Week03 - Summary

Data Mining and Hypothesis Testing

- High level overview of statistical tests and data mining (not a deep dive)
- Provide some tools for selecting appropriate statistical tests for evaluating a predictive model, and justifying the choice of tool
- Help you seek details of how to use a statistical method or tool in the data analytic process

Types of Statistical Studies

Observational Studies

1. **Retrospective Studies**

   Example Study: Tanning and Skin Cancer

   - The observational study involves 1500 people
   - Selected a group of people who had skin cancer and another group who did not have skin cancer
   - Asked all participants whether they used tanning beds

2. **Prospective Studies**

   Example Study: Average Computer Time vs Blood Pressure

   - Enroll 100 individuals in the observational study
   - Ask them to keep track of the computer time they spend each day
   - Measure blood pressure

Observational Studies only establish correlation, not causality

Experimental Studies

Example: randomised control trials, A/B tests, drug trials

- Previous “Average Computer Time vs Blood Pressure” as experimental study:
    - 1. Control Group (computer time max 30 minutes)
    - 2. Treatment Group (computer time of at least 2 hours)
    - From the 100 individuals, 50 subjects randomly assigned to each group
    - Factor: average computer time; response: measure blood pressure of each group

Setting up an Environment

- Between subjects:

  Each subject sees one and only one condition

- Within subjects:

  Subjects see more than one or all conditions

Research Question

- Research question (Q):
    - Asks whether the independent variable has an effect
    - “If there is a change in the independent variable, will there also be a change in the dependent variable?”

- Null hypothesis (H_0):
    - The assumption that there is no effect
    - “There is no change in the dependent variable when the independent variable changes”

Hypothesis Testing

- A hypothesis test examines two opposing hypotheses about a population parameter (e.g. the mean):
    - The null hypothesis
    - The alternative hypothesis

- The null hypothesis represents our initial assumption about the parameter, and we collect evidence to possibly reject the null hypothesis in favour of the alternative hypothesis
    - Example: determine whether the mean of a population differs significantly (this has a special meaning) from a specific value or from the mean of another population

Testing Reliability with p-Values

confidence interval.

		Decision	
		Accept H_0	**Reject H_0**
Truth	H_0 (no difference)	Right Decision	Type I Error
	H_1 (difference exists)	Type II Error	Right Decision

P-value	Indicates	Reject H_0?
≤α	Strong evidence against the null hypothesis	Yes
>α	Weak evidence against the null hypothesis	No
=α	Marginal	NA

Testing which approach is better between subjects

Generate Ratings Data

- We assume different subject groups for each condition

- Each subject sees one of the layouts and is asked to rate on a 5-point Likert scale how strongly he agree or disagree with the statement:
    - Question to subjects: Page gives a good overview?
    - 1=strongly agree; 2=agree; 3=neutral; 4=disagree; 5=strongly disagree
- Outcome:
    - G_data = [1,3,3,2,4,2,3,3,1,5,2,3,4,2,1,3,2,2,1,3,2,3,4,2,1, 3,2,2,1,3,1,3,3,2,4,2,3,3,1,5]
    - L_data = [4,5,2,4,4,3,5,4,3,5,1,4,5,3,4,4,2,3,4,5,1,4,5,3,4,4,2,3,4,5,4,5,2,4,4,3,5,4,3,5]
- G_data - grid view
- L_data - list view

Setup: comparing two versions of a display

- Subjects are users of the display (or summary, interface, etc)
    - Dependent variable is user rating (or comprehension, etc)
    - Independent variable is the version of the display
- Q: Do users prefer grid view?
- H_0: Grid and list data are drawn from the same distribution

Significance: Unpaired Student's t-Test

- Tests the null hypothesis that two population means are equal
- Assumes
    - The samples are independent
    - Populations are normally distributed
    - Standard deviations are equal
- Note:

    - Multiply two-tailed p-value by 0.5 for one-tailed p-value

      (e.g., to test A > B, rather than A > B or A < B)

Significance: Mann-Whitney U Test

- Nonparametric version of unpaired t-test
- Assumes:
    - The samples are independent
    - The data is at least ordinal

Testing whether groups differ

Significance: Analysis of Variance (ANOVA)

- Tests the null hypothesis two or more groups have the same population mean
- Assumes:
    - The samples are independent
    - Populations are normally distributed
    - Standard deviations are equal

Significance: Kruskall-Wallis H-test

- Nonparametric version of ANOVA
    - doesn't assume your data comes from a particular distribution such as normal distribution
- Assumes:
    - The samples are independent
- Note:
    - Not recommended for samples smaller than 5
    - Not as statistically powerful as ANOVA

- Both ANOVA and Kruskall-Wallis H-test are extensions of the Mann-Whitney test and Unpaired Student's t-test used to compare the means of more than two populations

Model Evaluation

Measurement: Accuracy, Precision, Recall, F1

g just stands for gold - standard level.

	s=1	s=0
g=1	*TP* *(true positives)*	*FN* *(false negatives)*
g=0	*FP* *(false positives)*	*TN* *(true negatives)*

simple specificity / sensitivity table.

- Accuracy: *(TP+TN) / N*
  % correct over all instances
- Precision: *TP / (TP+FP)*
  % correct system predictions
- Recall: *TP / (TP+FN)*
  % correct gold labels
- F1: 2PR / (P+R)
  Harmonic mean of Precision and Recall

Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

- **Holdout method**
  - splits the data randomly into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation (also: Validation Set)
  - **Random sampling:** a variation of holdout
    - Repeat holdout k times; accuracy = avg. of the accuracies obtained

- **Cross-validation** (k-fold, where k = 10 is most popular)

    - Randomly partition the data into k mutually exclusive subsets, each appox, equal size
    - Leave-One-Out is a particular form of cross-validation:
        - k folds wher k = # of tuples, for small sized data

Data Mining

Types of Statistical Studies: Post-hoc Analysis

Post-hoc Analysis

- testing hypotheses formulate after data collected
- aka data dredging or data mining
- test a wider range of hypotheses faster and cheaper
- may discover surprising patterns

- Be careful with our inferences
    - testing multiple hypotheses
    - control for family-wise error rate
- Confirm findings using experiments

- We'll focus on unsupervised machine learning techniques:
    - Dimensionality reduction
    - **Association rule mining**
    - Clustering
    - Outlier detection
    - etc

Association Rule Mining

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Market-basket transactions

TID: Transaction Identifier

Items: Transactions item set

- Predict occurrence of an item based on other items in the transaction, eg:

  {Diaper} → {Beer}

  {Milk, Bread} → {Eggs, Coke}

  {Beer, Bread} → {Milk}

- Note that arrows indicate co-occurrence, not causality

Definition: Itemset

- An **itemset** is a collection of one or more items

  {Milk, Bread, Diaper}

- A **k-itemset** is an interest containing k items

Definition: Frequent Itemset

- Support count (σ) is the itemset frequency

  S({Milk, Diaper, Beer}) = 2

- Support (s) is the normalised itemset frequency

$$s = \frac{s(\{\text{Milk,Diaper,Beer}\})}{|T|} = \frac{2}{5}$$

- A frequent itemset has

  s ≥ min_support

- An association rule is an implication of the form X → Y where X and Y are itemsets

  {Milk, Diaper} → {Beer}

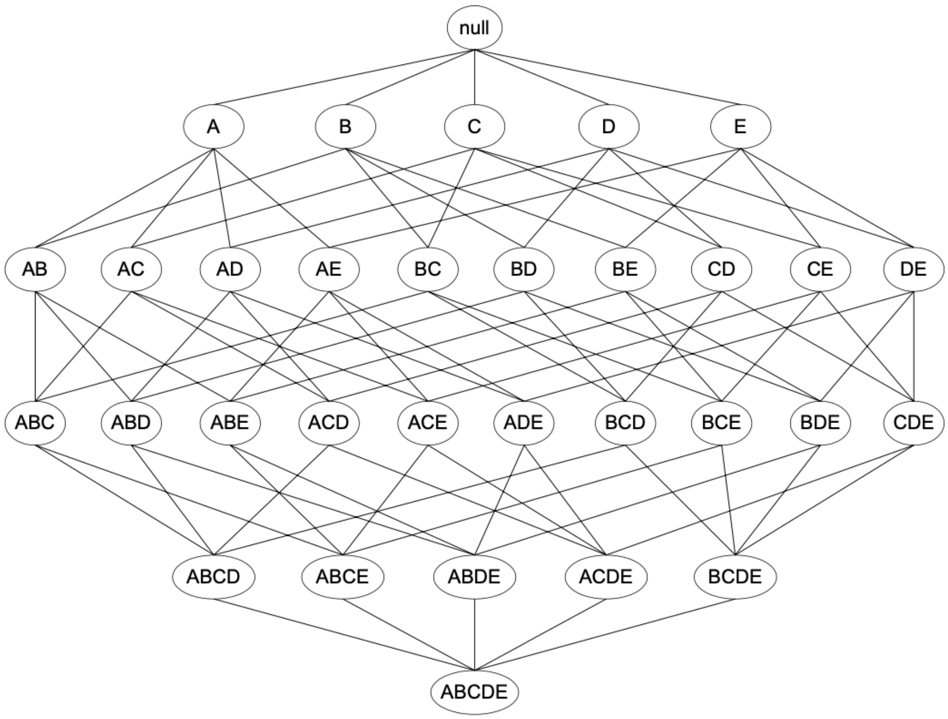
- Confidence (c) measures how often Y occurs in transactions with X

$$\boxed{c = \frac{s(\{\text{Milk,Diaper,Beer}\})}{s(\{\text{Milk,Diaper}\})} = \frac{2}{3}}$$

Mining Association Rules

1. **Frequent itemset generation**
   - Generate all itemises with s ≥ min_support
2. **Rule generation**
   - Generate high-confidence rules from each frequent itemset
   - Each rule is a binary partitioning of a frequent itemset

There are 2^d candidate itemsets!



Enumeration of 2^5 candidate itemsets for {A, B, C, D, E}

Reducing the number of candidates

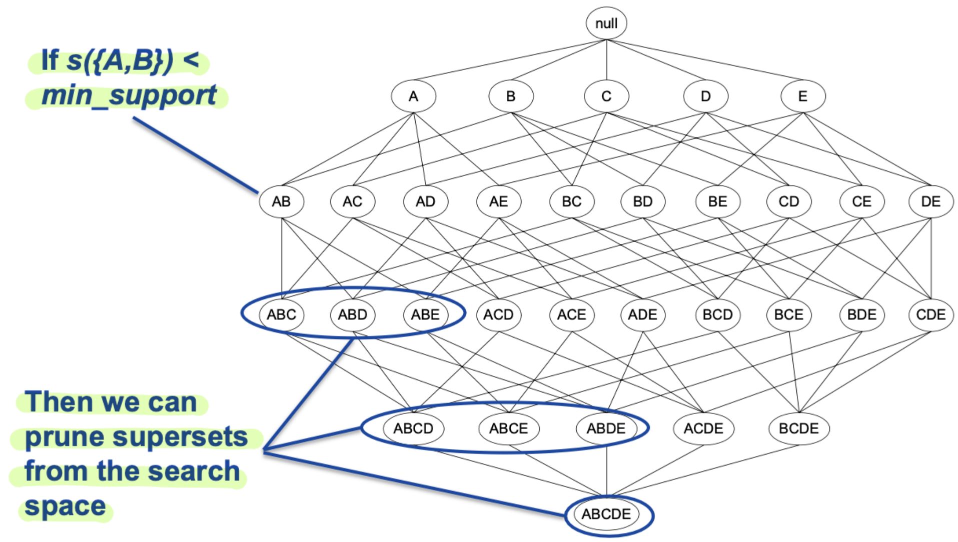
- **Apriori Principle**

  If an itemset is frequent, then all of its subsets are also frequent

- Conversely

  If an itemset is infrequent, then its supersets are also infrequent

Pruning the 2^d candidate itemsets



Apriori algorithm for generating frequent itemsets

```
While the list of (k-1)-itemsets is non empty:
  Generate candidate k-itemsets
  Identify and keep frequent k-itemsets
```

Create Initial 1-Itemsets

```
def createC1(dataset):
  "Create a list of candidate item sets of size one."
  c1 = []
  for transaction in dataset:
    for item in transaction:
      if not [item] in C1:
        c1.append([item])
  c1.sort()
  # frozenset because it will be a key of a dictionary.
  return list(map(frozenset, c1))
```

Identify itemsets that meet the support threshold

```
def scanD(dataset, candidates, min_support):
  "Returns all candidates that meets a minimum support level"
  sscnt = {}
  for tid in dataset:
    for can in candidates:
      if can.issubset(tid):
        sscnt.setdefault(can, 0)
        sscnt[can] += 1

  num_items = float(len(dataset))
  retlist = []
  support_data = {}
  for key in sscnt:
    support = sscnt[key] / num_items
    if support >= min_support:
      retlist.insert(0, key)
    support_data[key] = support
  return retlist, support_data
```

Generate the next list of candidates

```
def aprioriGen(freq_sets, k):
  "Generate the joint transactions from candidate sets"
  retList = []
  lenLK = len(freq_sets)
  for i in range(lenLK):
    for j in range(i + 1, lenLK):
      L1 = list(freq_sets[i])[:k - 2]
      L2 = list(freq_sets[j])[:k - 2]
      L1.sort()
      L2.sort()
      if L1 == L2:
        retList.append(freq_sets[i] | freq_sets[j]) # | is set union
  return retList
```

Generate all frequent itemsets

```
def apriori(dataset, min_support = 0.5):
  "Generate a list of candidate item sets"
  C1 = createC1(dataset)
  D = list(map(set, dataset))
  L1, support_data = scanD(D, C1, min_support)
  L = [L1]
```

```
  k = 2
  while (len(L[k - 2]) > 0):
    Ck = aprioriGen(L[k - 2], k)
    Lk, supK = scanD(D, Ck, min_support)
    support_data.update(supK)
    L.append(Lk)
    k += 1

  return L, support_data
```

Identify rules that meet the confidence threshold

```
def calc_confidence(freqSet, H, support_data, rules, min_confidence=0.7):
  "Evaluate the rule generated"
  pruned_H = []
  for conseq in H:
    conf = support_data[freqSet] / support_data[freqSet - conseq]
    if conf >= min_confidence:
      #print(freqSet - conseq, '-->', conseq, 'conf:', conf)
      rules_append((freqSet - conseq, conseq, conf))
      pruned_H.append(conseq)
  return pruned_H
```

Recursively evaluate rules

```
def rules_from_conseq(freqSet, H, support_data, rules, min_confidence=0.7):
  "Generate a set of candidate rules"
  m = len(H[0])
  if (len(freqSet) > (m + 1)):
    Hmp1 = aprioriGen(H, m + 1)
    Hmp1 = calc_confidence(freqSet, Hmp1, support_data, rules,
min_confidence)
    if len(Hmp1) > 1:
      rules from_conseq(freqSet, Hmp1, support_data, rules, min_confidence)
```

Mine all Association Rules

```
def generateRules(L, support_data, min_confidence=0.7):
  """Create the association rules
  L: list of frequent item sets
  support_data: support data for those itemsets
  min_confidence: minimum confidence threshold
  """
  rules = []
  for i in range(1, len(L)):
    for freqSet in L[i]:
      H1 = [frozenset([item]) for item in freqSet)
      # print("freqSet", freqSet, 'H1', H1)
      if (i > 1):
        rules_from_conseq(freqSet, H1, support_data, rules, min_confidence)
      else:
        calc_confidence(freqSet, H1, support_data, rules, min_confidence)
  return rules
```

Apriori_python

- There are also third party libraries available which implement the apriori algorithm directly
- Example:

```
from apriori_python import apriorir
dataset = [['A', 'C', 'D'], ['B', 'C', 'E'], ['A', 'B', 'C', 'E']
['B', 'E']]
freqItemSet, rules = apriori(dataset, minSup = 0.7, minConf = 0.0)
for r in rules:
  print('{}==>{} (c={])'.format(*r))
```