The CaRDS Institute

An Institute for Computation and Research in Data Science

Vertex Weighted Feature Engineering in Machine Learning

Jeff and Debra Knisley

Monday, October 17, 2016

Coming up with features is difficult, time-consuming, requires expert knowledge. "Applied machine learning" is basically feature engineering.

— Andrew Ng, Stanford University

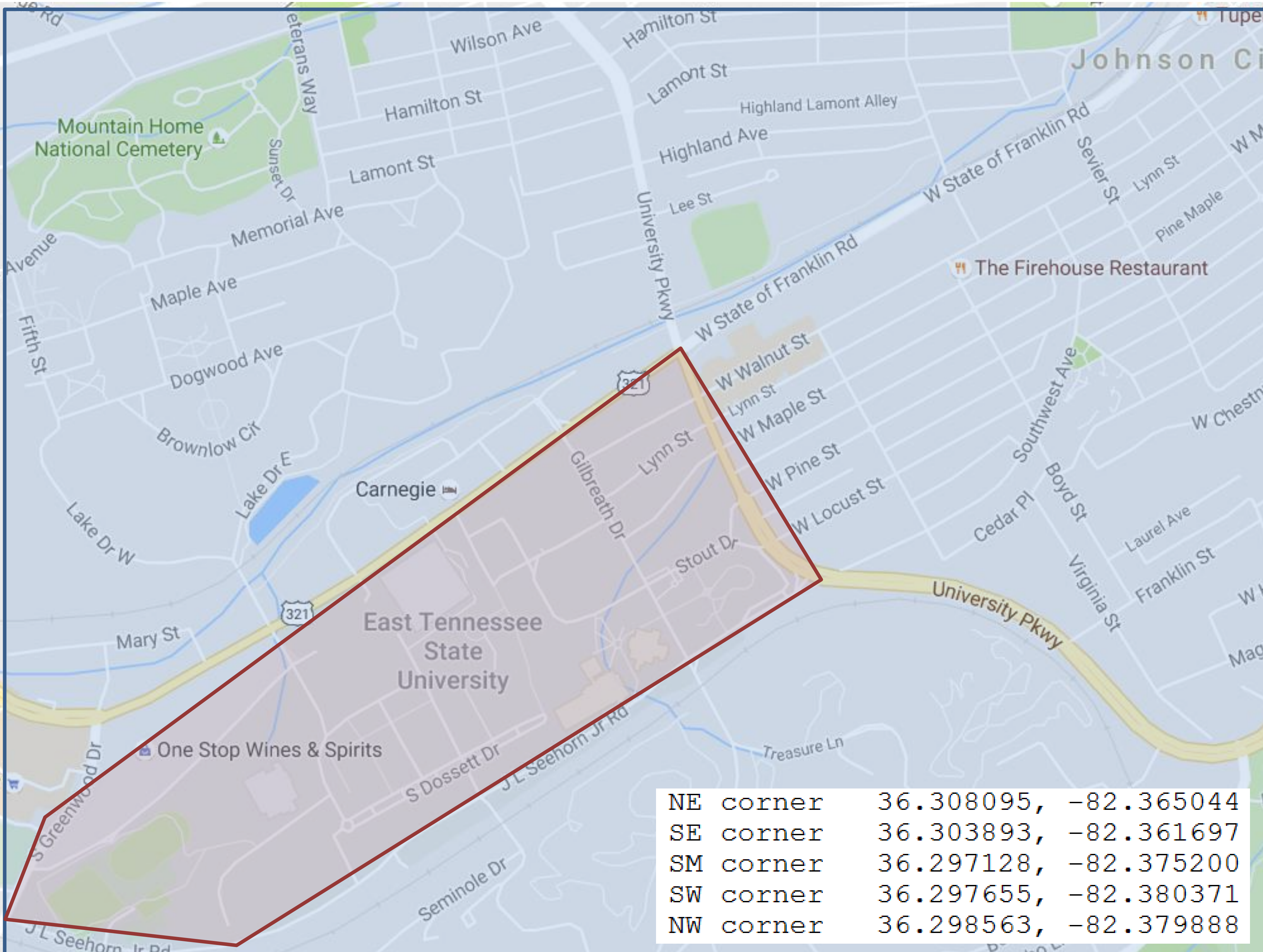
Quick Review: “Big Data”

- Data Scientists tend to use the “3 v’s”
  - **High Volume:** Extremely Large Datasets
  - **High Variety:** Many types, Highly C
  - **High Velocity:** Data so large or c t computational speed is a majo
- KEY CONCEPT: High Variety is
  - Kaggle Titanic Tutorial Competitio
    - Predict if a given passenger survived
    - High variety of passenger features and circumstances
    - Small Dataset: 1309 passengers each with 10 features
  - But Complexity, Variety often require “High Volume”

> Pedagogical Challenge:
>
> More High Variety with only medium volume.

Big Data Example: Twitter Data

- Easy to collect
  - Collected using python tweepy
  - Location based (used a box containing ETSU)



NE corner	36.308095, -82.365044
SE corner	36.303893, -82.361697
SM corner	36.297128, -82.375200
SW corner	36.297655, -82.380371
NW corner	36.298563, -82.379888

Twitter Data

- Easy to collect
  - Collected using python tweepy
  - Location based (used a box containing ETSU)
- Many features
  - 'text', 'in_reply_to_status_id_str', 'id', 'contributors', 'in_reply_to_screen_name', 'place', 'retweeted', 'lang', 'truncated', 'geo', 'in_reply_to_user_id_str', 'favorite_count', 'filter_level', 'user', 'in_reply_to_status_id', 'source', 'created_at', 'retweet_count', 'is_quote_status', 'entities', 'favorited', 'id_str', 'coordinates', 'timestamp_ms',

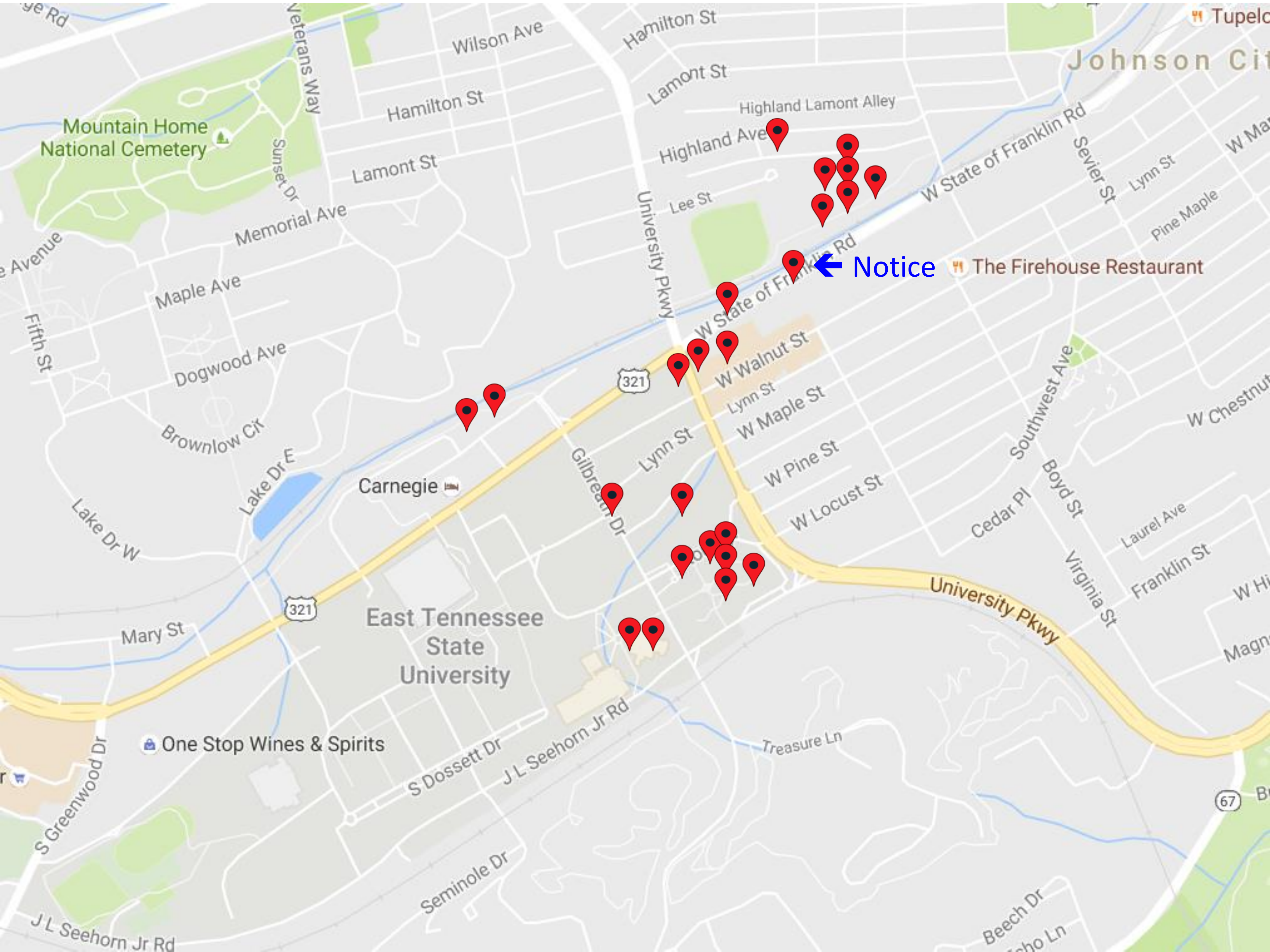
```
{'contributors': None,
 'coordinates': {'coordinates': [-87.2208409, 36.1075593], 'type':
 'created_at': 'Sun Oct 16 17:07:40 +0000 2016',
 'entities': {'hashtags': [{'indices': [45, 49], 'text': 'job'},
   {'indices': [88, 95], 'text': 'Retail'},
   {'indices': [96, 107], 'text': 'WHITEBLUFF'},
   {'indices': [112, 119], 'text': 'Hiring'},
   {'indices': [120, 130], 'text': 'CareerArc'}],
  'symbols': [],
  'urls': [{'display_url': 'bit.ly/2c6tco1',
    'expanded_url': 'http://bit.ly/2c6tco1',
    'indices': [64, 87],
    'url': 'https://t.co/r5Xw2lcB0d'}],
  'user_mentions': []},
 'favorite_count': 0,
 'favorited': False,
 'filter_level': 'low',
 'geo': {'coordinates': [36.1075593, -87.2208409], 'type': 'Point'}
 'id': 787701558070829061,
 'id_str': '787701558070829061',
 'in_reply_to_screen_name': None,
 'in_reply_to_status_id': None,
 'in_reply_to_status_id_str': None,
 'in_reply_to_user_id': None,
 'in_reply_to_user_id_str': None,
 'is_quote_status': False,
 'lang': 'en',
 'place': {'attributes': {},
  'bounding_box': {'coordinates': [[[-90.310298, 34.982924],
     [-90.310298, 36.678119],
     [-81.646901, 36.678119],
     [-81.646901, 34.982924]]],
   'type': 'Polygon'},
  'country': 'United States',
  'country_code': 'US',
  'full_name': 'Tennessee, USA',
  'id': '7f7d58e5229c6b6c',
  'name': 'Tennessee',
  'place_type': 'admin',
  'url': 'https://api.twitter.com/1.1/geo/id/7f7d58e5229c6b6c.json'
 'possibly_sensitive': False,
 'retweet_count': 0,
 'retweeted': False,
 'source': '<a href="http://www.tweetmyjobs.com" rel="nofollow">TweetMyJOBS</a>',
 'text': 'Join the Dollar General team! See our latest #job opening here: https://t.co/r5Xw2lcB0d #Retail #WHITEBLUFF, TN #Hiring #Ca
 'truncated': False,
 'user': {'contributors_enabled': False,
  'created_at': 'Tue May 12 15:26:12 +0000 2009',
  'default_profile': False,
  'default_profile_image': False,
  'description': 'Follow this account for geo-targeted Retail
  'favourites_count': 0,
  'follow_request_sent': None,
  'followers_count': 419,
  'following': None,
  'friends_count': 309,
  'geo_enabled': True,
  'id': 39521793,
  'id_str': '39521793',
  'is_translator': False,
  'lang': 'en',
  'listed_count': 116,
  'location': 'Nashville, TN',
  'name': 'TMJ-BNA Retail Jobs',
  'notifications': None,
  'profile_background_color': '253956',
  'profile_background_image_url': 'http://pbs.twimg.com/profi
  'profile_background_image_url_https': 'https://pbs.twimg.co
  'profile_background_tile': False,
  'profile_banner_url': 'https://pbs.twimg.com/profile_banner
  'profile_image_url': 'http://pbs.twimg.com/profile_images/6
  'profile_image_url_https': 'https://pbs.twimg.com/profile_i
  'profile_link_color': '4A913C',
  'profile_sidebar_border_color': '000000',
  'profile_sidebar_fill_color': '407DB0',
  'profile_text_color': '000000',
  'profile_use_background_image': True,
  'protected': False,
  'screen_name': 'tmj_bna_retail',
  'statuses_count': 449,
  'time_zone': 'Quito',
  'url': 'http://www.careerarc.com/job-seeker',
  'utc_offset': -18000,
  'verified': False}}
```


This is one tweet.

In 30 minutes, I collected 1000s of these from a small area around ETSU.

Twitter Data

- Easy to collect
  - Collected using python tweepy
  - Location based (used a box containing ETSU)
- Many features and many with many features
- Many ways of "grouping" the data
  - Grouping = Nearest Neighbor Network
    - By geo: Apartment or Dorm
    - By user: All tweets by same user
    - By time of day, By hashtag keyword, by @ keyword, …
  - Grouping leads to clustering…



Mountain Home National Cemetery
Wilson Ave
Hamilton St
Hamilton St
Lamont St
Lamont St
Highland Lamont Alley
Highland Ave
Lee St
Memorial Ave
Maple Ave
Dogwood Ave
Brownlow Cir
Fifth St
Veterans Way
Sunset Dr
University Pkwy
W State of Franklin Rd
W State of Franklin Rd
Notice
The Firehouse Restaurant
Johnson Cit
Tupelo
Sevier St
Lynn St
Pine Maple
W Walnut St
Lynn St
W Maple St
W Pine St
W Locust St
Lynn St
Gilbreath Dr
Southwest Ave
W Chestnut
Cedar Pl
Boyd St
Laurel Ave
Franklin St
Virginia St
University Pkwy
321
321
Carnegie
Lake Dr E
Lake Dr W
Mary St
East Tennessee State University
One Stop Wines & Spirits
S Greenwood Dr
S Dossett Dr
J L Seehorn Jr Rd
Treasure Ln
Seminole Dr
J L Seehorn Jr Rd
Beech Dr
67

Twitter Data

- Easy to collect
  - Collected using python tweepy
  - Location based (used a box containing ETSU)
- Many features and many with many features
- Many ways of “grouping” the data
  - Grouping = Nearest Neighbor Network
  - Grouping leads to clustering…
- Many questions that can be addressed
  - “Happiest” time of day, Academic versus social tweets
  - Words associated with “angry” tweets

How to Work With “Big Data”

- Our goal is to understand the *process* that produces a data set (data is only a tool for doing so)
  - Valid conclusions are those that remain true even if another data set were to be sampled from that process
  - Ultimate Goal: Identify, Explore, and Understand the *topology* (= underlying structure) of a process
  - Example: Tweet data is about users, not tweets
- Two issues we must continuously address
  - **Bias:** The degree to which *sample averages* converges to something other than *population averages*
  - **Overfitting:** The degree to which results are only true for a data set and not of the process which produced it

Data is Unstructured

- We are going to structure it for this presentation (but not necessary – see note very soon!)
  - We think of a tweet in terms of its **text** field
  - “Across the yard. [Cam 1] on Sunday, October 16, 2016 @ 2:05:11 PM #CarolinaWx #ClaytonNC”
  - “Shirts selling like crazy at Woolly Worm Festival this weekend in Banner Elk, NC. Want one?”
  - “Mom and Dad's dog Ruby says it's Sunday afternoon ... nap time.”
- Let’s vectorize the text fields…

Vectorizing Text Data

- Remove “stop” words (the, a, an, and, or, …)
- Remaining words become features
- Each tweet has count of # of occurrences of that word

Enormously Long!!

	Sunday	Worm	Nap	Crazy	Etsu
Observation 1	1	0	0	0	0
Observation 2	0	1	0	1	0
Observation 3	1	0	1	0	0

But most entries are zeros

Vectorizing Text Data

- Remove "stop" words (the, a, an, and, or, …)
- F
- E
  t

Unstructured data can always be reduced to key-value pairs, which can always be considered to be sparse representations of tensors.

Observation 2	0	1	0	1	0
Observation 3	1	0	1	0	0

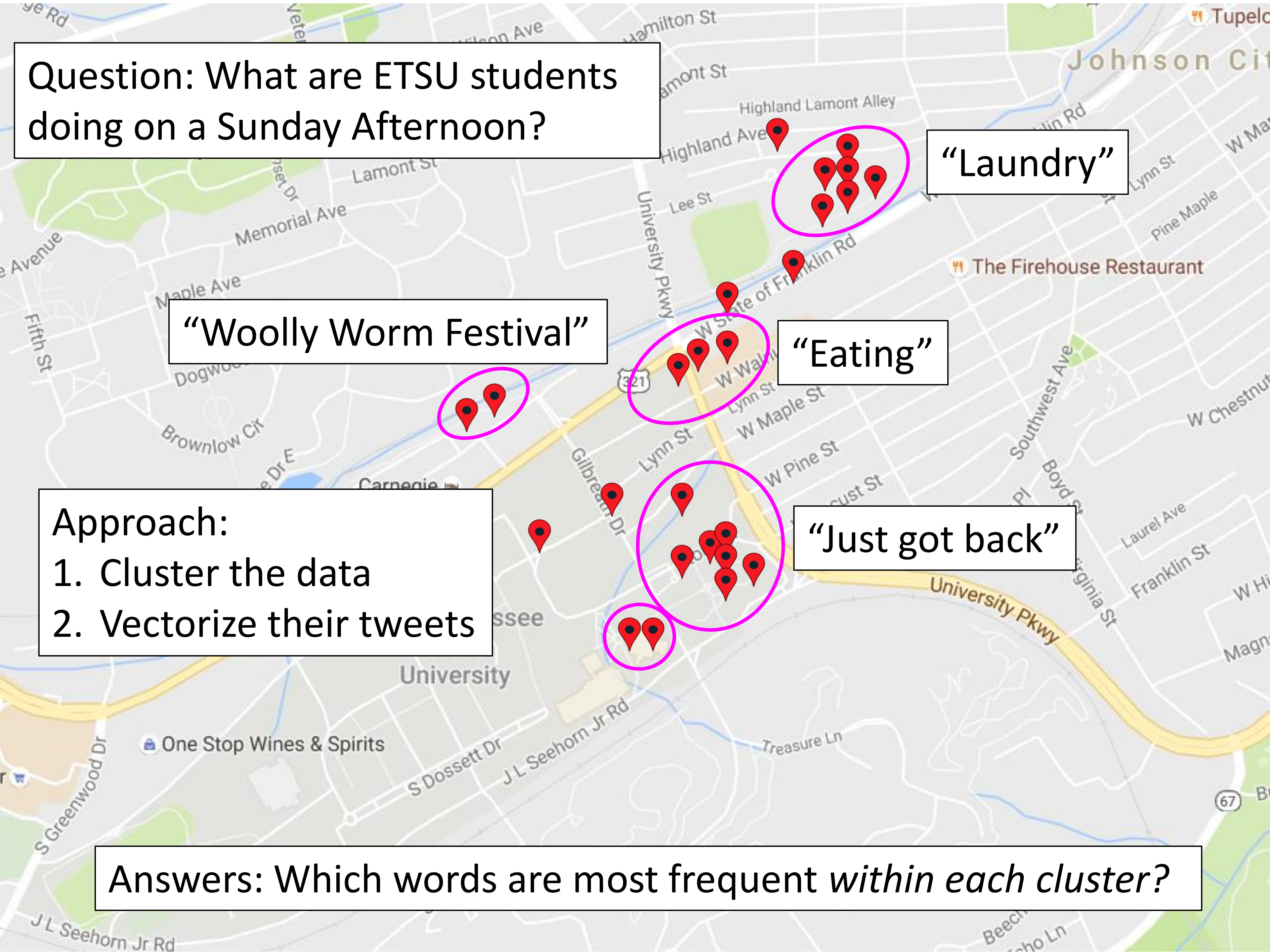
- Sparse representation – only store nonzeros
  - Data of the form ([row,column], value)
  - If no row/column entry, assume value is 0

Key, value pair

Recommender System

- Result is a *recommender system*
  - Each word has a rating (count) for each tweet, with a rating of 0 if word does not occur in **text** field
  - Also known as *collaborative filtering*
- Results produced by identifying groups (clusters) with similar ratings profiles
  - Needed: Measure of Similarity (e.g., cosine)
  - Typically, more sophisticated measures needed
- Nearest Neighbor Graph: Two observations are deemed close based on similarity measure

Question: What are ETSU students doing on a Sunday Afternoon?



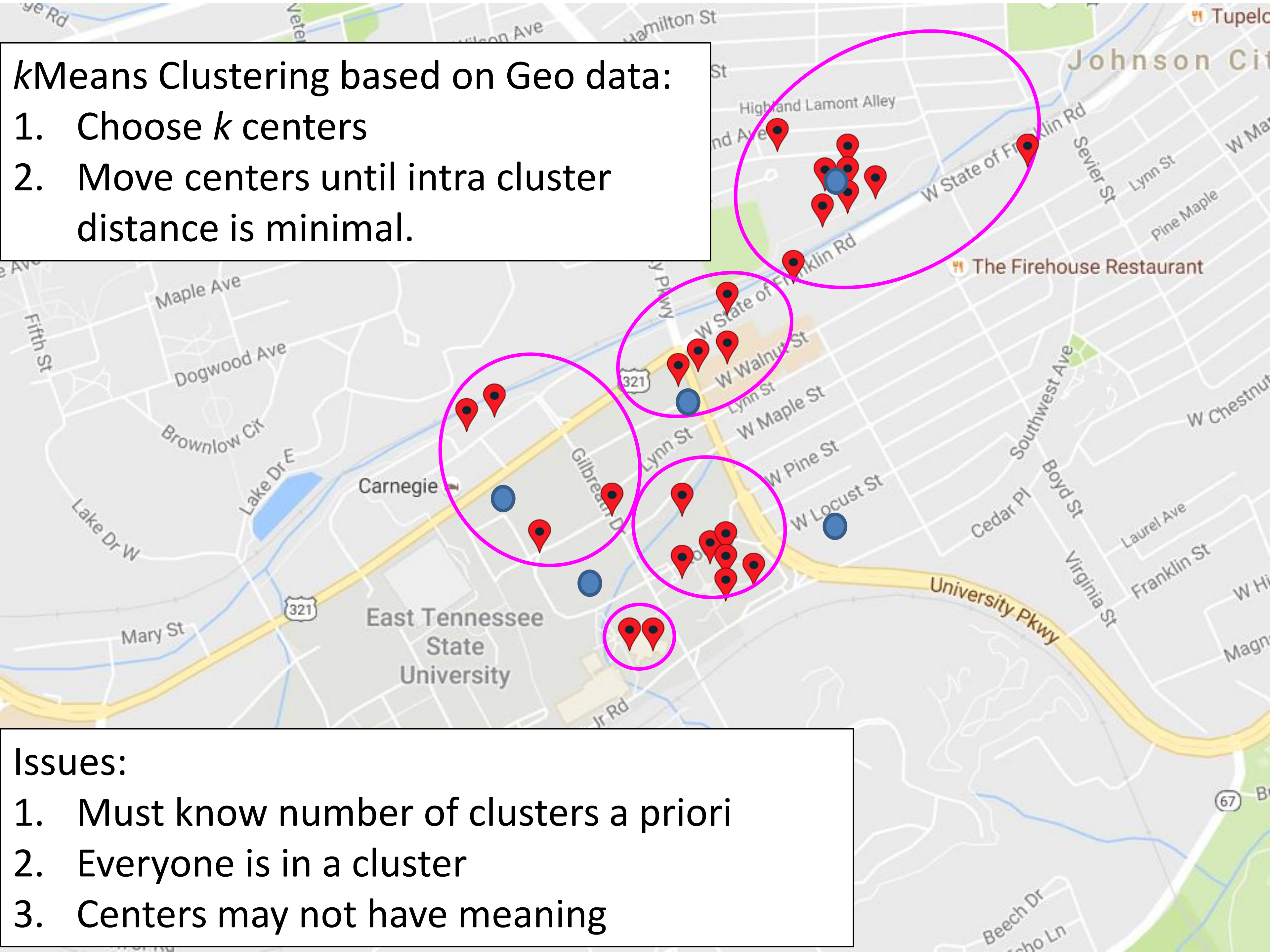
Approach:

1. Cluster the data
2. Vectorize their tweets

Answers: Which words are most frequent *within each cluster?*

*k*Means Clustering based on Geo data:

1. Choose *k* centers
2. Move centers until intra cluster distance is minimal.



Issues:

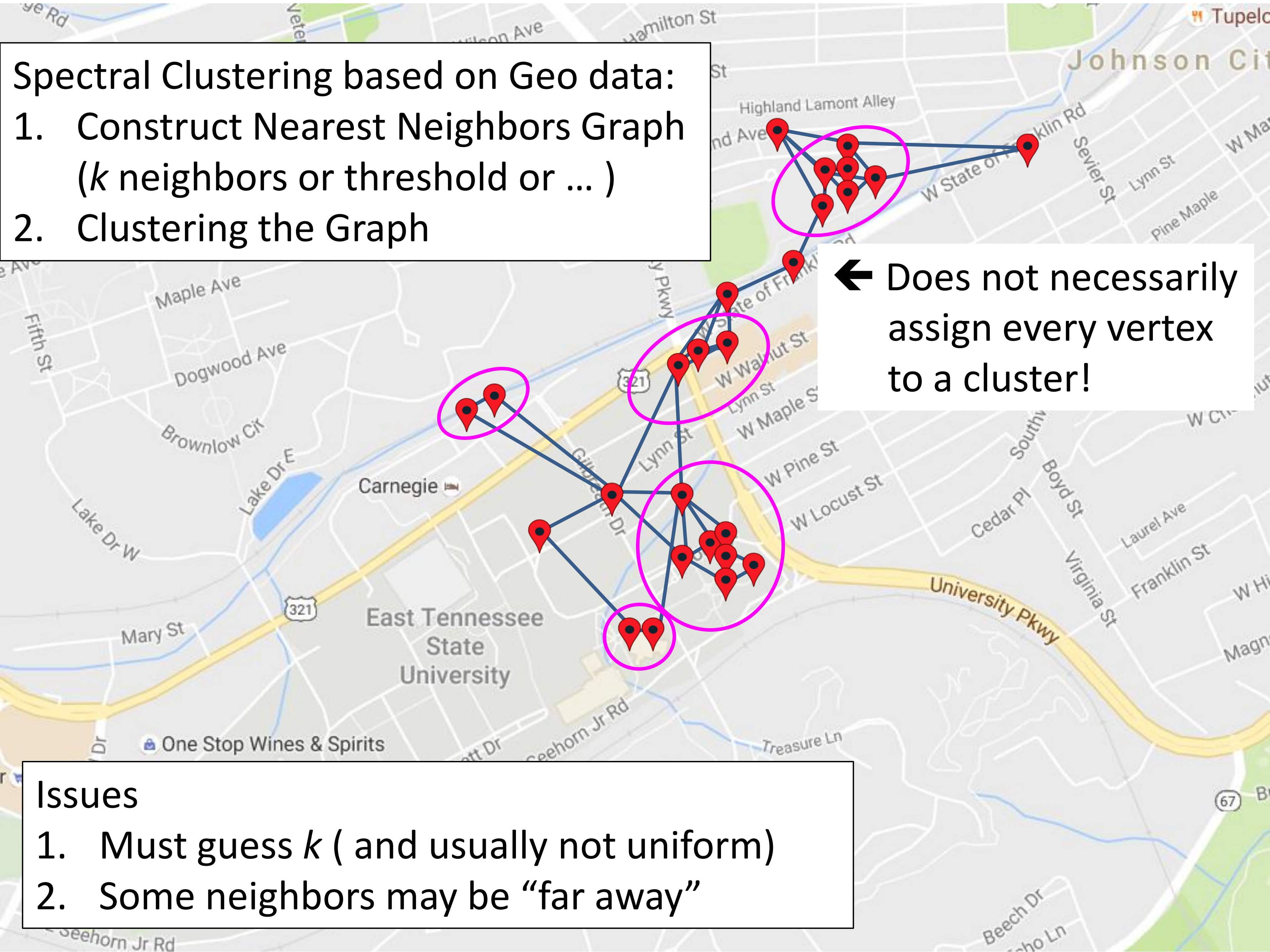
1. Must know number of clusters a priori
2. Everyone is in a cluster
3. Centers may not have meaning

Nearest Neighbor Graphs

- Basis for all machine learning
  - Simple to use, and applicable in any situation
  - Are the theoretical “structure” produced by high power, sophisticated algorithms (e.g., Random Forests)
- Classification of unlabeled observation(s)
  - Construct *k* nearest neighbors graph
  - Predict classification as majority vote of neighbors
- Regression: Predict value as statistic on neighbors
- Imputing Missing Values: as statistic on neighbors

Spectral Clustering based on Geo data:

1. Construct Nearest Neighbors Graph (*k* neighbors or threshold or …)
2. Clustering the Graph



← Does not necessarily assign every vertex to a cluster!

Issues

1. Must guess *k* (and usually not uniform)
2. Some neighbors may be “far away”

Two Possible Outcomes

- Explore
- Preproce
- Data trains assifi at works perfe
  - Often via a “S gorithm”
  - Random fores ral Net
  - “Black Box”
  - *What abo pr that produce data? oal is to u and the p*

- Explore
- Preprocess
- Repeated Refinements
  - Select/Modify Features
  - Train the Algorithm
  - Apply Metric(s)
- Predict, interpret, visualize, etcetera, …

Nearest Neighbor Graphs

- *If only we knew precisely what method to use for similarity and exactly how many neighbors to use for each node…*
  - Example: Similarity for predicting age of a passenger on the Titanic if their age is unknown
  -
```
def TitanicNearness(x,y):
    if x.Pclass == y.Pclass and x.SibSp == y.SibSp and x.Parch == y.Parch:
        return abs( x.Fare - y.Fare )
    elif( x.Pclass == y.Pclass and x.Parch == y.Parch and x.Sex == y.Sex):
        return 1000  ##  Some similarity here, so not infinity yet
    else:
        return inf   ## No reason to compare ==>  infinitely far apart
```
  - Obtained by repeated refinement based on metrics
  - Developed on training data, refined on validation data
  - Scored on testing data
- *Because not all Features are created equal…*

Feature Engineering

- Features (like words in a tweet) are the target
  - **Feature Selection:** Only need a subset of the features
  - **Dimensionality Reduction:** lower dimensional info (e.g., faces) in higher dimensional data (e.g., images)
  - **Manifold Reconstruction:** Geometric nature (*topology*) of the process is inferred from the structure

	Feature 1	Feature 2	Feature 3	…	Feature n
Observation 1	#	#	#	…	#
Observation 2	#	#	#	…	#
⋮	⋮	⋮	⋮	⋱	⋮
Observation *m*	#	#	#	…	#

Feature Engineering

- Suppose we consider the data to be a *matrix*

	Feature 1	Feature 2	Feature 3	…	Feature n
Observation 1	#	#	#	…	#
Observation 2	#	#	#	…	#
⋮	⋮	⋮	⋮	⋱	⋮
Observation *m*	#	#	#	…	#

$$X = [X_1, \ldots, X_n] = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

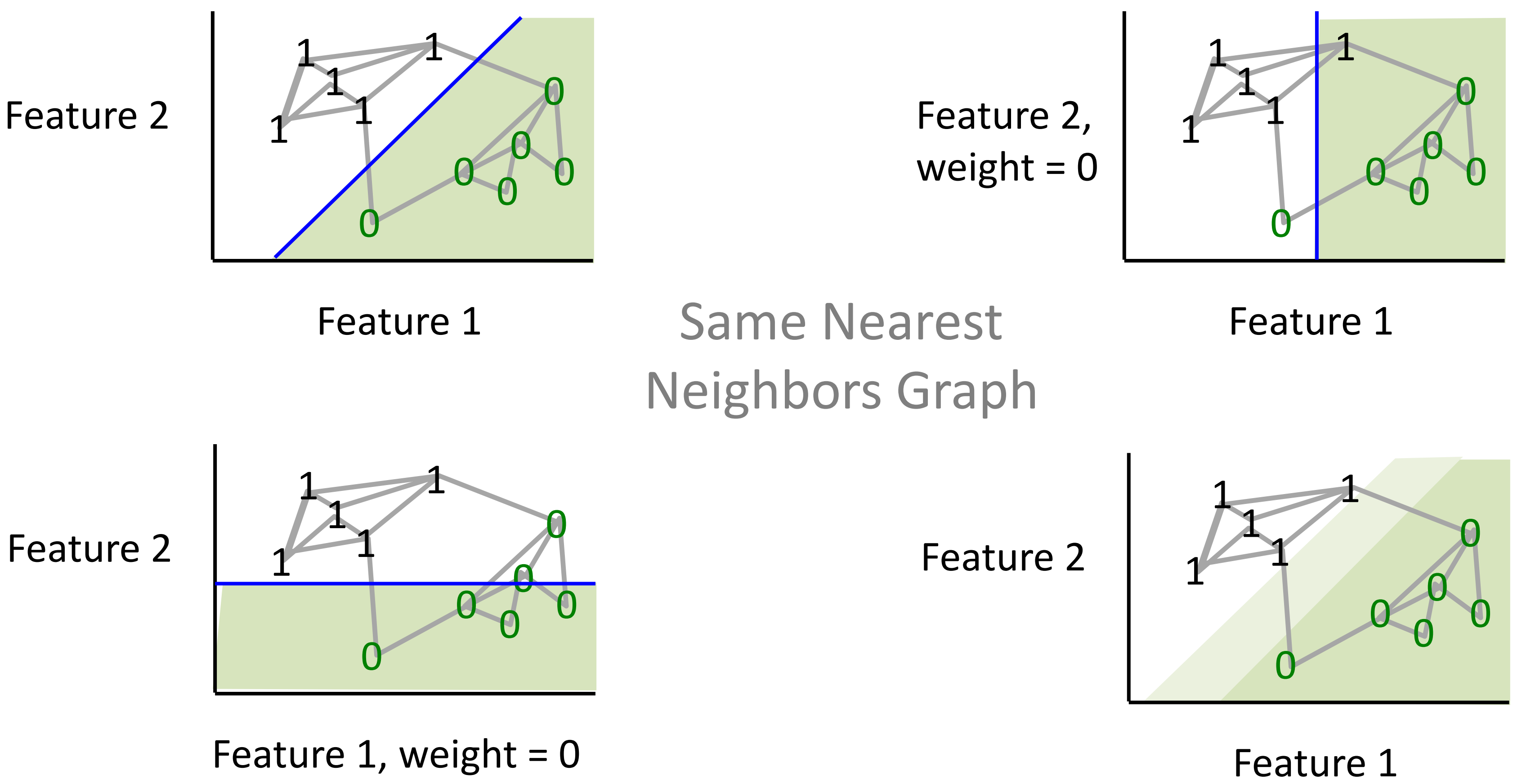
Feature Weights as Weighted data

$$W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

$$X_w = WX = [w_1 X_1, w_2 X_2, \ldots, w_n X_n]$$

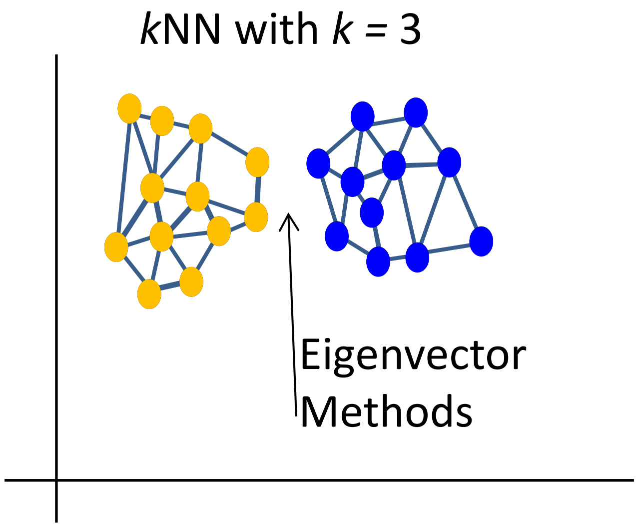
Feature Engineering

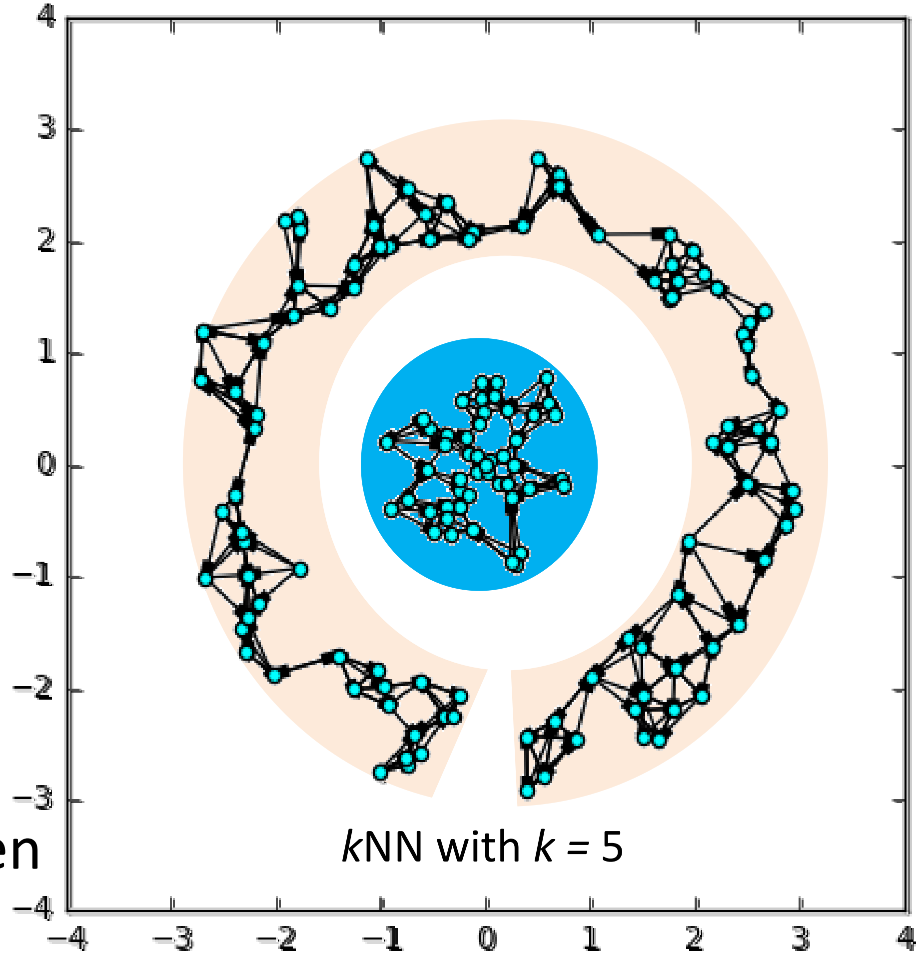
- “Powerful Algorithms” often related to “linearly separable” in some high dim representation



Graph Theory “learns” the underlying *topology* of the process the data comes from

Spectral Clustering = kNN + Eigenvector Methods

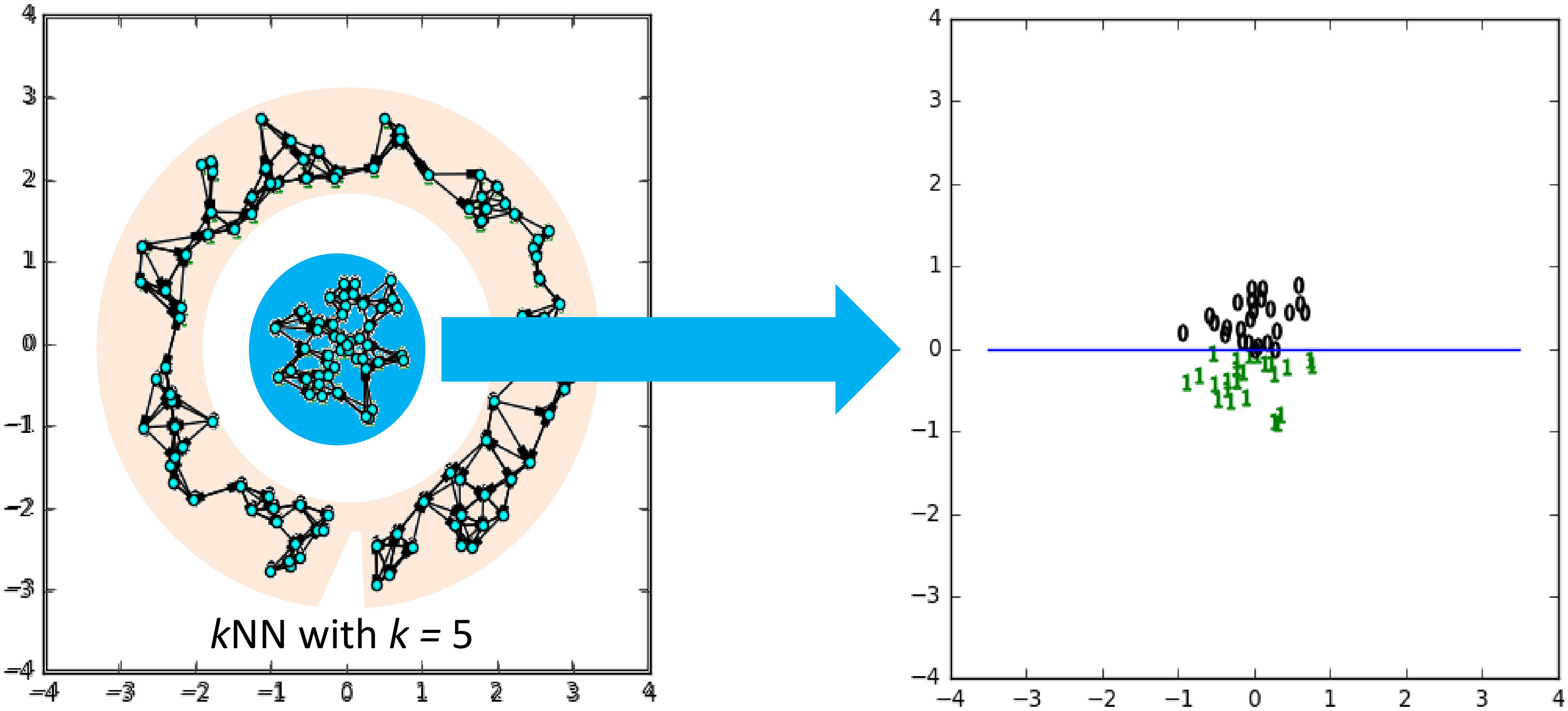




Clusters are not just “bunches”.

Key is similarity within clusters being higher than similarity between

Classifiers, Regressors, Linear Separability, etc. are based on the process' topology



Feature Engineering = "digging" (yes, it's hard work!) deeper and deeper into the topology

Example: Twitter Data

- Digging deeper means for example

Our goal is not “Black Box” classifier/regressor perfection, nor is it a collection of charts and tables with statistical outcomes and diagrams.

Our goal is to “go deeper and deeper” until we have a model for the process itself that produces such data.

Feature engineering is “how we go deeper and deeper.”

story, and we keep going until we know the story

- Often “many stories” in a set of “Big Data”

So how do we do that?

- We can always refine our models
  - Make changes, assess with metrics
    - Based on improved understanding from previous model
    - “Every big data problem is unique” (and ultimately, personal)
  - Combine into “bigger models” (ensembles)
  - “In machine learning, the best model is all of them.”
- We can always refine our graphs
  - Similar to above, but not the same
  - Relies on centuries of mathematics – graph theory, information theory, signal processing, Fourier analysis
  - And especially, lots and lots and lots of linear algebra

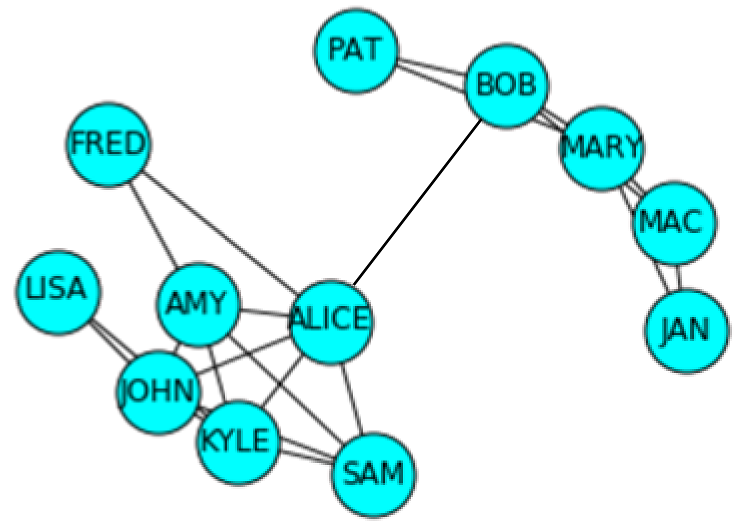
Lots and Lots of Linear Algebra

- Example: A simple 2 component graph
- Laplacian Matrix:
  $L = D - A$

  Degree Matrix (diagonal)

  Adjacency Matrix (symmetric)

  - 0 is an eigenvalue with multiplicity 2
  - Eigenvectors:



BOB JAN JOHN

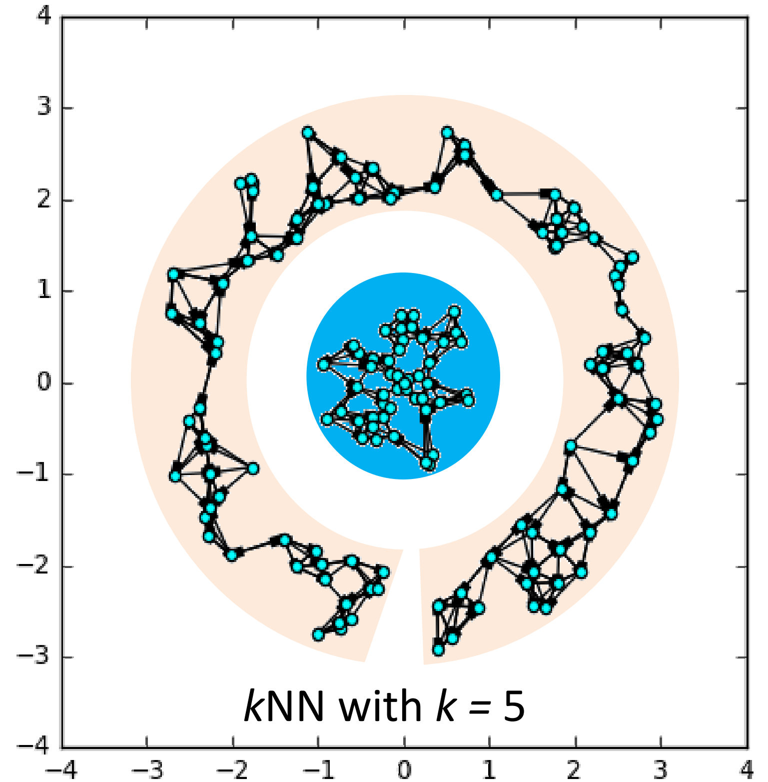
$[1, 1, \ldots, 1]^T$ and $[1, -1, 1, \ldots, 1, -1]^T$ (Fiedler Eigenvector)

FRED MARY

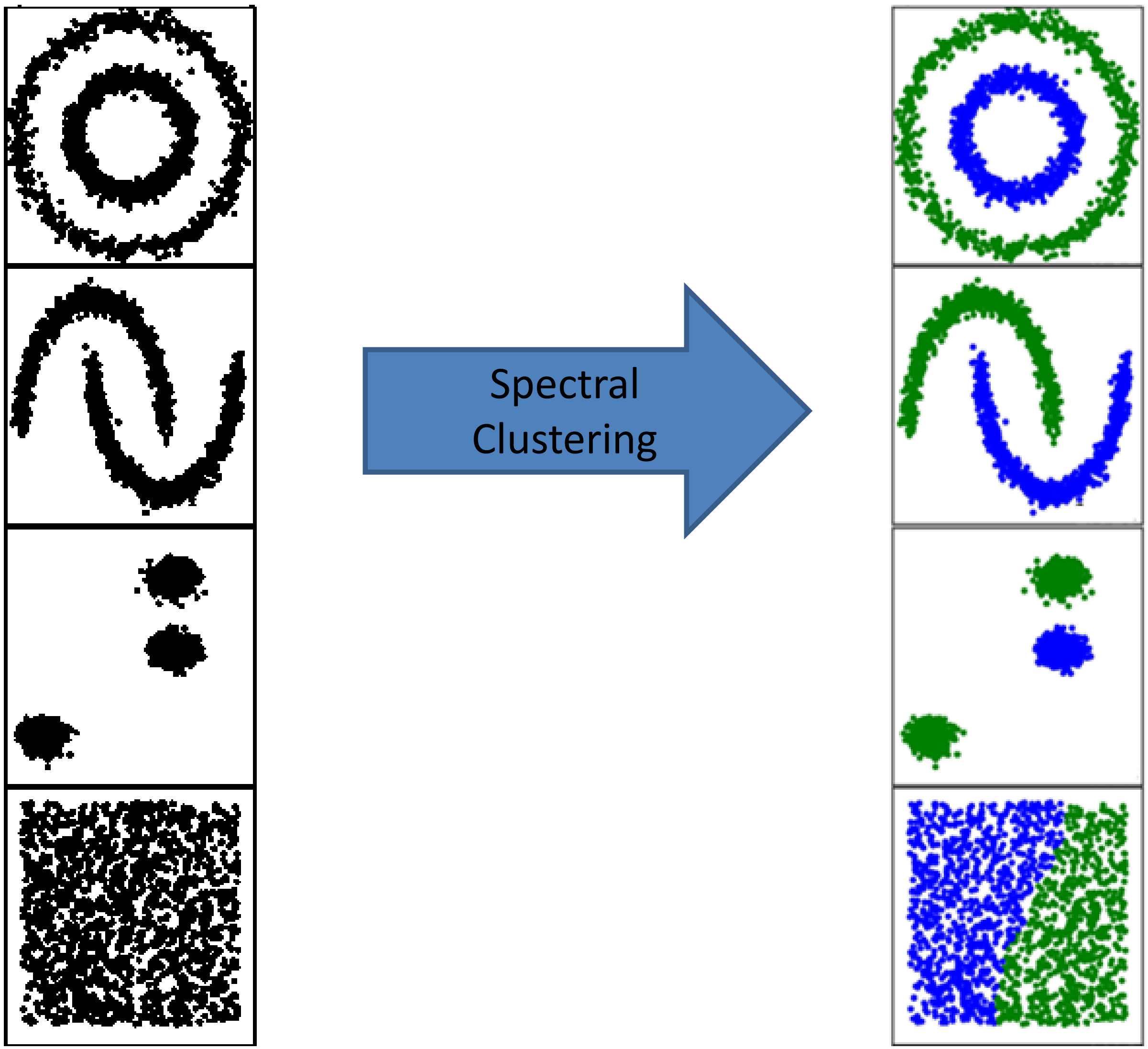
- Suppose Alice “Friends” Bob
  - A 1 component graph with 2 *clusters*
  - Fiedler Eigenvector signs remain the same

Spectral Clustering

- Given *m* observations of *n* features
  - Infer a graph for the *m* observations
  - Construct the Laplacian matrix for the graph
  - Use eigenvectors to cluster the *m* observations
- Example: (from sklearn)
  - Points in the plane with features as xy coords
  - For simplicity, only two clusters
  - Thus, clusters obtained from the Fiedler eigenvector



Spectral Clustering: Fiedler Eigenvector

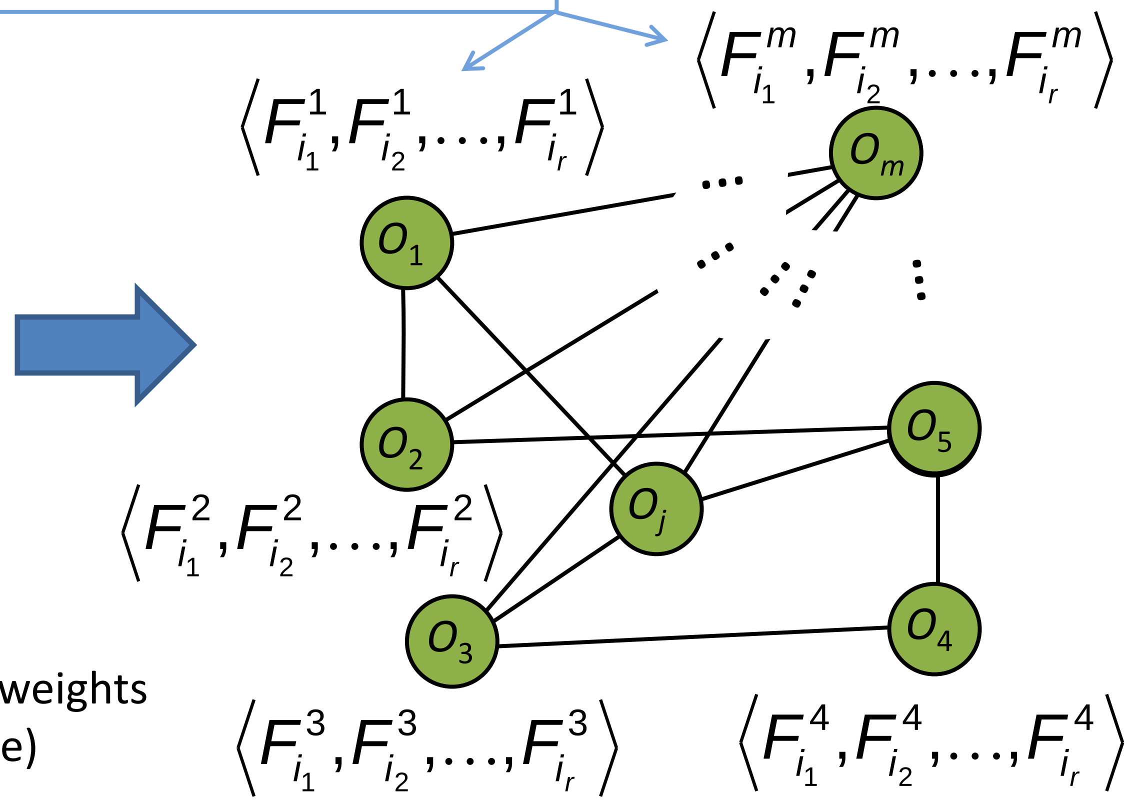


More than 2 clusters requires the use of more eigenvectors

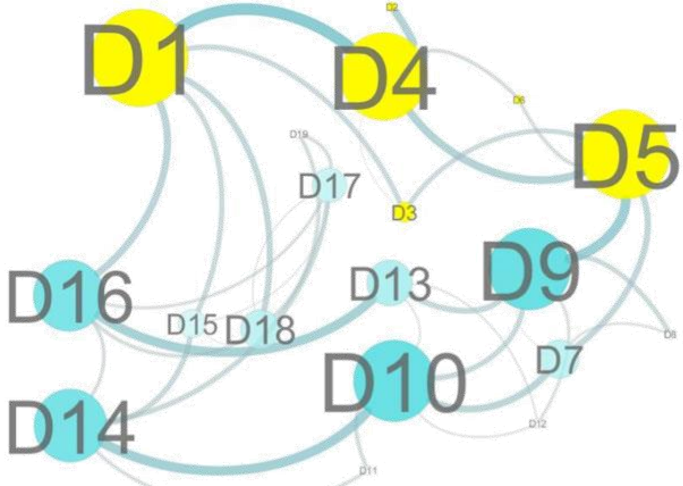
Vertex Weights

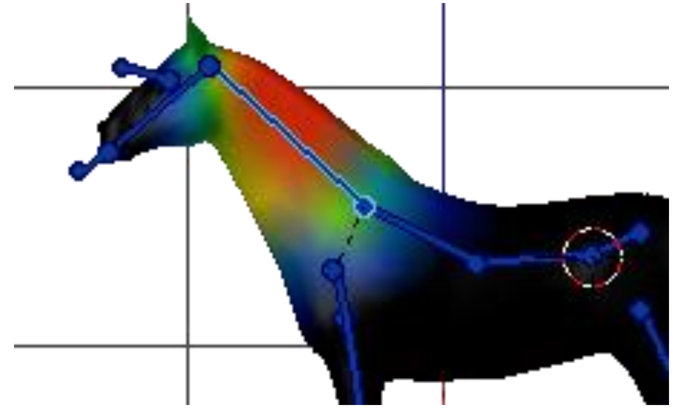
Some features used to determine edges
Remaining features become vertex weight vectors

	F_1	F_2	…	F_n
O_1	#	#	…	#
O_2	#	#	…	#
⋮	⋮	⋮	⋱	⋮
O_m	#	#	…	#

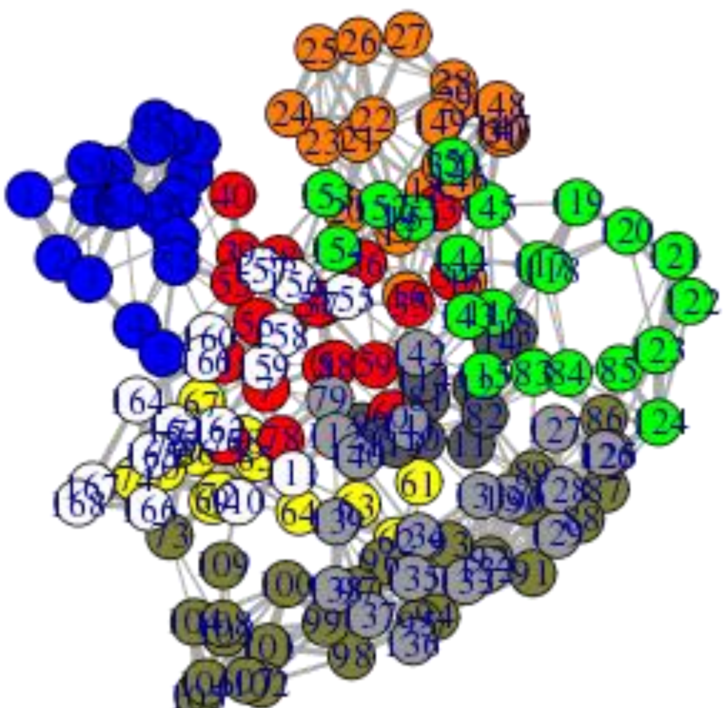


Vertex weights can generate edge weights
(via dot products, for example)
-- or –
can be used after clustering to generate
weights for clusters as vertices

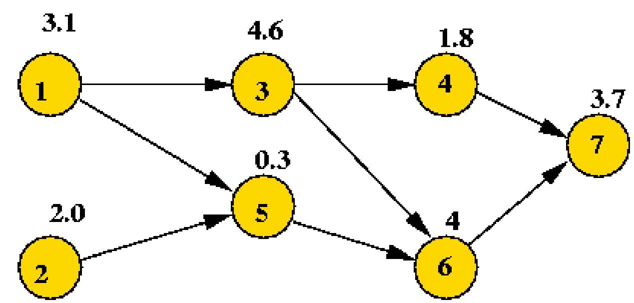




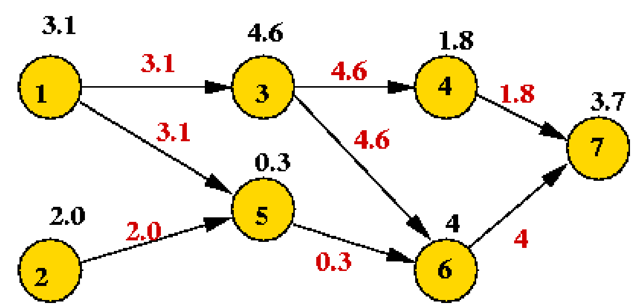
VERTEX WEIGHTED GRAPHS



- Consider the case where vertices have weights, but the edges don't, e.g.,



- To find shortest paths, apply a vertex's weight to each outgoing edge:



- Then solve as a regular DAG-SPT problem.

Rigorous Math (a bit early – Sorry!!)

- Spectral Graph Theory: If $G = (V,E)$ is a graph with vertex set V *and* edge set E, then
  - $W(G) = \{ f: V \to \mathbb{R} \}$ is a $|V|$ dim vector space
  - The Laplacian satisfies $f^T L\mathrm{f} = \sum_{(u,v)\in E} |f(u) - f(v)|^2$
- Vertex weighted graph theory: Let $m = |V|$
  - $W_v(G) = \{ f: V \to \mathbb{R}^n \}$ is an mn dim vector space
  - Inner product: $\langle f, g \rangle = \sum_{v\in V} f(v) \cdot g(v)$ where $\cdot$ is dot product on $\mathbb{R}^n$
  - The vertex weighted Laplacian satisfies

$$f^T L_v \mathrm{f} = \sum_{(u,v)\in E} \left\| f(u) - f(v) \right\|^2$$

Vertex Weighted Spectral Clustering Same Procedure – But with Vertex Weights

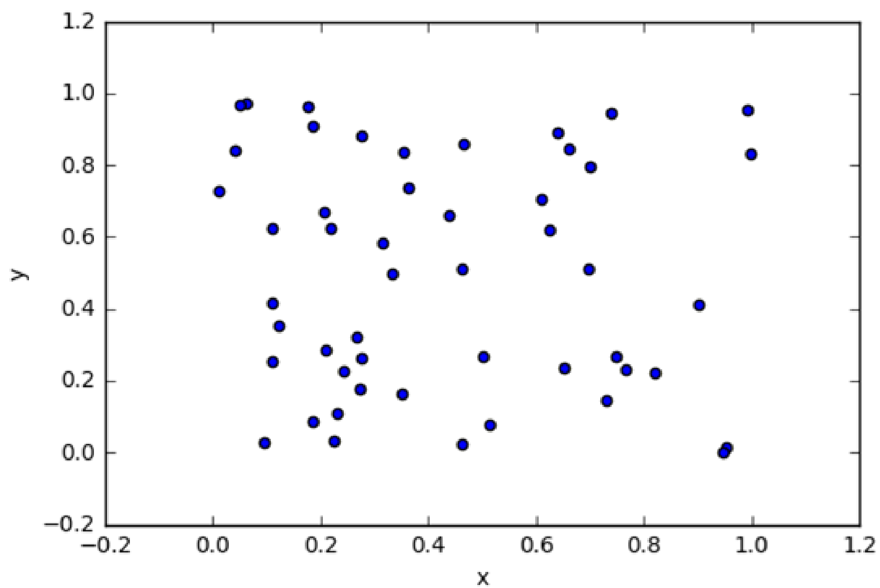
- Given *m* observations of *n* features
  - Infer a graph using $r < n$ of the features (e.g., geodata)
  - Use the $n - r$ remaining features as vertex weights
  - Construct the vertex weighted Laplacian matrix L_v
  - Use eigenvectors to cluster the *m* observations
- Notes:
  - Often feature partition is via a transformation
  - Does not "throw out" data, but does reduce dimension
  - Fielder vector is actually a "vector of vectors"

Example: Tweet-Like Data

	x	y	ETSU	UTK	Milligan	King
0	0.749094	0.268641	0	1	0	0
1	0.272644	0.176196	1	1	0	1
2	0.093655	0.029612	2	0	0	0
3	0.111131	0.254317	1	0	0	1
4	0.624397	0.619768	0	1	1	0

- Doesn't make sense to use same similarity measure on geo and word counts
- no clusters: *x,y* are uniformly distributed
- Vertex Weights:

	ETSU	UTK	Milligan	King
0	0	1	0	0
1	1	1	0	1
2	2	0	0	0
3	1	0	0	1
4	0	1	1	0



The Fielder Eigen"Vector"

- Is a Vector of Vectors:

$$f = \langle f_1, f_2, \ldots, f_m \rangle \text{ where } f_i = \begin{array}{cl} \begin{bmatrix} a_{i1} \\ a_{i2} \\ a_{i3} \\ a_{i4} \end{bmatrix} & \begin{array}{l} \text{ETSU} \\ \text{UTK} \\ \text{Milligan} \\ \text{King} \end{array} \end{array}$$

- Generalize "sign" clustering via inner product
  - $f_i \cdot f_j > 0$ ➔ vertex *i* and vertex *j* in same cluster
  - $f_i \cdot f_j > 0$ ➔ vertex *i* and vertex *j* in opposite clusters
  - $f_i \cdot f_j \approx 0$ ➔ no information (unassociated)
- Also can dot with a *reference vector*

Application to Data

- **Dimensionality Reduction:** Via PCA, Singular values are eigenvalues of $X_w X_w^T$
  - Squared singular values sum to norm
  - SVD: $X_w = U\Sigma V^T \; where \, \Sigma = diag\left(s_1, \ldots, s_k, s_{k+1}, \ldots, s_r\right)$
  - Use instead $X_w = U\Sigma' V^T \; where \, \Sigma' = diag\left(s_1, \ldots, s_k, 0, \ldots, 0\right)$
- What about the data corresponding to $s_{k+1}, \ldots, s_r$?
  - Another approach to dimensionality reduction is to use Laplacian of a nearest neighbors graph
  - In which case, the data "left over" becomes the vertex weights for the resulting graph

Application to Data

- **Dimensionality Reduction:** Find SVD of the Laplacian *L* of the nearest neighbor graph of X_w
  - SVD: $L = U\Sigma U^T \ where \Sigma = diag\left(s_1, \ldots, s_k, s_{k+1}, \ldots, s_m\right)$
  - Each u_j in *U* corresponds to an s_j and is length *m*
  - Replace observation *p* with $\left\langle u_1\left(p\right), \ldots, u_k\left(p\right)\right\rangle$
- What about the data corresponding to $s_{k+1},\ldots,s_r$?
  - Vertex weights of observation (= vertex) *p* are subsequently $\left\langle u_{k+1}\left(p\right), \ldots, u_m\left(p\right)\right\rangle$
  - Ignore vertex weights to get a standard approach

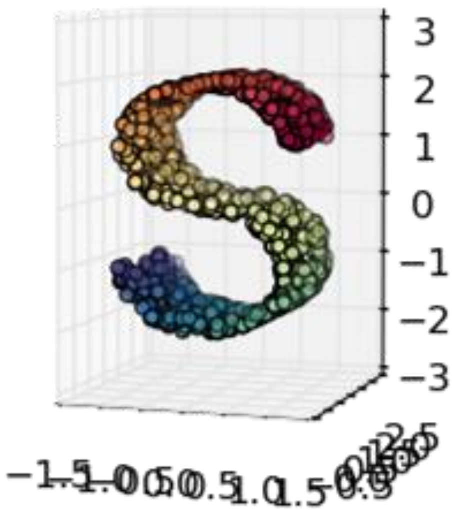
Advantages and Disadvantages

- Advantages
  - No data is “thrown out”.
  - Topological properties of the data are preserved
  - And can be refined using the vertex weights
  - *Can do no worse than a standard approach!*
- Disadvantages
  - Speed: Requires computation of entire SVD
  - Speed: Yes, it really is the only problem, and it is also a very big problem!

Quick Insight: Manifold Learning

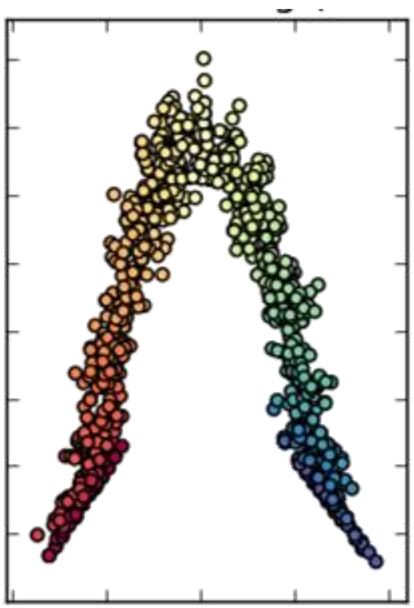
- Idea is that dimensionality is reduced because the *topology* is a lower dimensional manifold embedded in the high dimensional data
  - Let $f_1, f_2, \ldots, f_k, \ldots, f_m$ be eigenvectors of the Laplacian of the nearest neighbors graph
  - Topology of the data may only require a space with basis $f_1, f_2, \ldots, f_k$ for $k < m$
- And in particular, this same idea applies when using *tensors* as our vector space

Sklearn: Manifold Learning



1000 points
10 neighbors
per point

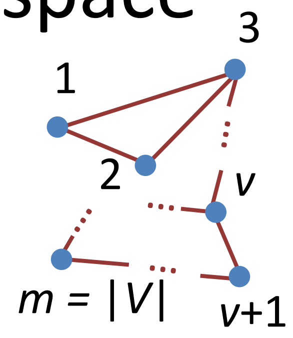
Laplacian matrix
Generated 2d
representation



Laplacian Matrix can be used to preserve the *topology* of a manifold represented by a combinatorial graph

Tensors and Spectral Clustering

- $W_v(G) = \{ f: V \to \mathbb{R}^n \}$ is an *mn* dim vector space
  - $f \in W_v(G)$ is $f = < f(v,r) >$, $v \in V$, $r=1,\ldots,n$
  - If $g = < g(v) > \in W(G)$ and $h = < h(r)> \in \mathbb{R}^n$, then $g \otimes h = < g(v)h(r) >$ (tensor product)
  - If $e_1,\ldots,e_m$ is an onb for $W(G)$ and $b_1,\ldots,b_n$ is an onb for $\mathbb{R}^n$, then $e_i \otimes b_j$ is an onb for $W_v(G)$



- If $T : V \to V$ is linear with matrix $[\ T(u,v)\ ]$, then define $T_v: W_v(G) \to W_v(G)$ by

$$T_v = \left[T(u,v) I_n \right]$$

- Everything "lifts" to $W_v(G)$ in this way

Final Comments

- Can also define $W_v(G) = \{ f: V \to \mathcal{B} \}$ for an arbitrary Banach Space *B.*
  - Deep Learning: Signal Processing combined with Machine Learning, usually via neural networks
  - Neural Network is a Multi-scale Nearest Neighbors algorithm which “learns” at its vertices
  - Deep Learning: Signal Processing on the vertex weights (*B*) with neural network on the graph *V*
- And apply to twitter data
- Successively refining and improving as we do

Thank You!

Any Questions?

```python
import time
import tweepy
from tweepy import OAuthHandler

from tweepy import Stream
from tweepy.streaming import StreamListener
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [5]:

```python
consumer_key = '[redacted]'
consumer_secret = '[redacted]'
access_token = '[redacted]'
access_secret = '[redacted]'

auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

api = tweepy.API(auth)
```

In [11]:

```python
class MyListener(StreamListener):

    def __init__(self):
        self.start_time = time.time()

    def on_data(self, data):
        if( time.time() - self.start_time ) >= 1800:
            return False
        try:
            with open('ETSUexperiment2.json', 'a') as f:
                f.write(data)
                return True
        except BaseException as e:
            print("Error on_data: %s" % str(e) )
        return True

    def on_error(self, status):
        print(status)
        return True

twitter_stream = Stream( auth, MyListener() )
twitter_stream.filter( track=['etsu'], locations = [ -82.380371, 36.297655,  -82.347723, 36.315562  ] )
```

In [37]:
```
%pylab inline

import json
import pandas as pd
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [38]:
```
tweets_data = []
with open("ETSUexperiment2.json", "r") as tweets_file:
    for line in tweets_file:
        try:
            tweet = json.loads(line)
            tweets_data.append(tweet)
        except:
            continue
```

In [47]:
```
tweets_data[90]['text']
```

Out[47]:
```
"These guys are writing their history ...@chaseelliott &amp; @austindillon3 are chasin' for
 this thing ya'll Don't count them out #NASCAR"
```

In [48]:
```
tweets_data[90]
```

Out[48]:

```
{'contributors': None,
 'coordinates': None,
 'created_at': 'Sun Oct 16 18:04:02 +0000 2016',
 'entities': {'hashtags': [{'indices': [129, 136], 'text': 'NASCAR'}],
  'symbols': [],
  'urls': [],
  'user_mentions': [{'id': 42900685,
    'id_str': '42900685',
    'indices': [40, 53],
    'name': 'Chase Elliott',
    'screen_name': 'chaseelliott'},
```