = \frac{\# \text{ examples with the values in the precondition and the conclusion}}{\# \text{ examples with the values in just the precondition}} \rightarrow \text{the support}
\]
Metric #2: Confidence (cont)

if \( \text{Sex} = \text{Male} \) and \( \text{IncomeRange} = 40-50K \)
then \( \text{CreditCardIns} = \text{No} \) and \( \text{LifeIns} = \text{No} \)

- confidence = \( \frac{\# \text{ examples with all four values}}{\# \text{ examples with Sex=Male and IncomeRange=40-50K}} \)
  \[ = \frac{3}{3} \]
  \[ = 1 \text{ or } 100\% \] (perfect accuracy on training examples)

<table>
<thead>
<tr>
<th>Sex</th>
<th>Income Range</th>
<th>Credit Card Insurance</th>
<th>Life Insurance Promotion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Female</td>
<td>30–40K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Male</td>
<td>30–40K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>50–60K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>20–30K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Male</td>
<td>30–40K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>20–30K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Female</td>
<td>30–40K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>40–50K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>20–30K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>50–60K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Female</td>
<td>20–30K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

if \( \text{CreditCardIns} = \text{Yes} \) and \( \text{LifeIns} = \text{Yes} \)
then \( \text{Sex} = \text{Male} \)

- confidence = \( \frac{\# \text{ examples with all three values}}{\# \text{ ex's with CreditCardIns=Yes and LifeIns=Yes}} \)
  \[ = \frac{2}{3} \]
  \[ = .667 \text{ or } 66.7\% \]
Practice: Support and Confidence

<table>
<thead>
<tr>
<th>Sex</th>
<th>Income Range</th>
<th>Credit Card Insurance</th>
<th>Life Insurance Promotion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Female</td>
<td>30–40K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Male</td>
<td>30–40K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>50–60K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>20–30K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Male</td>
<td>30–40K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>20–30K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Female</td>
<td>30–40K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>40–50K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>20–30K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>50–60K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Female</td>
<td>20–30K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

if LifeIns = Yes
then Sex = Female and CreditCardIns = No

• support = ?
• confidence = ?

Learning Association Rules

• For a given dataset, there are a large number of association rules that could be learned.
  • example:
    
    if CreditCardIns = Yes and LifeIns = Yes
    and IncomeRange = 20-30K
    then Sex = Female

    has a confidence of 100%, but it is only based on a single example (i.e., its support = 1)

• To cut down the number of rules that we consider, we limit ourselves to ones with sufficient support.

• Of these rules, we keep the most accurate ones – the ones with a confidence value that is above some minimum value.
Item Sets

- A **item set** is a collection of attribute values that appears in one or more training examples.
  - example: the item set `CreditCardIns=Yes, LifeIns=Yes` appears in 3 training examples
  - it could be used to form two different rules with support = 3:
    - if `CreditCardIns = Yes` then `LifeIns = Yes`
    - if `LifeIns = Yes` then `CreditCardIns = Yes`

<table>
<thead>
<tr>
<th>Sex</th>
<th>Income Range</th>
<th>Credit Card Insurance</th>
<th>Life Insurance Promotion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Female</td>
<td>30–40K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Male</td>
<td>30–40K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>50–60K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>20–30K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Male</td>
<td>30–40K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>20–30K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Female</td>
<td>30–40K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>40–50K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>20–30K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>50–60K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Female</td>
<td>20–30K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>30–40K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>30–40K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Female</td>
<td>30–40K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>40–50K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>20–30K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>20–30K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Apriori Algorithm for Learning Association Rules**

- The standard algorithm for learning association rules is called the **apriori algorithm**.

- It has two stages:
  1. gradually build up larger and larger item sets, keeping only the ones that appear in a sufficient number of training examples
    - this allows us to ensure that the rules formed from the item sets will have sufficient support
  2. form rules from the item sets and keep the ones with a confidence value that is >= some minimum value.
First Stage: Building Item Sets

- We'll limit ourselves to item sets that appear in >= 3 examples.

- We get 9 one-item sets that meet this criterion:
  - Sex=Male CreditCardIns=Yes
  - Sex=Female CreditCardIns=No
  - IncomeRange=20-30K LifeIns=Yes
  - IncomeRange=30-40K LifeIns=No
  - IncomeRange=40-50K
  - everything but IncomeRange=50-60K, which is in only 2 examples

First Stage: Building Item Sets (cont.)

- Next, we combine our one-item sets to form two-items sets, keeping those with support >= 3.

- We don't pair item sets that have different values for the same attribute, since such a pairing isn't possible.
  - example: Sex=Male and Sex=Female
First Stage: Building Item Sets (cont.)

• We end up with 15 two-item sets:
  - `Sex=Male, IncomeRange=30-40K`
  - `Sex=Male, IncomeRange=40-50K`
  - `Sex=Male, CreditCardIns=No`
  - `Sex=Male, LifeIns=Yes`
  - `Sex=Male, LifeIns=No`
  - `Sex=Female, CreditCardIns=No`
  - `Sex=Female, LifeIns=Yes`
  - `IncomeRange=20-30K, CreditCardIns=No`
  - `IncomeRange=30-40K, CreditCardIns=No`
  - `IncomeRange=30-40K, LifeIns=Yes`
  - `IncomeRange=40-50K, CreditCardIns=No`
  - `IncomeRange=40-50K, LifeIns=No`
  - `CreditCardIns=Yes, LifeIns=Yes`
  - `CreditCardIns=No, LifeIns=Yes`
  - `CreditCardIns=No, LifeIns=No`

• Within an item set, we write the items in the order given by the columns in the dataset file.

First Stage: Building Item Sets (cont.)

• To form three-item sets, we take the union of pairs of two-item sets with the same first item. Example:
  - `Sex=Male, IncomeRange=30-40K U Sex=Male, LifeIns=Yes = Sex=Male, IncomeRange=30-40K, LifeIns=Yes`

• It isn't necessary to consider any other pairs of two-item sets, even if they share an item in common.
  • example: although we could do
    - `Sex=Male, LifeIns=Yes U CreditCardIns=No, LifeIns=Yes = Sex=Male, CreditCardIns=No, LifeIns=Yes`

    we don't need to, because either:
    1) the resulting item set will be generated by two other item sets, S1 and S2, with the same first item:
      - `Sex=Male, CreditCardIns=No U Sex=Male, LifeIns=Yes`
    2) one or both of S1 and S2 didn't have enough support, and thus the resulting item set won't either
Practice: Taking the Union of Item Sets

• What possible three-item sets could we form from the following two-item sets?
  Sex=Male, IncomeRange=30-40K
  Sex=Male, IncomeRange=40-50K
  Sex=Male, CreditCardIns=No
  IncomeRange=30-40K, CreditCardIns=No
  IncomeRange=30-40K, LifeIns=Yes

First Stage: Building Item Sets (cont.)

<table>
<thead>
<tr>
<th>Sex</th>
<th>Income Range</th>
<th>Credit Card Insurance</th>
<th>Life Insurance Promotion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Female</td>
<td>30–40K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Male</td>
<td>30–40K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>50–60K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>30–40K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>20–30K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Male</td>
<td>30–40K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Female</td>
<td>30–40K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>40–50K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>20–30K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>50–60K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Female</td>
<td>20–30K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

• Out of the potential three-items sets, only 5 have sufficient support – appearing in at least 3 examples:
  Sex=Male, IncomeRange=40-50K, CreditCardIns=No
  Sex=Male, IncomeRange=40-50K, LifeIns=No
  Sex=Male, CreditCardIns=No, LifeIns=No
  Sex=Female, CreditCardIns=No, LifeIns=Yes
  IncomeRange=40-50K, CreditCardIns=No, LifeIns=No
First Stage: Building Item Sets (cont.)

- To form potential four-item sets, we take the union of pairs of three-item sets with the same first two items.
  - more generally, to form n-item sets, we take the union of pairs of (n – 1)-item sets with the same first n – 2 items
- We get only one potential four-item set: Sex=Male, IncomeRange=40-50K, CreditCardIns=No, LifeIns=No and it has enough support.
- There can't be any five-item sets (because there are only four attributes), so we're done building sets.

Results of the First Stage

- Here are all item sets with two or more items and support >= 3:
  Sex=Male, IncomeRange=30-40K
  Sex=Male, IncomeRange=40-50K
  Sex=Male, CreditCardIns=No, LifeIns=Yes
  Sex=Male, LifeIns=Yes, CreditCardIns=No, LifeIns=Yes
  Sex=Male, LifeIns=No, CreditCardIns=No, LifeIns=No
  IncomeRange=20-30K, CreditCardIns=No
  IncomeRange=30-40K, CreditCardIns=No
  IncomeRange=30-40K, LifeIns=Yes
  IncomeRange=40-50K, CreditCardIns=No
  IncomeRange=40-50K, LifeIns=No
  IncomeRange=40-50K, CreditCardIns=No, LifeIns=No
  IncomeRange=40-50K, LifeIns=No
  IncomeRange=40-50K, CreditCardIns=No, LifeIns=No
  IncomeRange=40-50K, LifeIns=No

Sex=Male, IncomeRange=40-50K, CreditCardIns=No, LifeIns=No
Second Stage: Forming the Rules

- A given item set can produce a number of potential rules.

- **Example:** the item set
  
  \[
  \text{Sex}=\text{Male}, \text{IncomeRange}=40-50K, \text{CreditCardIns}=\text{No}
  \]
  
  produces the following potential rules:
  
  - \(a)\) if \(\text{Sex}=\text{Male} \text{ and IncomeRange}=40-50K\)
    
    then \(\text{CreditCardIns}=\text{No}\)
  
  - \(b)\) if \(\text{Sex}=\text{Male} \text{ and CreditCardIns}=\text{No}\)
    
    then \(\text{IncomeRange}=40-50K\)
  
  - \(c)\) if \(\text{IncomeRange}=40-50K \text{ and CreditCardIns}=\text{No}\)
    
    then \(\text{Sex}=\text{Male}\)
  
  - \(d)\) if \(\text{Sex}=\text{Male}\)
    
    then \(\text{IncomeRange}=40-50K \text{ and CreditCardIns}=\text{No}\)
  
  - \(e)\) if \(\text{IncomeRange}=40-50K\)
    
    then \(\text{Sex}=\text{Male} \text{ and CreditCardIns}=\text{No}\)
  
  - \(f)\) if \(\text{CreditCardIns}=\text{No}\)
    
    then \(\text{Sex}=\text{Male} \text{ and IncomeRange}=40-50K\)

- We keep only the ones with confidence \(\geq\) some min value.

---

Second Stage: Forming the Rules (cont.)

<table>
<thead>
<tr>
<th>Sex</th>
<th>Income Range</th>
<th>Credit Card Insurance</th>
<th>Life Insurance Promotion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Female</td>
<td>30–40K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Male</td>
<td>30–40K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>50–60K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>20–30K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Male</td>
<td>30–40K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>20–30K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Female</td>
<td>30–40K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>40–50K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>20–30K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Female</td>
<td>50–60K</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Male</td>
<td>40–50K</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Female</td>
<td>20–30K</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- **Example:** assume we require confidence = 1.0 (100% accuracy).
  Would either of the following rules be kept?
  
  - \(a)\) if \(\text{Sex}=\text{Male} \text{ and IncomeRange}=40-50K\)
    
    then \(\text{CreditCardIns}=\text{No}\)
    
    support = ?
    
    confidence = ?
  
  - \(b)\) if \(\text{Sex}=\text{Male} \text{ and CreditCardIns}=\text{No}\)
    
    then \(\text{IncomeRange}=40-50K\)
    
    support = ?
    
    confidence = ?
Second Stage: Forming the Rules (cont.)

• In our example, there are 13 rules with conf = 1.0:

1) if LifeIns=No
   then CreditCardIns=No
2) if Sex=Male and LifeIns=No
   then CreditCardIns
3) if IncomeRange=40-50K
   then CreditCardIns=No
4) if Sex=Male and IncomeRange=40-50K
   then CreditCardIns=No and LifeIns=No
5) if IncomeRange=40-50K and LifeIns=No
   then Sex=Male and CreditCardIns=No
6) if Sex=Male and IncomeRange=40-50K and
   CreditCardIns=No
   then LifeIns=No
7) if Sex=Male and IncomeRange=40-50K and
   LifeInsPromo=No
   then CreditCardIns=No
8) if IncomeRange=40-50K and CreditCardIns=No and
   LifeInsPromo=No (continued)

Second Stage: Forming the Rules (cont.)

• 13 rules (cont.)

9. if Income Range=40-50K and LifeIns=No
    then CreditCardIns=No
10. if Sex=Male and IncomeRange=40-50K
    then LifeInsPromo=No
11. if IncomeRange=40-50K and LifeInsPromo=No
    then Sex=Male
12. if Sex=Male and IncomeRange=40-50K
    then CreditCardIns=No
13. if CreditCardIns=Yes
    then LifeIns=Yes
Learning Association Rules in Weka

- Use the Associate tab in the Weka Explorer.

- Apriori is the default algorithm, but you may want to click on its name and change some of the parameters, such as:
  - \textit{lowerBoundMinSupport}: the minimum support that item sets must have, as a percentage of the training examples (expressed in decimal form)
    - in our example, this would be $3/15 = 0.2$
  - \textit{minMetric}: the minimum confidence that the rules must have (expressed in decimal form)
    - in our example, this would be 1.0
  - \textit{numRules}: the number of rules you want to find
  - \textit{outputItemSets}: do you want to see the item sets that were formed?

Learning Association Rules in Weka (cont.)

- When it outputs item sets, Weka includes the support of each item set (as a number, not a percentage).
  - example:
    \begin{verbatim}
    Sex=Male IncomeRange=40-50000 3
    Sex=Male IncomeRange=30-40000 3
    Sex=Male CreditCardIns=No 6
    \end{verbatim}

- Weka uses the following format for the rules:
  \begin{verbatim}
  precondition \rightarrow conclusion
  \end{verbatim}
  - example
    \begin{verbatim}
    LifeIns=No 6 ==> CreditCardIns=No 6  conf:(1)
    \end{verbatim}
    which is equivalent to
    \begin{verbatim}
    if LifeIns=No
    then CreditCardIns=No
    \end{verbatim}

- note that Weka includes both support and confidence values with the rule
Managing the Efficiency of the Algorithm

• The apriori algorithm tries to generate the rules *efficiently* – i.e., taking as few steps as possible.
  • this is extremely important, since the number of possible rules increases exponentially with the number of attributes and the number of possible values of each attribute

• We've already seem some ways that it does this:
  • by only considering item sets with sufficient support
  • by building larger item sets from smaller ones that have enough support

• It also builds rules with larger conclusions (i.e., with more attributes in the then clause) from rules with smaller conclusions.

• Even with these steps, the algorithm may take too long for very large datasets.

Managing the Efficiency of the Algorithm (cont.)

• To improve the efficiency even further, we can:
  • specify a large initial support value
    • the larger the support value, the sooner the first phase will finish
  • have the algorithm gradually decrease this support value and rerun the algorithm until it has generated enough rules
    • the *delta* parameter in Weka specifies how much the support should be decreased each time