COMP5310: Principles of Data Science

W7: Data Mining

Presented by Claire Hardgrove
School of Computer Science

THE UNIVERSITY OF SYDNEY

Preliminaries: Project work

Some more links hypothesis testing references

- Laerd Statistics. Hypothesis testing (3 pages). https://statistics.laerd.com/statistical-guides/hypothesis-testing.php
- Frost. Understanding hypothesis tests (series of 3 blog posts). http://blog.minitab.com/blog/adventures-in-statistics/understanding-hypothesis-tests%3A-why-we-need-to-use-hypothesis-tests-in-statistics
- Frost. How to correctly interpret P values. http://blog.minitab.com/blog/adventures-in-statistics/how-to-correctly-interpret-p-values

Overview of Week 7

Today: Data Mining

Objective

Learn techniques for unsupervised learning, with tools in Python.

Lecture

- Association rule mining
- Clustering with k-means
- Choosing k
- Evaluating clustering

Readings

- Intro to Data Mining, Ch. 6
  http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf
- Intro to Data Mining, Ch. 8
  http://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf
- Data Science from Scratch, Ch. 11&19

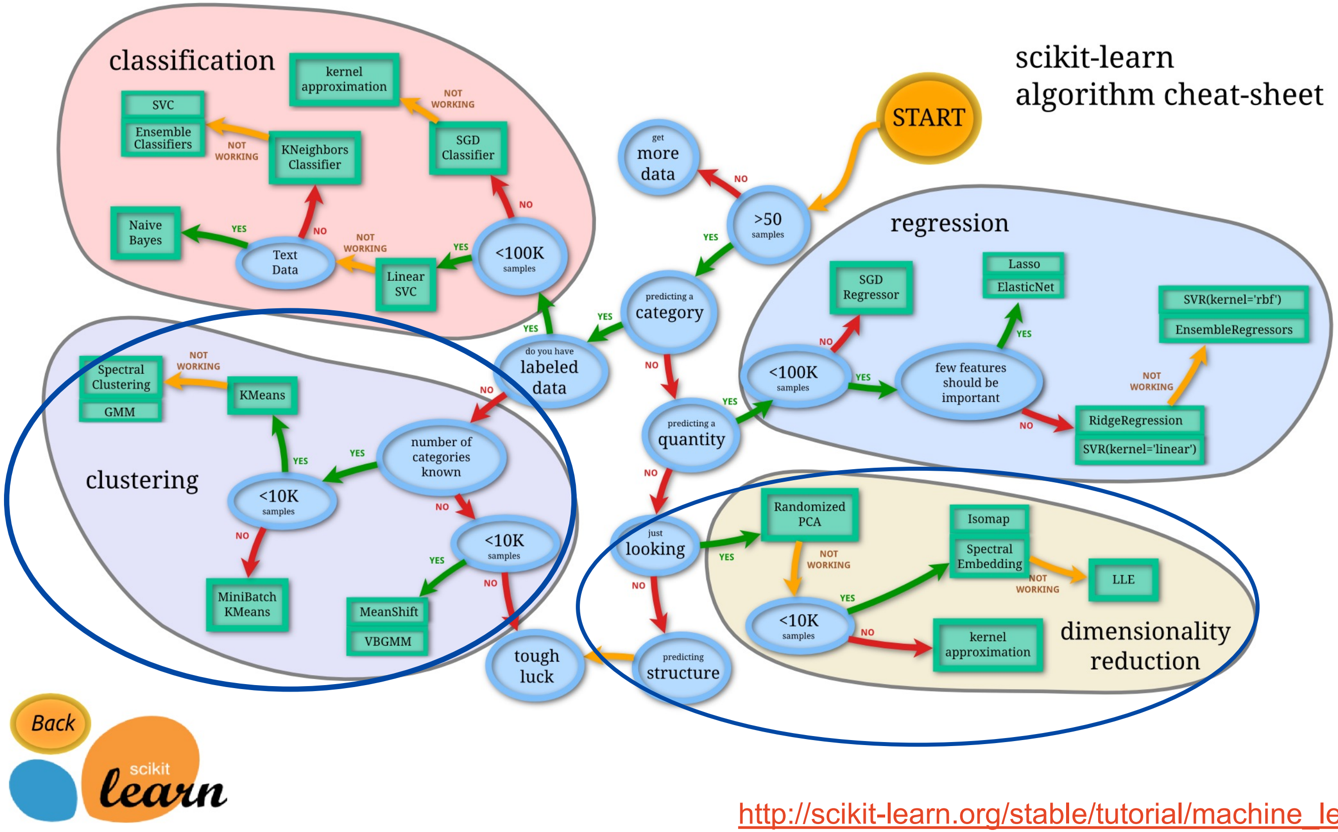
Exercises

- sklearn: clustering
- Associations from scratch

What is Data Mining?

- Extraction of knowledge from data
  https://en.wikipedia.org/wiki/Data_mining
- We'll focus on unsupervised machine learning techniques
  - Dimensionality reduction
  - *Association rule mining*
  - *Clustering*
  - Outlier detection
  - Etc.
- Textbooks often include some supervised learning as well
- Grew out of database community, often business-oriented

Machine Learning Map from Scikit-learn



http://scikit-learn.org/stable/tutorial/machine_learning_map/

Not THAT kind of Data Mining!

- Sometimes refers to *p* hacking and other bad science, e.g.:
  - Deriving hypotheses from data exploration
  - Drawing unrepresentative samples to support a hypothesis
  - Making multiple comparisons to get a significant *p*-value

- https://en.wikipedia.org/wiki/Data_dredging

Association Rule Mining

Association Analysis

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: Transaction item set

How can businesses improve sales by analysing customer purchase data?

Slides adapted from Tan et al. Introduction to data mining.
http://www-users.cs.umn.edu/~kumar/dmbook/
http://www-users.cs.umn.edu/~kumar/dmbook/dmslides/chap6_basic_association_analysis.pdf

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: Transaction 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

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: Transaction item set

- An **itemset** is a collection of one or more items
  `{Milk,Bread,Diaper}`
- A **k-itemset** is an itemset containing k items

Definition: Frequent Itemset

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: Transaction item set

- **Support count** (σ) is the itemset frequency
  $\sigma(\{\text{Milk,Diaper,Beer}\}) = 2$
- **Support (s)** is the normalised itemset frequency
  $$s = \frac{\sigma(\{\text{Milk,Diaper,Beer}\})}{|T|} = \frac{2}{5}$$
- A **frequent itemset** has $s \geq min\_support$

Definition: Association Rule

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: Transaction item set

- 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

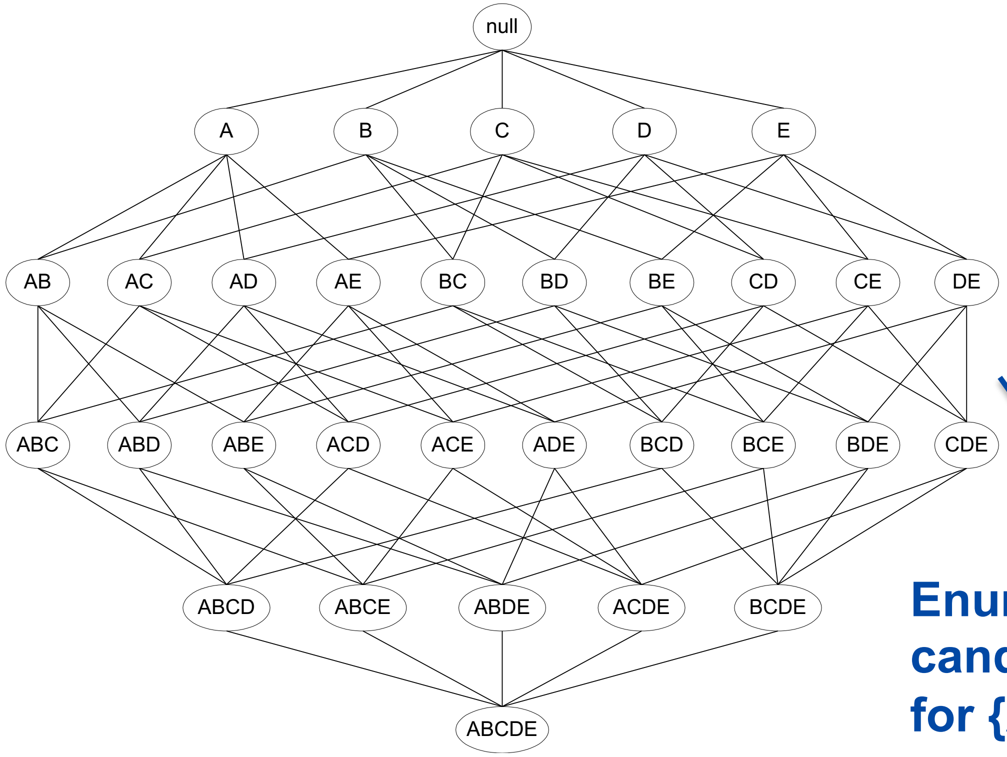
$$c = \frac{\sigma(\{\text{Milk,Diaper,Beer}\})}{\sigma(\{\text{Milk,Diaper}\})} = \frac{2}{3}$$

Mining Association Rules

1. **Frequent itemset generation**
   – Generate all itemsets with $s \geq min\_support$
2. **Rule generation**
   – Generate high-confidence rules from each frequent itemset
   – Each rule is a binary partitioning of a frequent itemset

Easy! But brute force enumerate is computationally prohibitive..

There are 2^d candidate itemsets!



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

Reducing the number of candidates

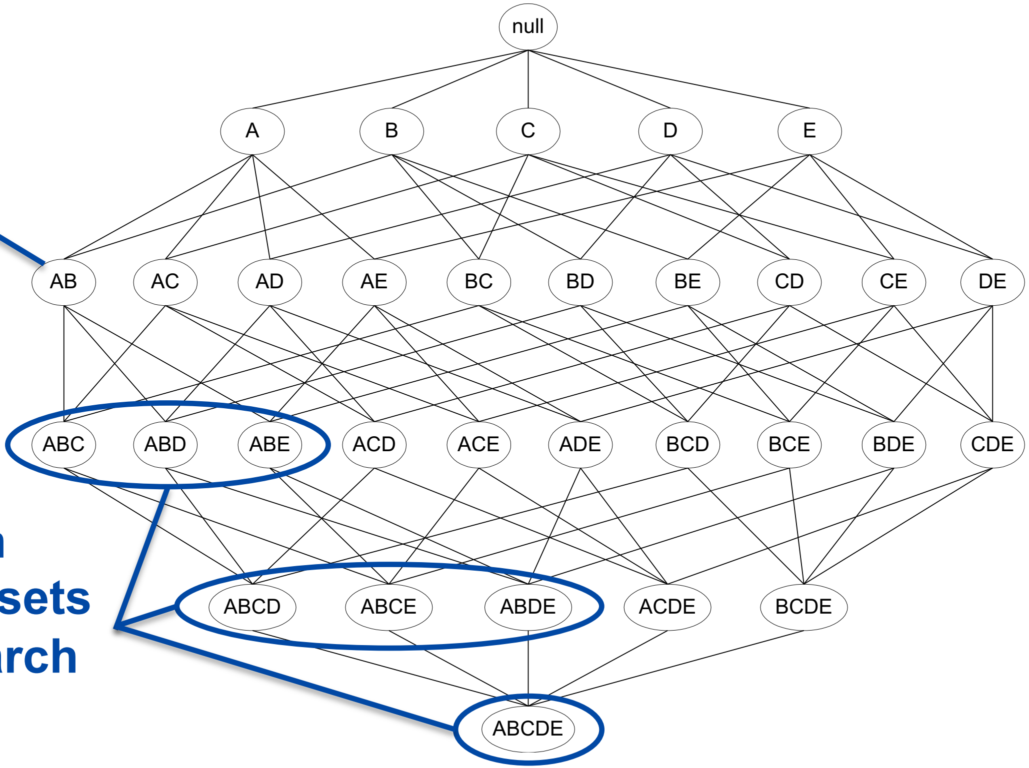
- **Apriori Principle**
  If an itemset is frequent, then all of its subsets are also frequent

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

Pruning the 2^d candidate itemsets

If *s({A,B})* ≤ *min_support*

Then we can prune supersets from the search space



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

Add each item to the initial list of candidate itemsets

```python
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 ket of a dictionary.
    return list(map(frozenset, c1))
```

Sort and return as list of sets

Identify itemsets that meet the support threshold

Calculate support counts for each candidate

```
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
```

Check whether candidates meet threshold

Generate the next list of candidates

(k-1)-itemsets

Iterate through all pairs of itemsets

Check whether pairs differ by a single item

A|B returns the union of A and B

```
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
```

Initialise L with frequent 1-itemsets

While the list of (k-1)-itemsets is non-empty:

Generate candidate k-itemsets

Identify frequent k-itemsets

Keep frequent k-itemsets

Exercise: Generating Frequent Itemsets

- Frequent itemset generation
  - code cell after “Generate frequent itemsets”
  - code cell after “Itemset generation on sample data”
- Exploring support
  - Try different support thresholds
  - What’s a reasonable value for real data?
  - Do you have datasets that resemble transactions?
  - What about the apps/websites you use?

Identify rules that meet the confidence threshold

Frequent itemset (rule components)

Possible consequences (RHS of rule)

Rule accumulator

```
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
```

Calculate confidence

Return consequences that pass the confidence threshold

Add rule to accumulator if $c \geq min\_confidence$

Recursively Evaluate Rules

Need at least 1 item for LHS

Generate candidate consequence itemsets

Update rules and return consequences that pass confidence threshold

```python
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)
```

Recurse with new consequence candidates

Mine all Association Rules

For each k
For each k-itemset

```
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
```

Initial consequence candidates

Recursively evaluate rules if k>1

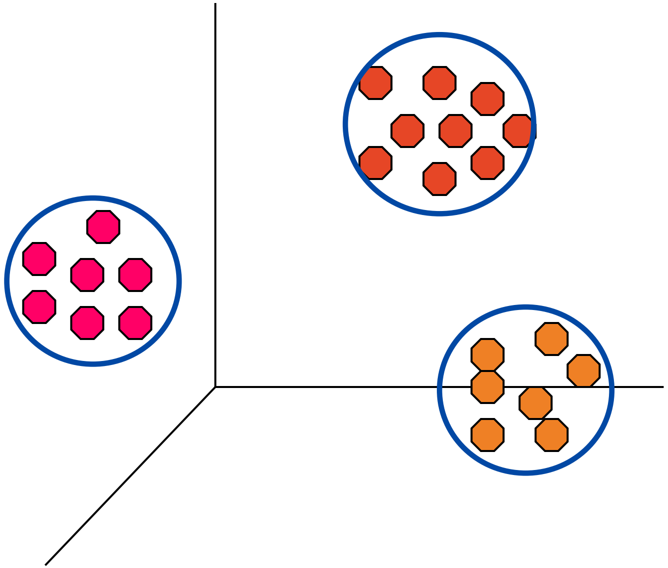
Evaluate H1 only

Exercise: Association Rule Mining

- Frequent itemset generation
  -  code cell after “Mine association rules”
  -  code cell after “Rule mining on sample data”
- Exploring confidence
  - Try different confidence thresholds
  - What’s a reasonable value for real data?
  - Can we use this for recommendation (e.g., Amazon, Netflix)?

Clustering with *k*-Means

Clustering: Group Similar Objects



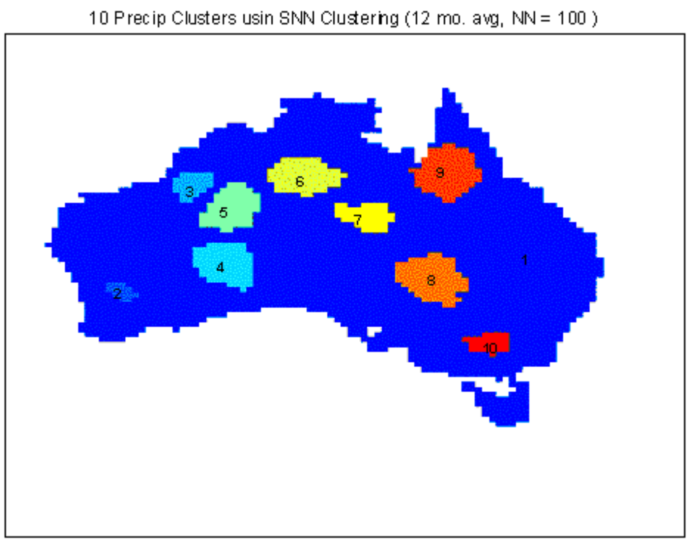
- Groups objects into clusters
- Objects in the same cluster are similar/related
- Objects in different clusters are dissimilar/unrelated

Slides adapted from Tan et al. Introduction to data mining.
http://www-users.cs.umn.edu/~kumar/dmbook/
http://www-users.cs.umn.edu/~kumar/dmbook/dmslides/chap6_basic_association_analysis.pdf

Clustering for Understanding

- Group related documents for browsing
- Group genes, proteins, or cells that have similar functionality
- Group stocks with similar price fluctuations
- etc

Clustering for Summarisation



- Reduces the size of large datasets
- Summarise data before further analysis
- Vector quantisation for, e.g., images, audio, video
- etc

http://www-users.cs.umn.edu/~kumar/dmbook/dmslides/chap8_basic_cluster_analysis.pdf

Notion of a Cluster can be ambiguous

How many clusters..?

Maybe six clusters

Two clusters?

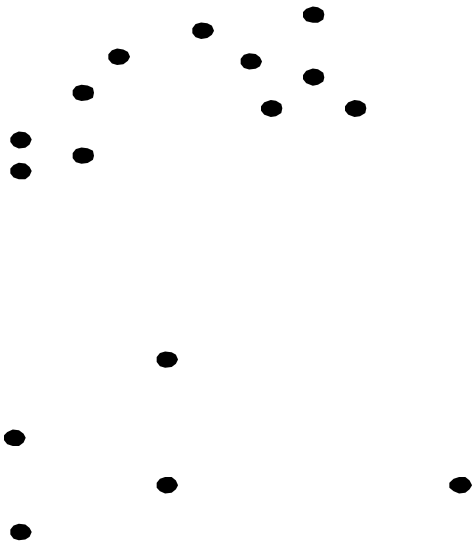
Or four clusters

http://www-users.cs.umn.edu/~kumar/dmbook/dmslides/chap8_basic_cluster_analysis.pdf

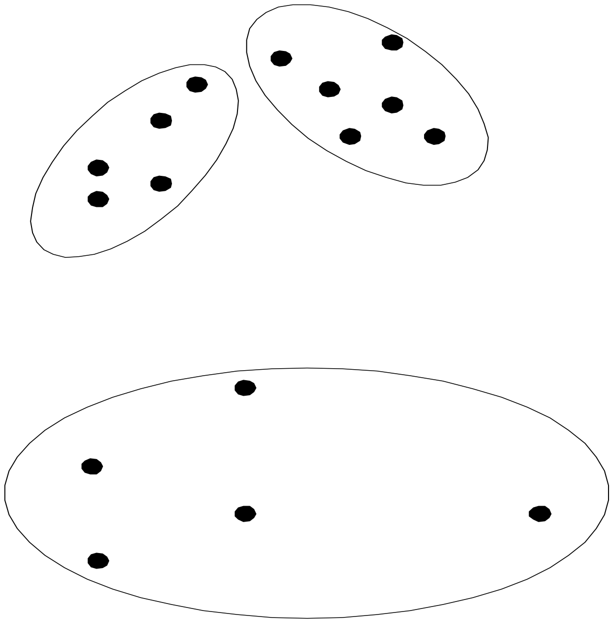
Types of Clusterings

- A clustering is a set of clusters
- Important distinction between hierarchical and partitional sets of clusters
- ***Partitional clustering***
  A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- ***Hierarchical clustering***
  A set of nested clusters organized as a hierarchical tree

Partitional Clustering

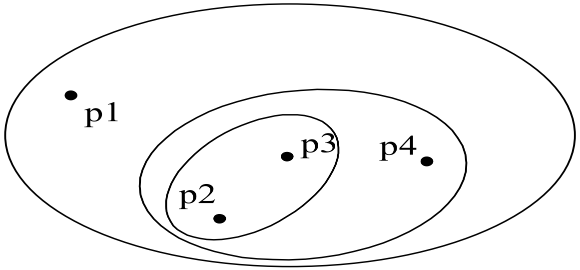


Original Points

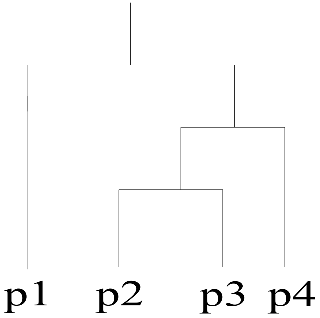


A Partitional Clustering

Hierarchical Clustering



Hierarchical clustering



Dendrogram

k-Means Clustering

- Partitional clustering approach
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, *k*, must be specified
- The basic algorithm is very simple

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

k-Means Clustering – Details

- Initial centroids are often chosen randomly
- Clusters produced vary from one run to another
- K-means will converge for common similarity measures
- Most of the convergence happens in the first few iterations
- Complexity is O($n * k * i * d$)
  - n = number of points, k = number of clusters,
    i = number of iterations, d = number of attributes (or dimensions)
  - Sklearn suggests <10k points
  - Also need to consider k, i and d

Exercise: Cluster Analysis

- Clustering
  - code cell after “Load handwritten digit dataset”
  - code cell after “Clustering handwritten digits”
  - TODO Explore different initialisations

Optimal clustering in 1D

- Optimal interval clustering: Application to Bregman clustering and statistical mixture learning
  https://arxiv.org/pdf/1403.2485.pdf

Evaluating Clustering

Cluster Validity

- For supervised classification we have a variety of measures to evaluate how good our model is (accuracy, precision, recall)
- For cluster analysis, the analogous question is how to evaluate the "goodness" of the resulting clusters?
- While clusters are in the eye of the beholder, we may still want to evaluate them…
  - To avoid finding patterns in noise
  - To compare clustering algorithms
  - To compare two sets of clusters
  - To compare two clusters
- http://scikit-learn.org/stable/modules/clustering.html#clustering-evaluation

Measures of Cluster Validity

- Numerical measures that are applied to judge various aspects of cluster validity, are classified into the following three types.
- External Index: Measure the extent to which cluster labels match externally supplied class labels (e.g., v-measure)
- Internal Index: Measure the goodness of a clustering structure *without* respect to external information (e.g., SSE)
- Relative Index: Compare two different clusterings or clusters (often an external or internal index is used)

External Evaluation Measures

- Homogeneity ranges from 0 to 1, measuring whether clusters contain data points that are part of a single class
  (analogous to precision)
- Completeness ranges from 0 to 1, measuring whether classes contain data points that are part of a single cluster
  (analogous to recall)
- V-measure is the harmonic mean of homogeneity and completeness
  (analogous to F1 score)
- http://scikit-learn.org/stable/modules/clustering.html#homogeneity-completeness-and-v-measure

Internal: Sum of Squared Error (SSE, or inertia)

- For each point, the error is the distance to the nearest cluster
- To get SSE, we square these errors and sum them.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

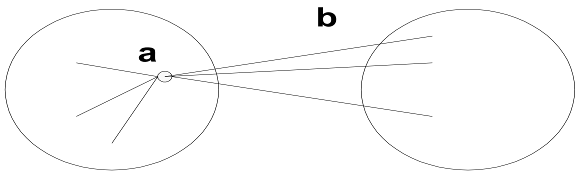
- x is a data point in cluster C_i and m_i is the representative point (mean) for cluster C_i

Internal: Silhouette Coefficient

- For an individual point i
  - Calculate $\boldsymbol{a}$ = average distance of i to points in its cluster
  - Calculate $\boldsymbol{b}$ = average distance of i to points in the next nearest cluster
  - The silhouette coefficient for a point is then given by

$s = 1 – a/b$ if $a < b$, (or $s = b/a - 1$ if $a \geq b$, not the usual case)

  - The closer to 1 the better



- Silhouette coefficient for dataset is average across all i

Exercise: Evaluation

- Evaluating with respect to a gold partition
  -  code cell after “Evaluating clustering”
  -  code cell after “Comparing initialisations”
  - TODO Discuss evaluation output

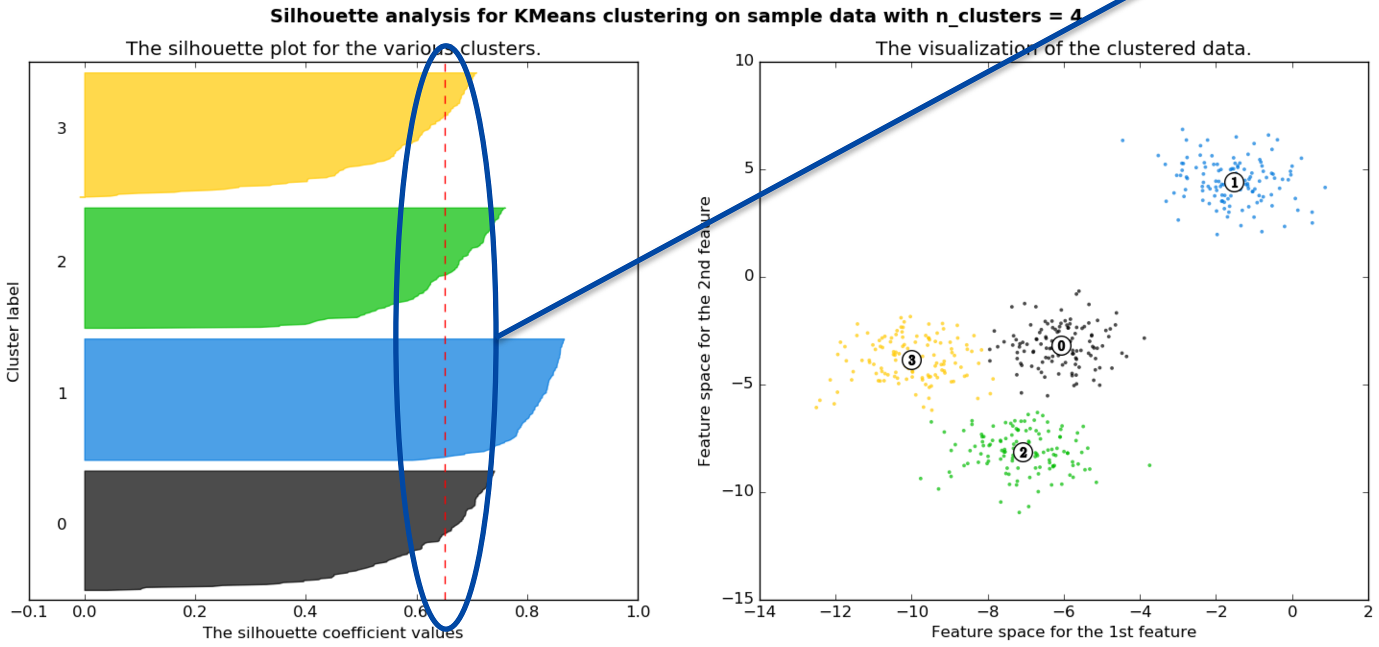
Choosing k

Choosing *k*

- Often we don't know the number of clusters in our data
- Selecting *k* is generally and interactive process
- There are some approaches to aid interactive selection, and sometimes automate it

Using Silhouettes to choose *k*

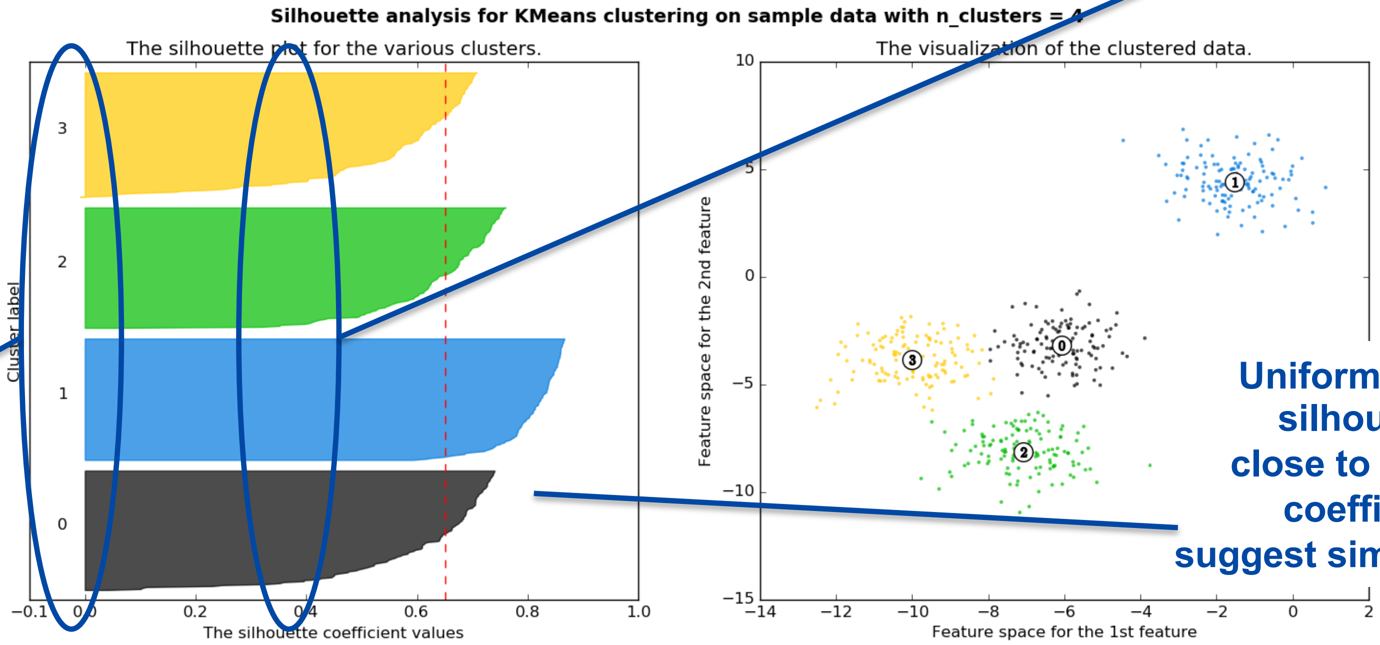
High average silhouette indicates points far away from neighbouring clusters



http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Using Silhouettes to choose *k*

Bar chart showing silhouette values for each item grouped by cluster



Cluster IDs

Uniform cluster silhouettes close to average coefficient suggest similar quality

http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Exercise: Choosing *k*

- Selecting the number of clusters
  -  code cell after “Create example data for choosing k”
  -  code cell after “Choosing k using silhouette analysis”
  - TODO Choosing k by plotting sum of squared error

Review

THE UNIVERSITY OF SYDNEY

Pre-Processing for Clustering

- We only looked at choosing *k*, but more pre-processing needed
- Data Cleansing
- Data Transformation
- **Data Normalisation**
- **Dimensionality Reduction** / choice or projection of dimensions
  - closely related to choice of **distance metric**
  - we used Euclidean Metric so far (L_2-Norm)
  - but others possible too, eg. Manhatten Distance (L_1-Norm)
  - "Curse of high dimensionality"; cf Aggarwal et al.: "On the Surprising Behavior of Distance Metrics in High Dimensional Space", 2001.

More on Clustering

- *k*-Means clustering is a well-known, simple algorithm
  - but requires choice of *k* and is sensitive to start-choice of *k*
- Many more sophisticated algorithms available nowadays
  - eg. density-based clustering which supports unknown number of clusters
  - eg. automatic sub-space clustering (choice of relevant dimensions)
  - eg. support of per-cluster feature-selection
  - eg. outlier handling/pruning
  - etc.

- cf. Kriegel et al.: "Clustering High-Dimensional Data: A Survey on Subspace Clustering, Pattern-Based Clustering, and Correlation Clustering", ACM KDD 2009.

Today: Data Mining

Objective

Learn techniques for unsupervised learning, with tools in Python.

Lecture

- Association rule mining
- Clustering with k-means
- Choosing k
- Evaluating clustering

Readings

- Intro to Data Mining, Ch. 6
  http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf
- Intro to Data Mining, Ch. 8
  http://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf
- Data Science from Scratch, Ch. 11&19

Exercises

- sklearn: clustering
- Associations from scratch

TODO in W7

- define experimental framework for A2

Additional Reading (not examinable)

- Tan et al. Introduction to data mining.
  https://goo.gl/hWwuZb
- Aggarwal. Data mining: the textbook.
  https://goo.gl/lQqLwT
- Han. Data mining: concepts and techniques.
  https://goo.gl/CFIMMs
- Scikit-learn user guide, § 2 (Unsupervised learning).
  http://scikit-learn.org/stable/unsupervised_learning.html

Other tools and Techniques (not examinable)

- Scikit-learn user guide, § 4.4 (Dimensionality reduction).
  http://scikit-learn.org/stable/modules/unsupervised_reduction.html
- Scikit-learn user guide, § 2.7 (Outlier detection).
  http://scikit-learn.org/stable/modules/outlier_detection.html
- Etc

Assignment 2: Project Stage 2

THE UNIVERSITY OF SYDNEY

Project Stage 2A: Summarise and Analyse

Objective

Summarise and analyse the data

Output

- See the specification on Canvas

Marking

- 10% of overall mark

Project and Discussion Time

Time for you to talk
to tutors, instructors and each other
about experiment setup, approach
details and results/analysis.