Applied Data Mining

Rockhurst University

Processing and partitioning

Section 1

Preprocessing

Stepping Back

We've spent two weeks diving into classification and prediction models.

Along the way, we've glossed over some details about how to set up the your analysis and the decisions that you need to make, often before a model is ever run.

Let's start with some of the most important, and often most **painful** work that data scientists do: *data preprocessing* (aka data wrangling)

Data quality

One thing you need to consider is the **quality** of the data. How can you trust your model if the data it trains on is low quality?

According to Han, Kamber, and Pei (2012), data quality consists of six elements:

- **Accuracy**: are the data "correct"?
- **Completeness**: do you have all the data you need?
- **Consistency**: do the data all measure the same thing?
- **Timeliness**: do you have the data when you need it?
- **Believability**: do you trust the data?
- **Interpretability**: do you understand what the data mean?

Ensuring or increasing data quality is the reason for data preprocessing.

Three Tasks of Data Preprocessing

There's three main activities that we undertake when we talk about preprocessing:

- **Data Cleaning**: fill in missing values; smoothing noisy data; identifying and removing outliers; and resolving inconsistencies.
- **Data Transformation**: normalization; discretization; and concept hierarchy generation.
- **Data Reduction**: dimensionality and numerosity reduction.

Data cleaning

In the real world, data are dirty:

- **Incomplete**: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data

    e.g., Occupation = “ ” (missing data)
- **Noisy**: containing noise, errors, or outliers

    e.g., Salary = “-10” (an error)
- **Inconsistent**: containing discrepancies in codes or names,

    Age = “42”, Birthday = “03/07/2010”
    Was rating “1, 2, 3”, now rating “A, B, C”
- **Intentional** (e.g., *disguised missing* data)

    Jan. 1 as everyone’s birthday?

Incomplete (missing) data

Why do we run into missing data?

- Data don't exist or are not available: can't get the price of a smart phone in 1976.
- Error or malfunction: equipment breaks down, batteries die, etc.
- Helpful people: it was wrong so I deleted it
- Unhelpful people: refusal to provide age or income
- Shortsightedness: why do we need to keep that?

So what do you do?

Handling missing data

There are several options for dealing with missing data:

- Ignore/drop the record: usually done when class label or target variable value is missing. Generally not effective when the % of missing values per attribute varies considerably
- Fill in the missing value manually: tedious + infeasible?
- Fill in it automatically with:
  - A global constant : e.g., "unknown', a new class?!
  - The attribute mean
  - The attribute mean for all samples belonging to the same class. This is a smarter option.
  - The most probable value: inference-based such as Bayesian formula or decision tree imputation. There are many statistical methods for this.

Noisy data

What do we mean if we say data are “noisy”?

Generally it means that there is some random error or variance in a measured variable.

Why might data be noisy?

- faulty data collection instruments
- data entry problems
- data transmission problems
- technology limitation
- inconsistency in naming convention

In addition, duplicate records, incomplete data, or inconsistent data add noise to a dataset.

Handling noisy data

What can you do with noisy data?

- **Binning**: first sort data and partition into bins then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.
- **Regression**: smooth by using the fitted values from a regression function
- **Clustering**: can detect and remove outliers
- **Inspection**: automated processes detect suspicious values and humans inspect them
- **Averaging**: time series data are often smoothed by taking a *moving average*

Data transformation

Sometimes the data we have isn't a form that we can use. We can transform the data using a function that maps the entire set of values of a given attribute to a new set of replacement values such that each old value can be identified with one of the new values.
For example:

- Smoothing techniques transform noisy data.
- Attribute or feature construction: create new attributes or variables from the originals.
- Aggregation: use summary statistics
- Normalization: scale the data to fall within a specified range
- Discretization: create discrete values or ranges for continuous data

Transformation examples

- Smoothing: common for time series. More info
- Attribute or feature construction: creation of dummy variables for factors.
- Aggregation: Often done to mask individual observations: like combining people in a state or region. Also used when individual level data is “too big” — averaging a cell to get one observation for a set of characteristics instead of many observations.
- Normalization: z-score and min-max calculations for KNN.
- Discretization: Income measured in dollars becomes categorical variable with values High, Medium, Low.

Normalization reminders

Min-max normalization: map range from $[min_A, max_A]$ to $[new\_min_A, new\_max_A]$

How do we convert old value v to new value v'?

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

Say we have an income variable that ranges from \$12,000 to \$98,000 and we want it to be normalized to [0.0, 1.0]. If one observation has income = \$73,600, that gets mapped to:

$$\frac{73{,}600 - 12{,}000}{98{,}000 - 12{,}000}(1.0 - 0) + 0 = 0.716$$

Normalization 2

And we've used **Z-score normalization** (μ: mean, σ: standard deviation): $v' = \frac{v - \mu_A}{\sigma_A}$

If our income variable has $\mu = 54,000$ and $\$sigma = 16,000$ then our zscore- scaled observation is: $\frac{73{,}600 - 54{,}000}{16{,}000} = 1.225$

One more kind: **Decimal scaling**: $v' = \frac{v}{10^j}$ where j is the smallest integer such that $max(|v'|) < 1$

We do this so often in practice, we don't always realize that it's a normalization. For this method, our \$73,600 is scaled to 0.736 and the variable is now measured in hundreds of thousands of dollars and not just dollars. (What is j here?)

Discretization

Recall there are three types of attributes:

- **Nominal** — values from an unordered set, e.g., color, profession
- **Ordinal** — values from an ordered set, e.g., military or academic rank
- **Numeric** — real numbers, e.g., integer or real numbers

Discretization: Divide the range of a continuous attribute into intervals

- Interval labels can then be used to replace actual data values
- Split (top-down) vs. merge (bottom-up)
- Discretization can be performed recursively on an attribute
- Prepare for further analysis, e.g., classification

Simple discretization: binning

You have sorted data for price (in dollars): 4, 8, 9, 15, 20, 21, 24, 25, 26, 28, 29, 34 but you need a 3 class attribute: Low, Medium, High. How do you create the bins?

Partition into equal-frequency bins:

- Bin 1: 4, 8, 9, 15
- Bin 2: 20, 21, 24, 25
- Bin 3: 26, 28, 29, 34

Partition into equal-width bins (3-13, 14-24,25-35)

- Bin 1: 4, 8, 9
- Bin 2: 15, 20, 21, 24
- Bin 3: 25, 26, 28, 29, 34

Partition based on business knowledge/question (rebate based on spending level 0 — 10, 11— 25, over 25)

- Bin 1: 4, 8, 9
- Bin 2: 15, 20, 21, 24, 25
- Bin 3: 26, 28, 29, 34

Section 2

Partitioning

Partitioning

Partitioning is where we split our data up into different subsets to use for different purposes.

We touched on partitioning already: dividing the known observations into two subgroups: a) a training set and b) a test.

We fit our model with the training set and then tests the model's performance on the test set. Common splits include 60-40 (60% training set and 40% test set), 70-30, and 80-20.

Manual splits

For our first few models, we created our test and training datasets using a very simplistic rule: the first 80% of the data were the training dataset and the rest were the test dataset.
How might that be problematic?

Let's consider the "buying a computer problem again" but this time our data file looks like this:

	Age	Income	Student	Credit rating	Buys computer
1	Under30	high	no	fair	no
2	Under30	high	no	excellent	no
3	Under30	medium	no	fair	no
4	Under30	low	yes	fair	yes
5	Under30	medium	yes	excellent	yes
6	30to40	high	no	fair	yes
7	30to40	low	yes	excellent	yes
8	30to40	medium	no	excellent	yes
9	30to40	high	yes	fair	yes
10	Over40	medium	no	fair	yes
11	Over40	low	yes	fair	yes
12	Over40	low	yes	excellent	no
13	Over40	medium	yes	fair	yes
14	Over40	medium	no	excellent	no

Randomized splits

Because you can't always know how the data file you are working with was created, it is often a good idea to randomize the data when you are creating the test and training datasets.

You can do this as a preprocessing step and randomize the order of the whole data frame. Or you can use some of the functions built into R packages to do it at the time that you are creating the test and training datasets. We'll see this in the R code.

Do we always want to randomize the input data? *It depends*

In general, if you are working with **time series** data — data where the observations are each associated with a particular point in time — you may not want to randomize the data. If you are trying to predict future values, you need to know what is past and what is present so you probably don't want them to be mixed up. (Much more detailed discussion takes place in BIA 6315)