Data Mining V: Preparing the Data

Computer Science 105
Boston University
David G. Sullivan, Ph.D.

The Data Mining Process

- Key steps:
  - assemble the data in the format needed for data mining
    - typically a text file
  - perform the data mining
  - interpret/evaluate the results
  - apply the results
Denormalization

• Recall: in designing a database, we try to avoid redundancies by normalizing the data.

• As a result, the data for a given entity (e.g., a customer) may be:
  • spread over multiple tables
  • spread over multiple records within a given table

• To prepare for data warehousing and/or data mining, we often need to denormalize the data.
  • multiple records for a given entity \(\rightarrow\) a single record

• Example: finding associations between courses students take.
  • our university database has three relevant relations: Student, Course, and Enrolled
  • we might need to combine data from all three to create the necessary training examples

Transforming the Data

• We may also need to reformat or transform the data.
  • we can use a Python program to do the reformatting!

• One reason for transforming the data: many machine-learning algorithms can only handle certain types of data.
  • some algorithms only work with nominal attributes – attributes with a specified set of possible values
    • examples: \{yes, no\}
      \{strep throat, cold, allergy\}
  • other algorithms only work with numeric attributes
Discretizing Numeric Attributes

• We can turn a numeric attribute into a nominal/categorical one by using some sort of discretization.

• This involves dividing the range of possible values into subranges called buckets or bins.
  • example: an age attribute could be divided into these bins:
    child: 0-12
    teen: 12-17
    young: 18-35
    middle: 36-59
    senior: 60-

Simple Discretization Methods

• What if we don’t know which subranges make sense?

• Equal-width binning divides the range of possible values into N subranges of the same size.
  • bin width = (max value – min value) / N
  • example: if the observed values are all between 0-100, we could create 5 bins as follows:
    width = (100 – 0)/5 = 20
    bins: [0-20], (20-40], (40-60], (60-80], (80-100]
      [ or ] means the endpoint is included
      ( or ) means the endpoint is not included
  • typically, the first and last bins are extended to allow for values outside the range of observed values
    (-infinity-20], (20-40], (40-60], (60-80], (80-infinity)
  • problems with this equal-width approach?
Simple Discretization Methods (cont.)

- *Equal-frequency or equal-height binning* divides the range of possible values into N bins, each of which holds the same number of training instances.
  - example: let's say we have 10 training examples with the following values for the attribute that we're discretizing:
    
    5, 7, 12, 35, 65, 82, 84, 88, 90, 95

    to create 5 bins, we would divide up the range of values so that each bin holds 2 of the training examples:
    
    5, 7, 12, 35, 65, 82, 84, 88, 90, 95

- To select the boundary values for the bins, this method typically chooses a value halfway between the training examples on either side of the boundary.
  - examples: \( \frac{7 + 12}{2} = 9.5 \quad \frac{35 + 65}{2} = 50 \)

- Problems with this approach?

Other Discretization Methods

- Ideally, we'd like to come up with bins that capture distinctions that will be useful in data mining.
  - example: if we're discretizing *body temperature*, we'd like the discretization method to learn that 98.6 F is an important boundary value
  - more generally, we want to capture distinctions that will help us to learn to predict/estimate the class of an example

- Both equal-width and equal-frequency binning are considered *unsupervised* methods, because they don't take into account the class values of the training examples.

- There are *supervised* methods for discretization that attempt to take the class values into account.
Discretization in Weka

- In Weka, you can discretize an attribute by applying the appropriate filter to it.

- After loading in the dataset in the Preprocess tab, click the Choose button in the Filter portion of the tab.

- For equal-width or equal-height, you choose the Discretize option in the filters/unsupervised/attribute folder.
  - by default, it uses equal-width binning
  - to use equal-frequency binning instead, click on the name of the filter and set the useEqualFrequency parameter to True

- For supervised discretization, choose the Discretize option in the filters/supervised/attribute folder.

Nominal Attributes with Numeric Values

- Some attributes that use numeric values may actually be nominal attributes.
  - the attribute has a small number of possible values
  - there is no ordering to the values, and you would never perform mathematical operations on them
  - example: an attribute that uses numeric codes for medical diagnoses
    - 1 = Strep Throat, 2 = Cold, 3 = Allergy

- If you load into Weka a comma-separated-value file containing such an attribute, Weka will assume that it is numeric.

- To force Weka to treat an attribute with numeric values as nominal, use the NumericToNominal option in the filters/unsupervised/attribute folder.
  - click on the name of the filter, and enter the number(s) of the attributes you want to convert
Removing Problematic Attributes

- Problematic attributes include:
  - irrelevant attributes: ones that don't help to predict the class
    - despite their irrelevance, the algorithm may erroneously include them in the model
  - attributes that cause overfitting
    - example: a unique identifier like Patient ID
  - redundant attributes: ones that offer basically the same information as another attribute
    - example: in many problems, date-of-birth and age provide the same information
    - some algorithms may end up giving the information from these attributes too much weight

- We can remove an attribute manually in Weka by clicking the checkbox next to the attribute in the Preprocess tab and then clicking the Remove button.

Undoing Preprocess Actions

- In the Preprocess tab, the Undo button allows you to undo actions that you perform, including:
  - applying a filter to a dataset
  - manually removing one or more attributes

- If you apply two filters without using Undo in between the two, the second filter will be applied to the results of the first filter.

- Undo can be pressed multiple times to undo a sequence of actions.
Dividing Up the Data File

- To allow us to validate the model(s) learned in data mining, we'll divide the examples into two files:
  - n% for training
  - 100 – n% for testing: these should not be touched until you have finalized your model or models
  - possible splits:
    - 67/33
    - 80/20
    - 90/10

- You can use Weka to split the dataset for you after you perform whatever reformatting/editing is needed.

- If you discretize one or more attributes, you need to do so before you divide up the data file.
  - otherwise, the training and test sets will be incompatible

Dividing Up the Data File (cont.)

- Here's one way to do it in Weka:
  1) shuffle the examples by choosing the Randomize filter from the filters/unsupervised/instance folder
  2) save the entire file of shuffled examples in Arff format.
  3) use the RemovePercentage filter from the same folder to remove some percentage of the examples
     - whatever percentage you're using for the training set
     - click on the name of the filter to set the percentage
  4) save the remaining examples in a new file
     - this will be our test data
  5) load the full file of shuffled examples back into Weka
  6) use RemovePercentage again with the same percentage as before, but set invertSelection to True
  7) save the remaining examples in a new file
     - this will be our training data