

Exploring graphs

- Have seen examples of finding paths through graphs that represent physical networks
- Can also use graph search to explore changes to state of a physical system
 - Nodes represent states of system
 - Edges represent actions that cause a change of state
 - Want to find sequence of actions to convert system to desired state

Solving the 8-puzzle

1	2	5
6	3	8
	4	7

	1	2
3	4	5
6	7	8

- Slide any tile into blank space
- Find sequence of slides to reach goal

An example solution

1	2	5
6	3	8
	4	7

An example solution

1	2	5
	3	8
6	4	7

An example solution

1	2	5
3		8
6	4	7

An example solution

1	2	5
3	4	8
6		7

An example solution

1	2	5
3	4	8
6	7	

An example solution

1	2	5
3	4	
6	7	8

An example solution

1	2	
3	4	5
6	7	8

An example solution

1		2
3	4	5
6	7	8

An example solution

	1	2
3	4	5
6	7	8

How could we solve this?

- Each node in graph is a state of the puzzle
 - Specific layout of tiles in puzzle
- Each edge of the graph specifies which tile to slide to get to a new state of the puzzle
- Solution space is very large:
 - $9!$ or 362880 nodes
 - Each node has 2, 3, or 4 edges
 - Graph has almost 1 million edges
- Is there another way?

An implicit graph

- Suppose we represent each node in graph as a state of the puzzle:
 - Specific layout of tiles
 - Rather than generate all nodes, only generate as needed to explore paths
 - Edges then become implicit – create to generate new node

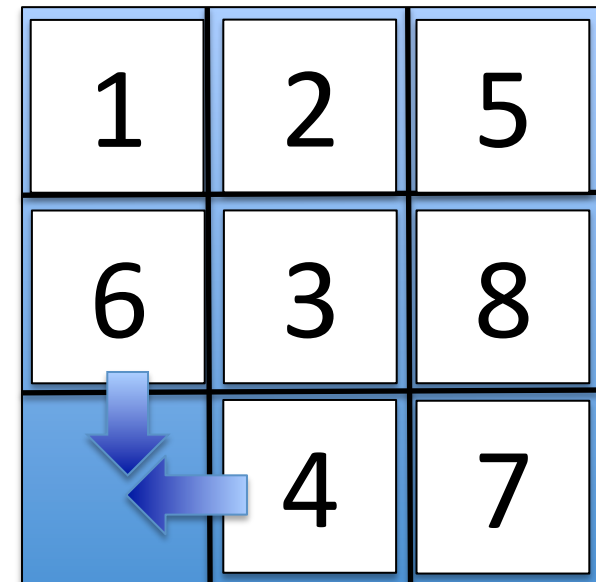
A node

- Convert sequence of tiles into a string of labels – use to represent a node
 - Example: 125638047
- Keep track of blank tile
 - In slot 6 in example
- Need a way to encode edges

1	2	5
6	3	8
	4	7

An “edge”

- Need to encode legal shifts
 - In example, tile in 3rd spot can shift down to 6th spot
 - And tile in 7th spot can shift left to 6th spot
- Could keep track of these legal shifts in a dictionary
 - Index by location in grid
 - Return legal new locations



An implementation: node

```
class puzzle(object):
    def __init__(self, order):
        self.label = order
        for index in range(9):
            if order[index] == '0':
                self.spot = index
        return None
    def transition(self, to):
        label = self.label
        blankLocation = self.spot
        newBlankLabel = str(label[to])
        newLabel = ''
        for i in range(9):
            if i == to:
                newLabel += '0'
            elif i == blankLocation:
                newLabel += newBlankLabel
            else:
                newLabel += str(label[i])
        return puzzle(newLabel)
    def __str__(self):
        return self.label
```


An implementation: implicit edges

```
shiftDict = {}  
shiftDict[0] = [1, 3]  
shiftDict[1] = [0, 2, 4]  
shiftDict[2] = [1, 5]  
shiftDict[3] = [0, 4, 6]  
shiftDict[4] = [1, 3, 5, 7]  
shiftDict[5] = [2, 4, 8]  
shiftDict[6] = [3, 7]  
shiftDict[7] = [4, 6, 8]  
shiftDict[8] = [5, 7]
```

	1	2
3	4	5
6	7	8



An implementation: search

```
def BFSWithGenerator(start, end, q = []):  
    initPath = [start]  
    q.append(initPath)  
    while len(q) != 0:  
        tmpPath = q.pop(0)  
        lastNode = tmpPath[len(tmpPath) - 1]  
        if lastNode.label == end.label:  
            return tmpPath  
        for shift in shiftDict[lastNode.spot]:  
            new = lastNode.transition(shift)  
            if notInPath(new, tmpPath):  
                newPath = tmpPath + [new]  
                q.append(newPath)  
    return None
```

Checking for loops

```
def notInPath(node, path):  
    for elt in path:  
        if node.label == elt.label:  
            return False  
    return True
```

An implementation: search

```
def DFSWithGenerator(start, end, stack = []):  
    initPath = [start]  
    stack.insert(0, initPath)   
    while len(stack) != 0:  
        tmpPath = stack.pop(0)  
        lastNode = tmpPath[len(tmpPath) - 1]  
        if lastNode.label == end.label:  
            return tmpPath  
        for shift in shiftDict[lastNode.spot]:  
            new = lastNode.transition(shift)  
            if notInPath(new, tmpPath): #avoid cycles  
                newPath = tmpPath + [new]  
                stack.insert(0, newPath)   
    return None
```

Searching for a puzzle solution

- Running the breadth first search finds a solution with 8 moves
 - Searches 220 nodes
- Running the depth first search takes a very long time!! (and runs out of memory on my machine!!)

Maximum cliques

- For some problems, finding subgraphs of a graph that are complete can be important
- Complete means that for every node in the graph, it is connected to every other node
- Examples:
 - Finding sets of people in a social network that all know each other
 - Finding subjects in an infected population that all have had contact with one another

Clique example

- Given a set of users on a social network
 - For a given user, want to find the set of all other users that know this person, and know everyone else in the group
 - Want to find the largest set of users who all know each other
 - Want to separate the network into collections of cliques

Use of cliques

- Analyzing communication networks
- Designing circuits
- Analyzing gene expression data
- Analyzing social networks
- Analyzing disease networks among infected populations

Max clique

- While more efficient algorithms exist, the brute force method can solve small sized problems
 - Find all subgraphs of a graph
 - Test each one to see if complete
 - Keep track of the largest
- Can extend to recursively find other large cliques by removing nodes of largest clique and repeating

Generating a power set

- Given a set of elements, would like to find set of all subsets
 - [1, 2, 3]
 - Leads to [], [1], [2], [3], [1, 2], [1, 3], [2, 3], [1,2,3]
- To find, can use recursive approach:
 - Find power set of all but first element
 - Then copy each element of that set, with first element added
 - Combine both sets as answer

Powerset

```
def powerSet(elts):  
    if len(elts) == 0:  
        return [[]]  
    else:  
        smaller = powerSet(elts[1:])  
        elt = [elts[0]]  
        withElt = []  
        for s in smaller:  
            withElt.append(s + elt)  
        allofthem = smaller + withElt  
        return allofthem
```

Finding cliques

- Generate power set of nodes – gives set of all possible subgraphs
- Test each one to see if complete (i.e., all nodes connected)
- Keep track of largest clique found

Power Graph

```
def powerGraph(gr):  
    nodes = gr.nodes  
    nodesList = []  
    for elt in nodes:  
        nodesList.append(elt)  
    pSet = powerSet(nodesList)  
    return pSet
```

Complete graphs

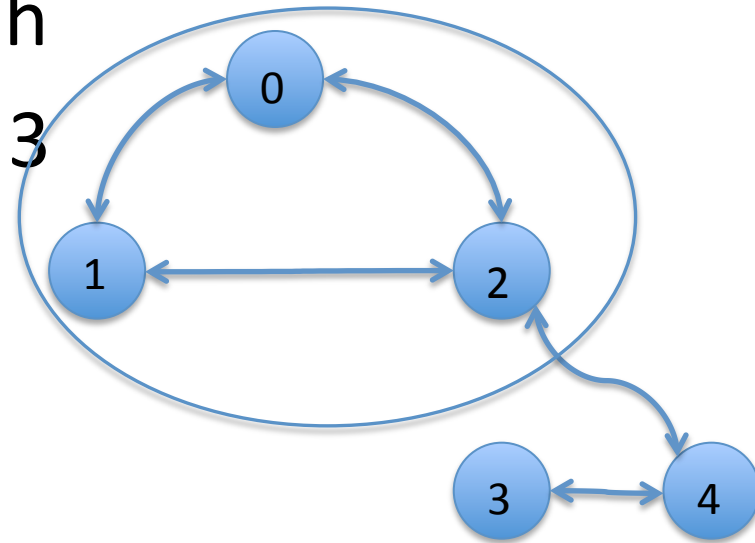
```
def allConnected(gr, candidate):  
    for n in candidate:  
        for m in candidate:  
            if not n == m:  
                if n not in gr.childrenOf(m):  
                    return False  
    return True
```

Max Clique implementation

```
def maxClique(gr):  
    candidates = powerGraph(gr)  
    keepEm = []  
    for candidate in candidates:  
        if allConnected(gr, candidate):  
            keepEm.append(candidate)  
    bestLength = 0  
    bestSoln = None  
    for test in keepEm:  
        if len(test) > bestLength:  
            bestLength = len(test)  
            bestSoln = test  
    return bestSoln
```

An example

- Simple example of graph
- Largest Clique is of size 3



Graphs

- Useful data structure for capturing relationships between objects
- Optimization problems can often be cast as graph search problems
 - Depth first and breadth first searches are common ways to find optimal paths through graph

A Quick Introduction to Machine Learning

Lecturer: John Guttag



What Is Machine Learning (ML)?

Automating automation

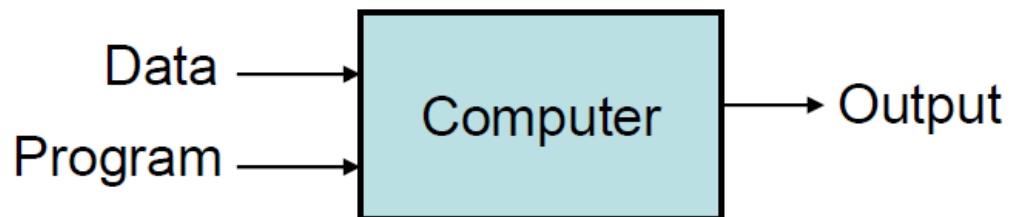
Computer programs can automatically follow rules
How do we determine these rules automatically?

ML focuses on getting computers to program themselves

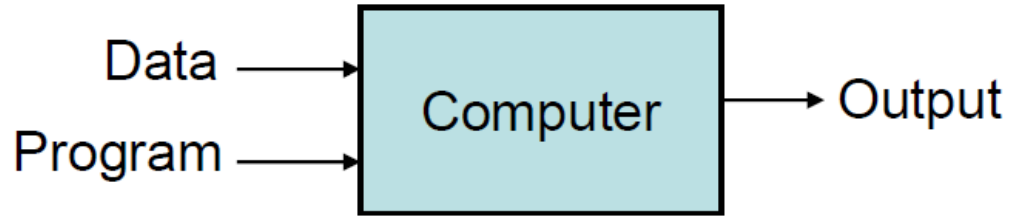
Let the data do the work

Automatically generate programs that create
useful outputs from data

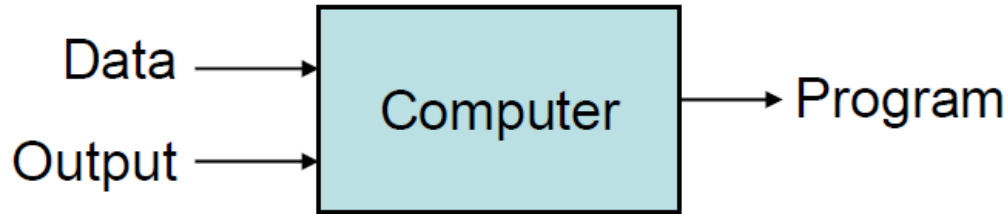
Traditional Programming



Traditional Programming



Machine Learning



How Do People Learn Things?

Memorization

Accumulation of individual facts

Limited by

Time to observe and memory to store the facts



How Do People Learn Things?

Generalization (what machine learning uses)

Deduce new facts from old ones



How Do People Learn Things?

Generalization (what machine learning uses)

Deduce new facts from old ones



How Do People Learn Things?

Generalization (what machine learning uses)

Deduce new facts from old ones

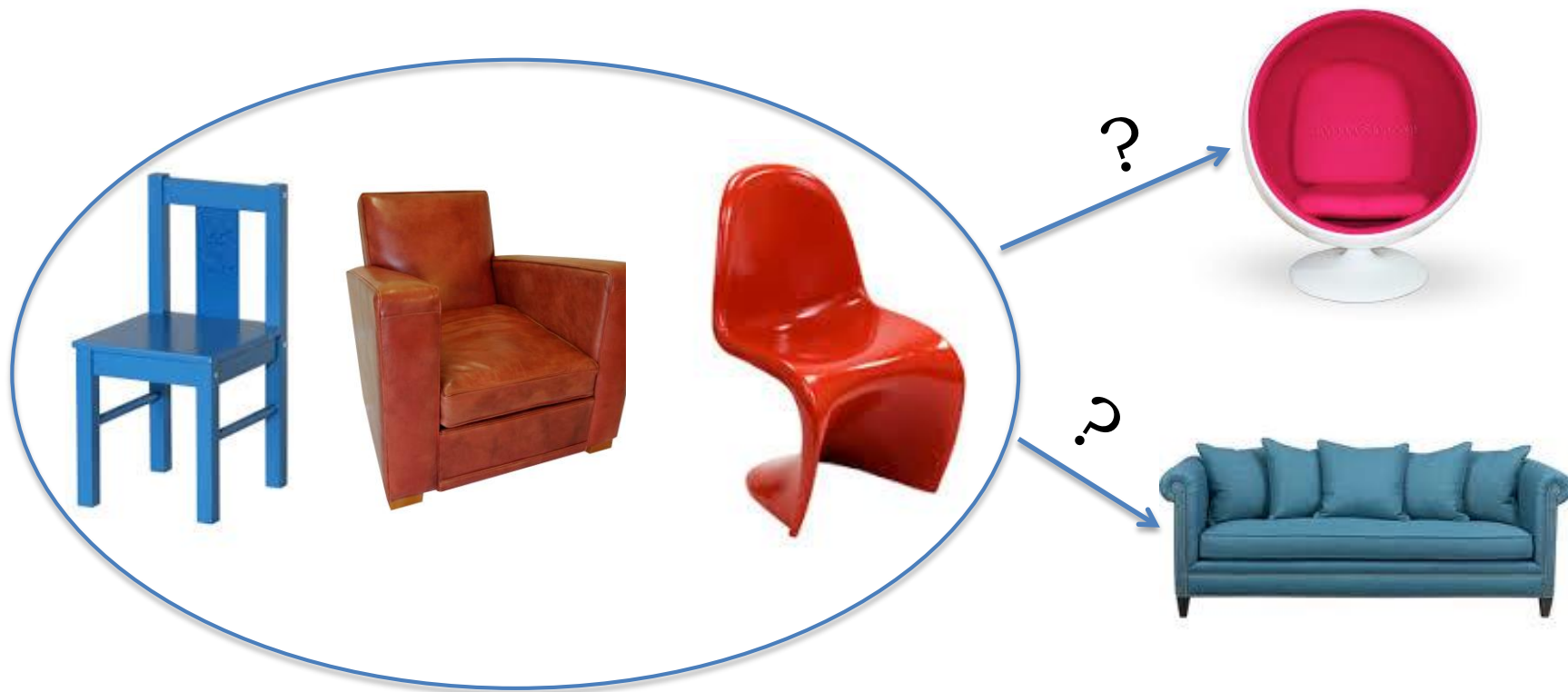


Limited by number of facts and accuracy of deductions

A Quick Introduction to Machine Learning

Lecturer: John Guttag

Machine Learning Is About Generalization



Major Components

Method for representing the data

E.g., represent a place by its GPS coordinate

Metric for assessing goodness of the model (program)

E.g., given the GPS of a new place, how good is the model at guessing the continent

Optimization method for learning the model

E.g., linear regression

Two Broad Classes

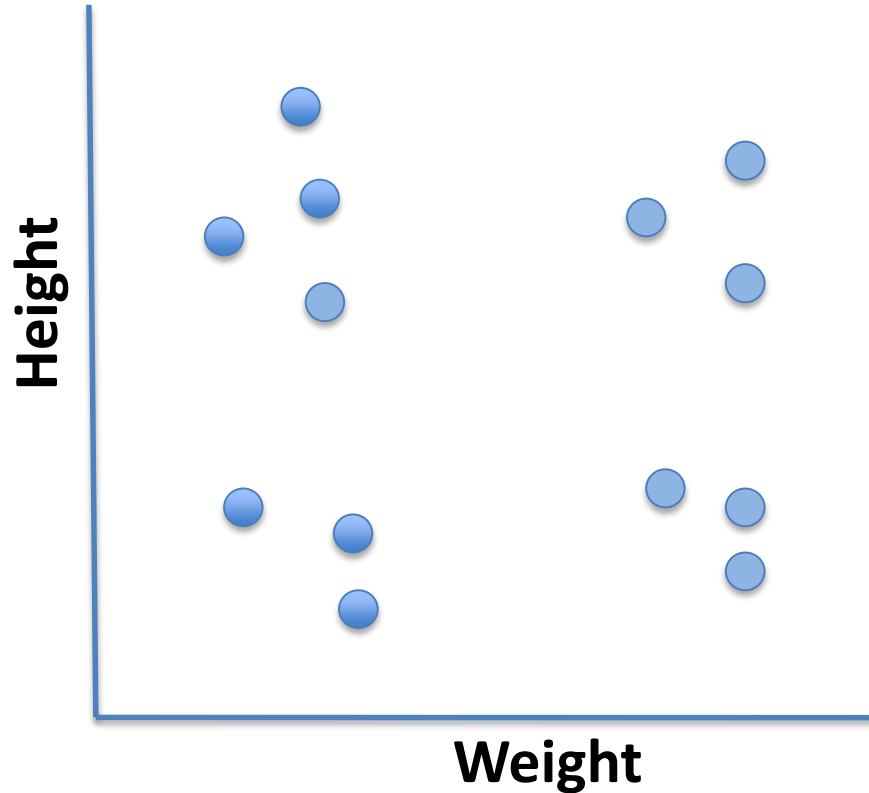
Supervised

Given a set of feature/label pairs, find a rule that predicts the label associated with a previously unseen input

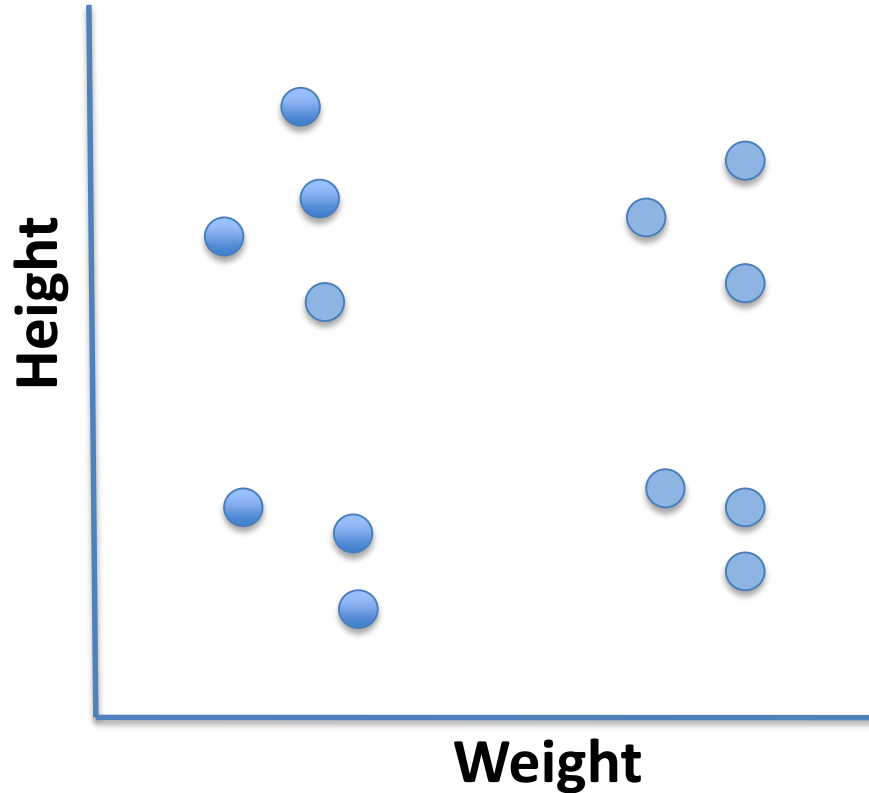
Unsupervised

Given a set of feature vectors (without labels), group them into “natural clusters”

Unsupervised Learning Example



Unsupervised Learning Example



Supervised Learning Example

Consider building a model to predict whether a person speaks English

Suppose we chose to represent a person using

- Three features

 - eye color, gender, citizenship

- Label indicating whether they speak English

What Rule Might We Learn?


Given

P1: <blue, male, U.S.> : true


P2: <brown, male, U.S.>: true

P3: <blue, female, Chile>: false

We probably want to learn a rule something like

For all x, y : < x, y , U.S.>  speaks English

Equally likely to learn

For all x, z : < x , male, z >  speaks English

Moral

If the number of examples is small relative to the number of features, it's easy to make false generalizations

John Guttag is male and speaks English
Eric Grimson is male and speaks English
Therefore all males speak English

Moral

If the number of examples is small relative to the number of features, it's easy to make false generalizations

Weather for Mumbai, Maharashtra, India



Moral

If the number of examples is small relative to the number of features, it's easy to make false generalizations



Suppose We Had 100 Million Samples Drawn from Around the Globe

Probably would not learn the rule

For all x, z : $\langle x, \text{male}, z \rangle \Rightarrow$ speaks English

But would we learn the rule

For all x, y : $\langle x, y, \text{U.S.} \rangle \Rightarrow$ speaks English?

It Depends

Not if we want prediction to be right 100% of time

Need to use a method that allows for training error

Want to learn something that is probably true

Remember: “All models are wrong, but some are useful”

-- George Box

A Quick Introduction to Machine Learning (Clustering)

Lecturer: John Guttag

Clustering

Find an intrinsic grouping in set of unlabeled examples

Of great practical utility

- Marketing

- Biology

- Insurance

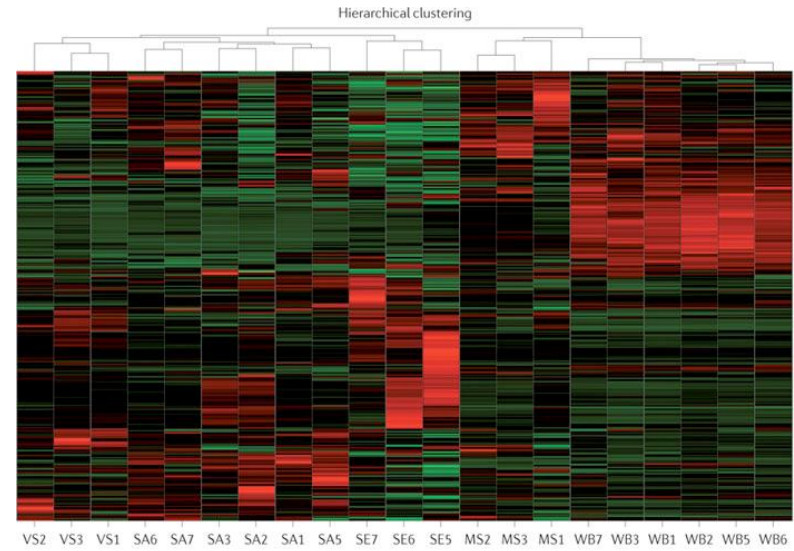
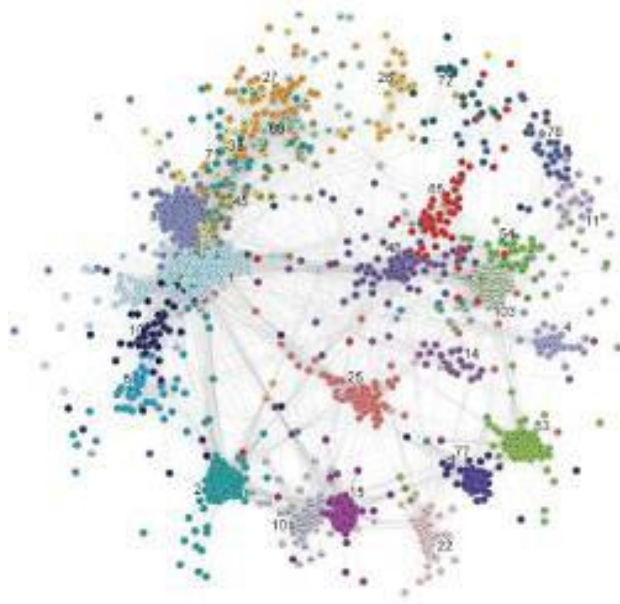
- Medicine

- ...

Marketing



Biology and Medicine



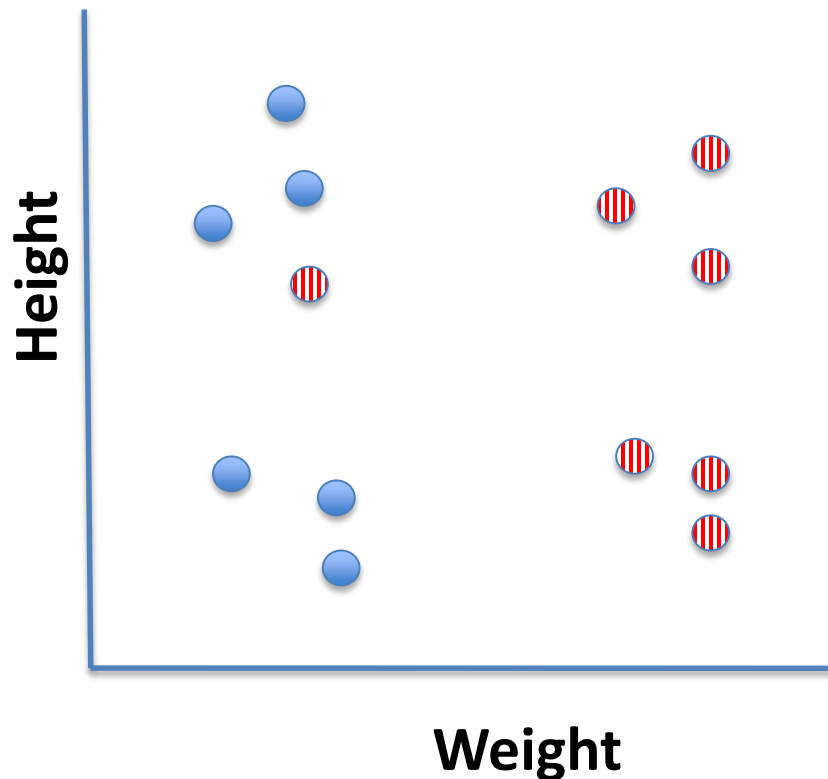
Nature Reviews | **Genetics**

Insurance



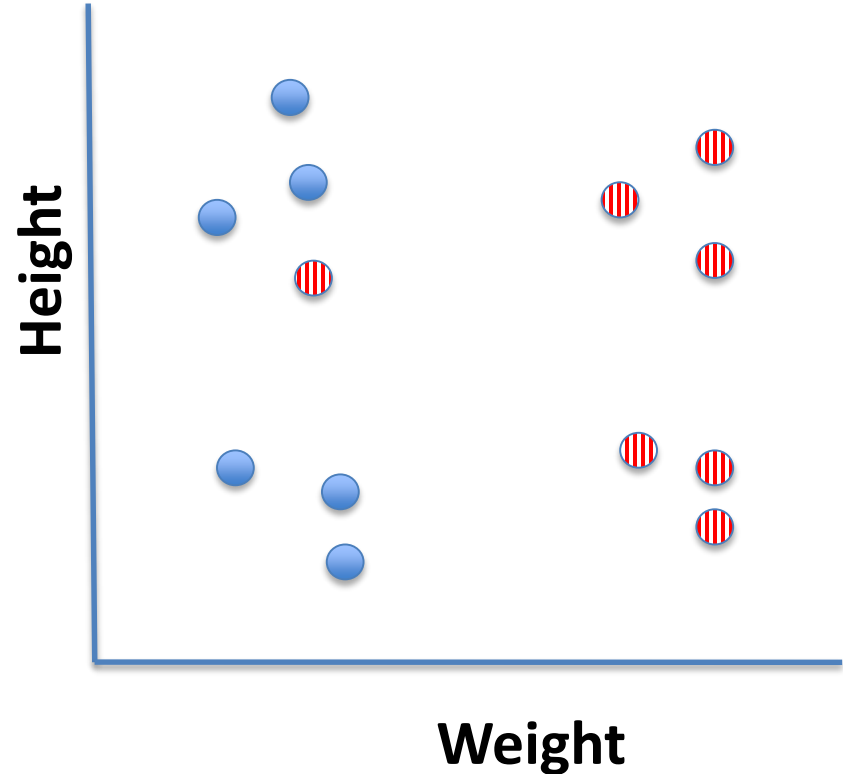
What Makes a Clustering Good?

Depends upon the application



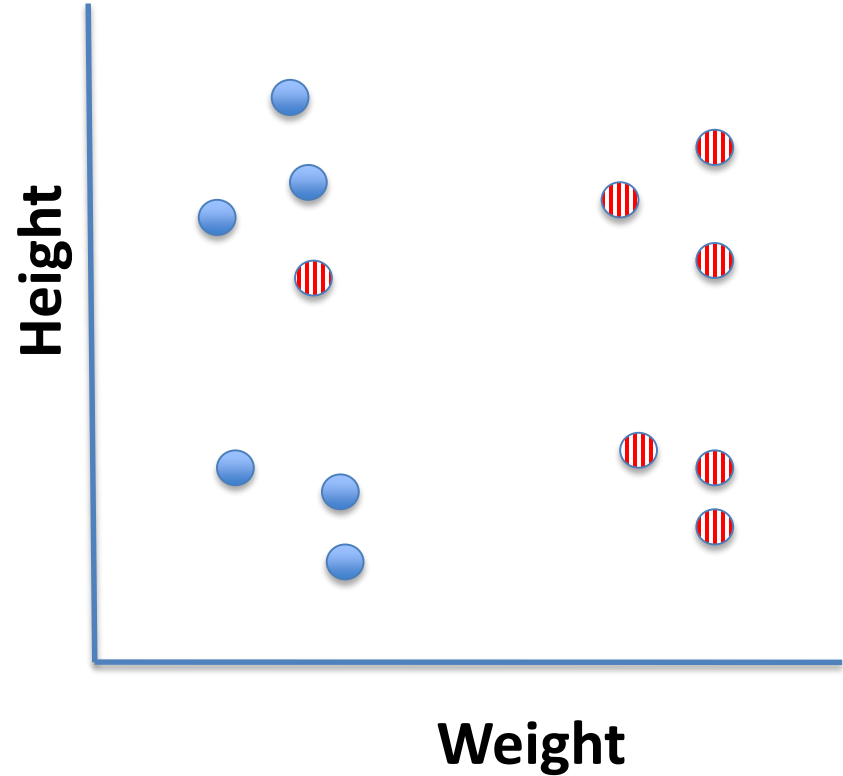
What Makes a Clustering Good?

Depends upon the application
Basketball player



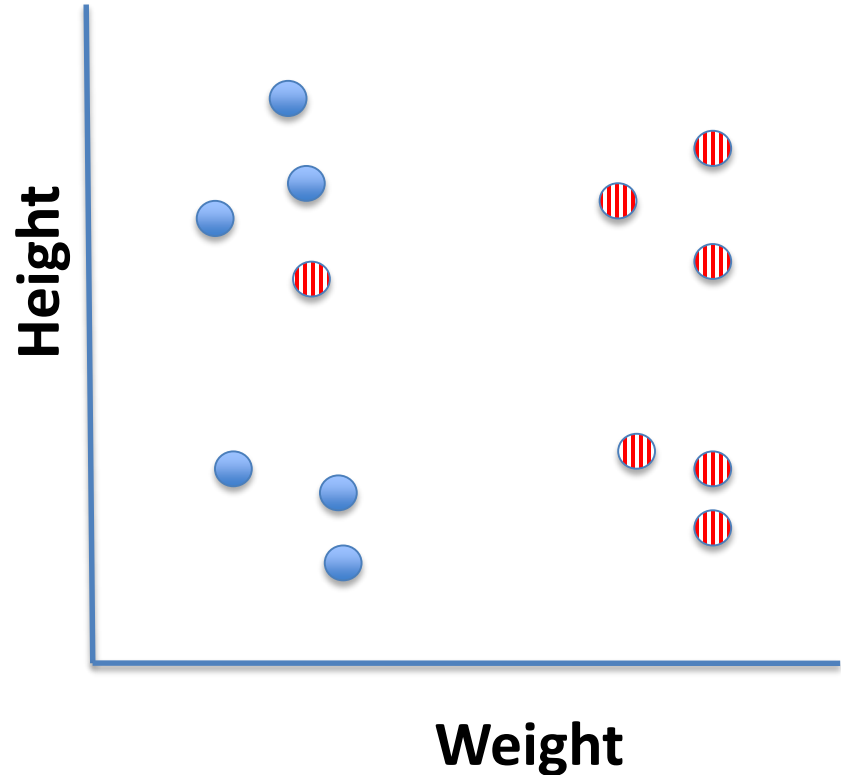
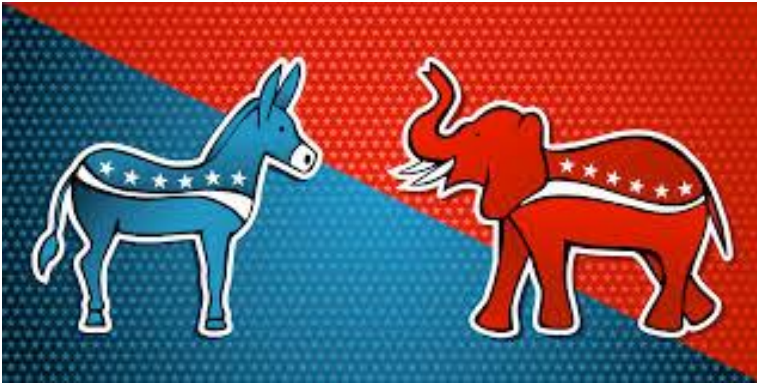
What Makes a Clustering Good?

Depends upon the application
Sumo wrestler



What Makes a Clustering Good?

Depends upon the application
Political candidates



Like All ML, It's an Optimization Problem

Like All ML, It's an Optimization Problem

Need an objective function

Low intra-cluster dissimilarity

High inter-cluster dissimilarity

Intra-cluster Dissimilarity

$$V(c) = \sum_{x \in c} (\text{mean}(c) - x)^2$$

$$\text{badness}(C) = \sum_{c \in C} V(c)$$

Are We Done?

Sufficient to find a set of clusters, C , such that $\text{badness}(C)$ is minimized?

Suppose each example is in a cluster of size 1?

$$\text{badness}(C) = ?$$

What do we need?

Need a Constraint

Maximum distance between clusters is D

The maximum number of clusters is k

A Classic Formulation of Optimization

An objective function and a constraint

Like many optimization problems, computationally nasty

Usually rely on a greedy approximation

- K-means

- Hierarchical

A Quick Introduction to Machine Learning (Hierarchical Clustering)

Lecturer: John Guttag

Clustering an Optimization Problem

An objective function and a constraint

Like many optimization problems, computationally nasty

Usually rely on a greedy approximation

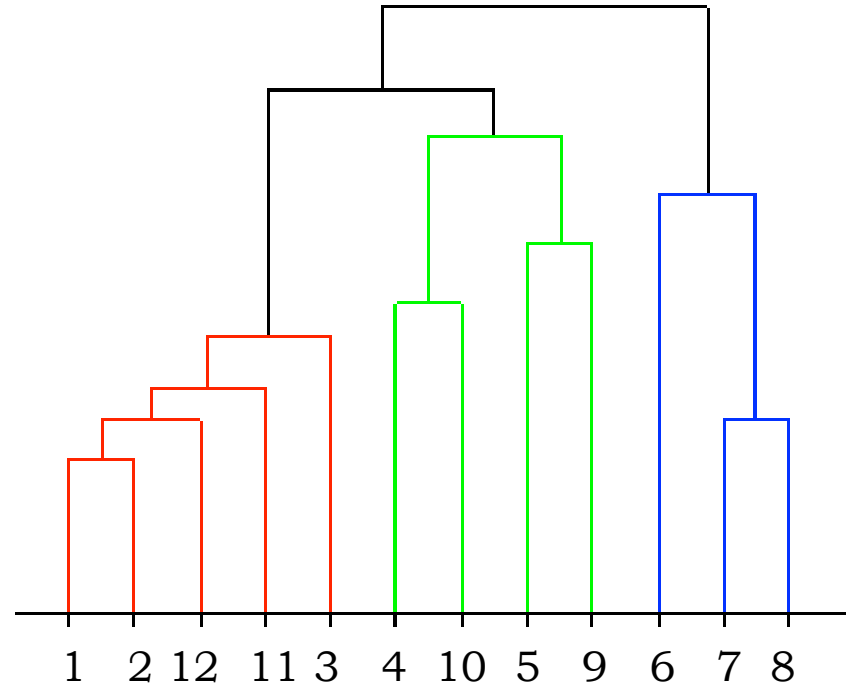
- Hierarchical

- K-means

Hierarchical Clustering

1. Start by assigning each item to a cluster, so that if you have N items, you now have N clusters, each containing just one item.
2. Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now you have one cluster fewer.
3. Continue the process until all items are clustered into a single cluster of size N .

Dendrogram (Monthly Temperatures)



Linkage Criteria

In *single-linkage* clustering (also called the *connectedness* or *minimum* method), we consider the distance between one cluster and another cluster to be equal to the shortest distance from any member of one cluster to any member of the other cluster.

Linkage Criteria, continued

In *complete-linkage* clustering (also called the *diameter* or *maximum* method), we consider the distance between one cluster and another cluster to be equal to the greatest distance from any member of one cluster to any member of the other cluster.

Linkage Criteria, continued

In *average-linkage* clustering, we consider the distance between one cluster and another cluster to be equal to the average distance from any member of one cluster to any member of the other cluster. A slight variant of this uses the median instead of the mean.

Linkage Criterion

	BOS	NY	CHI	DEN	SF	SEA
BOS	0	206	963	1949	3095	2979
NY		0	802	1771	2934	2815
CHI			0	966	2142	2013
DEN				0	1235	1307
SF					0	808
SEA						0

BOS NY CHI DEN SF SEA

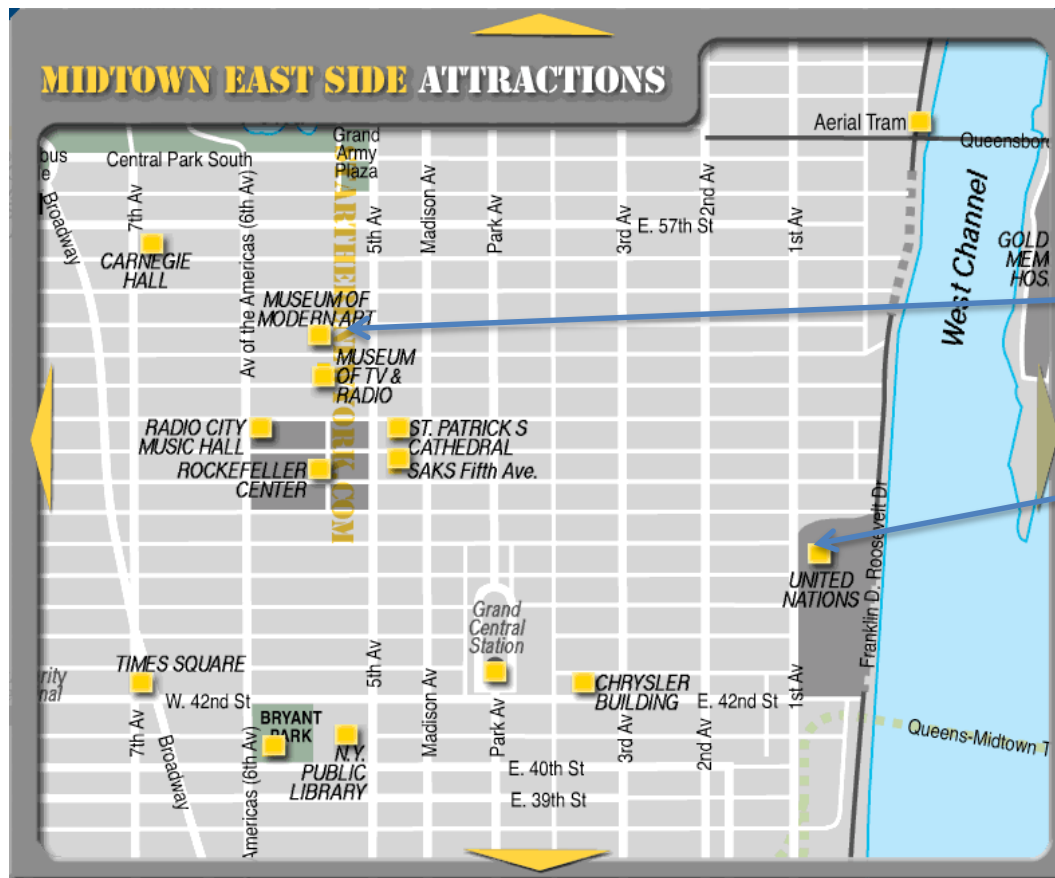
{BOS, NY} CHI DEN SF SEA

{BOS, NY, CHI} DEN SF SEA

{BOS, NY, CHI} DEN {SF, SEA}

{BOS, NY, CHI, DEN} {SF, SEA, DEN}

{BOS, NY, CHI, DEN, SF, SEA}



55th and 5th

46th and 1st

Distance(<55,5>, <46,1>)

Minkowski Metric

$$\text{dist}(X1, X2, p) = \left(\sum_{k=1}^{\text{len}} \text{abs}(X1_k - X2_k)^p \right)^{1/p}$$

p = 1: Manhattan Distance

P = 2: Euclidean Distance

