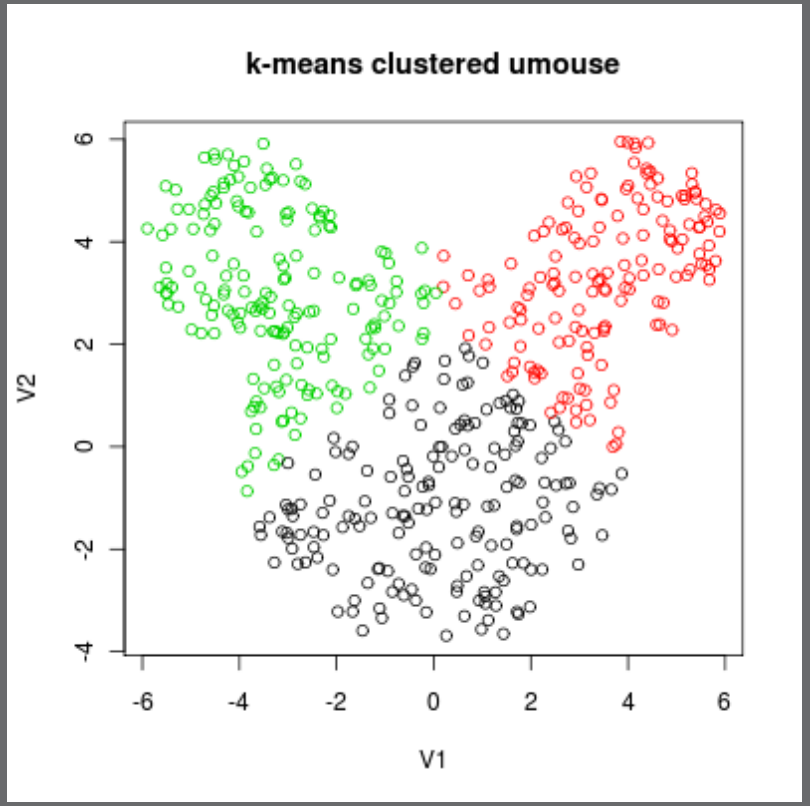
K-Means Clustering

Elliot Cohen
Taryn Heilman
Morning Lecture - Dec. 6, 2017

galvanize



Learning Objectives

galvanize

- Introduce unsupervised learning, compare to supervised
- Introduce clustering concept
- Enumerate clustering use-cases
- Define K-means algorithm:
  - How to code
  - Centroid initialization
  - Stopping Criteria
  - Evaluating the algorithm (metrics)
  - How to choose k

Review: Queue the Questionator

galvanize

- What does supervised learning mean? Give an example
- Describe the difference between parametric and non-parametric learners, and give an example of each
- How do we measure the “success” of a supervised learning algorithm? Give examples for classification and regression
- Describe the euclidean distance metric. Name two additional distance metrics we have discussed and describe.
- What is the curse of dimensionality?

Supervised vs. Unsupervised Learning

galvanize

Supervised

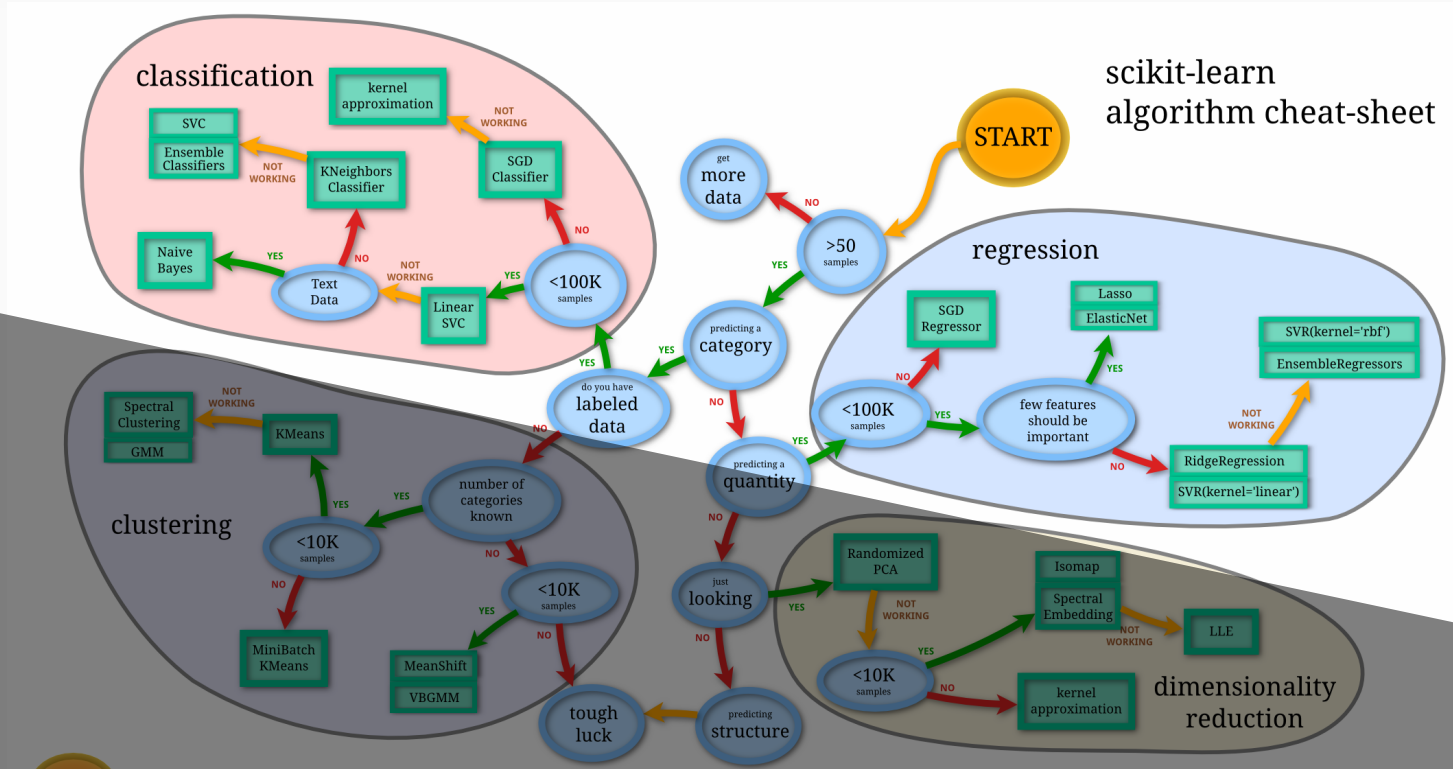
- Have a target/label that we model
- Models look like functions that take in features (X) and predict a label (y)
- Have an error metric that we can use to compare models
- Used to predict future unlabeled data

Unsupervised

- No target/label to predict
- Goal is to find underlying structure, patterns, or organization in data
- No stark error metric to compare models - determining if you have the optimal solution is very challenging. Cross validation often not applicable.

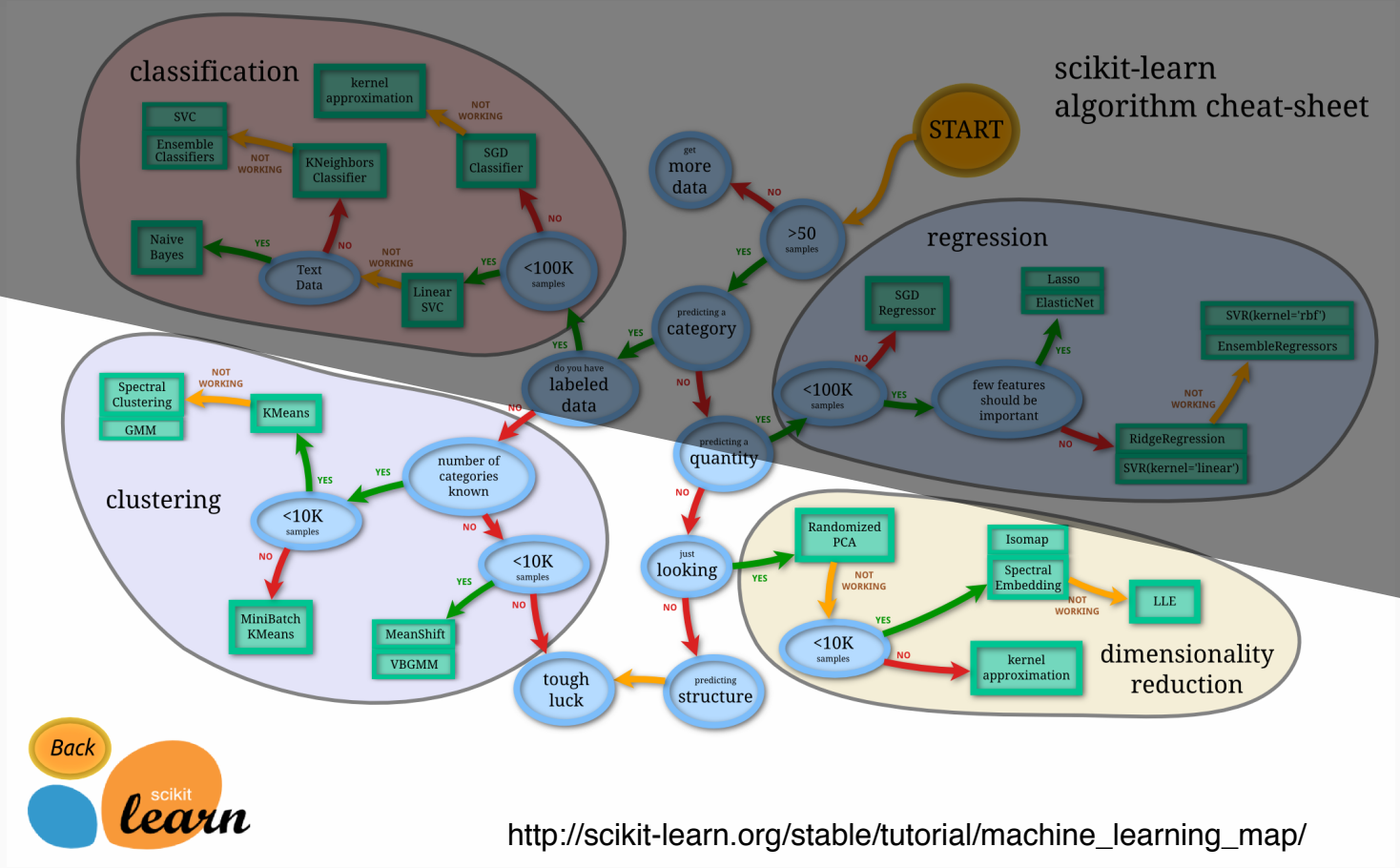
Where We've Been - Supervised Learning

galvanize



Unsupervised Learning - Where We are Going!

galvanize



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

Basic K-Means Algorithm

galvanize

Initialize k centroids*

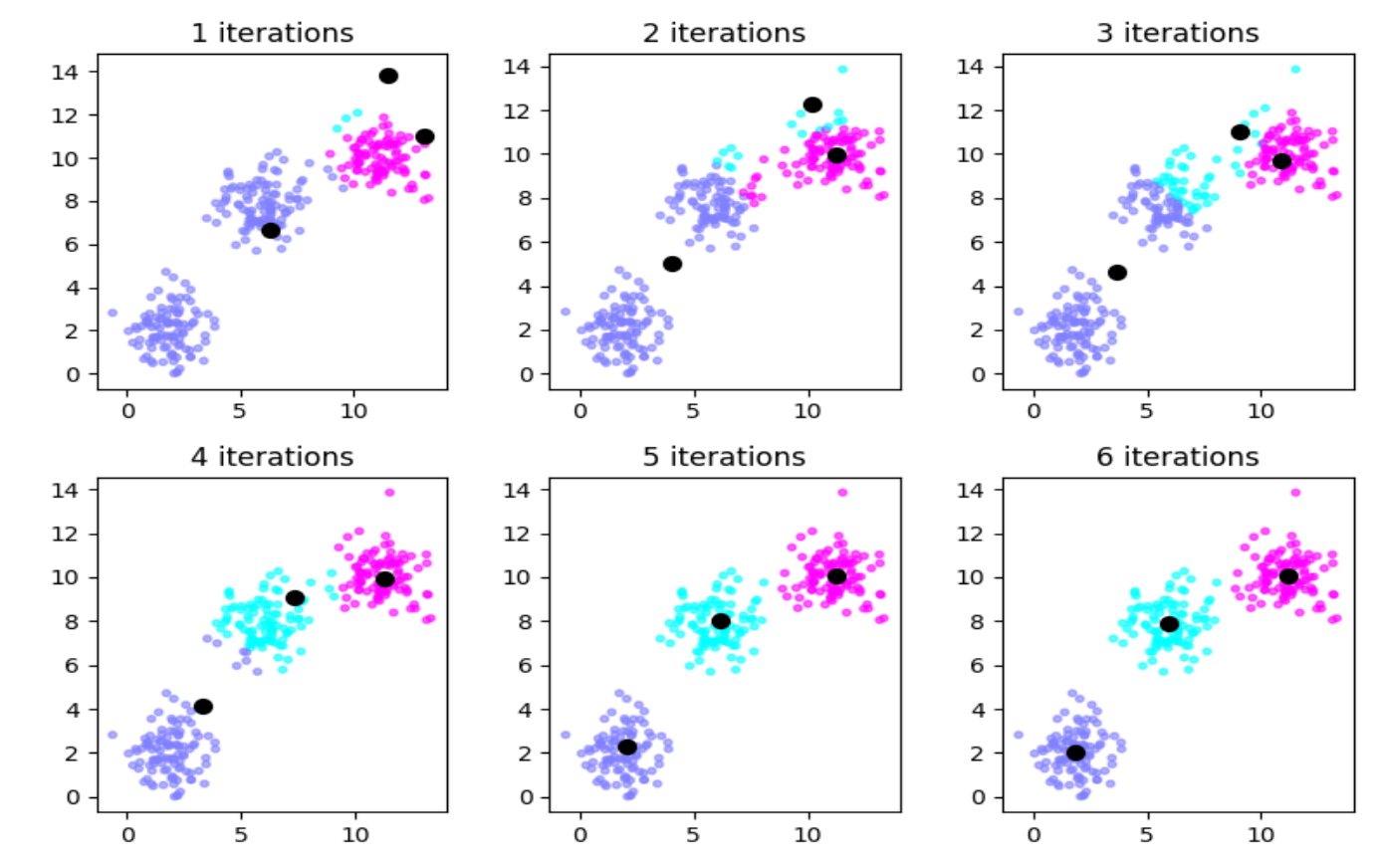
Until convergence**:

- assign each data point to the nearest centroid
- recompute the centroids as the mean of the data points

```
# psuedocode
initialize_centroids
while not converged:
  assign_data_to_centroids
  compute_new_centoid_means
```

K-Means - What it looks like

galvanize



Centroid Initialization Methods

galvanize

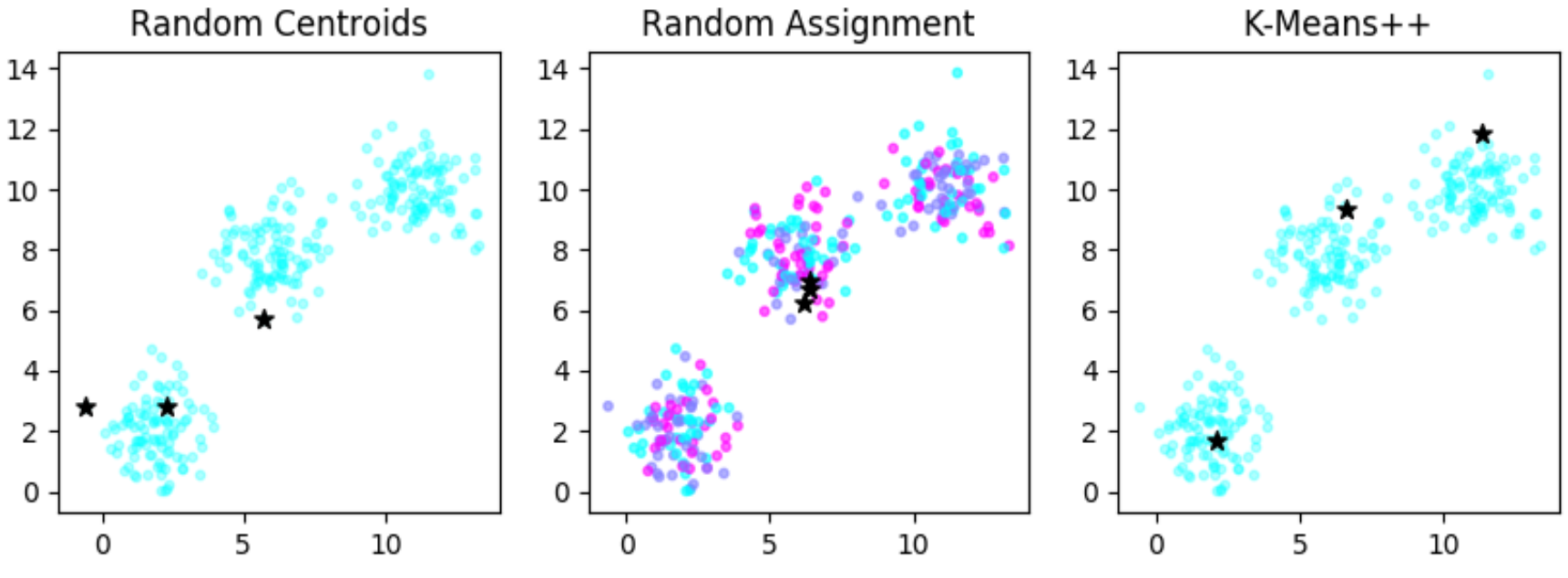
1) Randomly choose k points from your data and make those your initial centroids (simplest)

2) Randomly assign each data point to a number 1-k and initialize the kth centroid to the average of the points with the kth label (what happens as N becomes large?)

3) **k-means++** chooses well spread initial centroids. First centroid is chosen at random, with subsequent centroids chosen with probability proportional to the squared distance to the closest existing centroid. *(*default initialization in *sklearn).*

Centroid Initialization Methods

galvanize



Stopping Criteria

galvanize

We can update for:

1) A specified number of iterations (*sklearn default : max_iter= 1000*)

2) Until the centroids don't change at all

3) Until the centroids don't move by very much (*sklearn default : tol= .0001*)

Quick Review

galvanize

Goal of unsupervised learning?

Describe the 2 steps in each k-means fitting iteration

Name 3 ways to choose centroids

Name 3 stopping criteria

Is K-Means deterministic?

Should we standardize features?

Breakout (numpy practice)

galvanize

Use these arrays:

```
features = np.array([[ 3,  4],
                     [ 2,  2],
                     [ 5,  4],
                     [ 6,  9],
                     [-1,  0]])

labels = np.array([1, 0, 1, 0, 1])
```

1. Use masking to grab the rows of features that correspond to the label of 1
2. Find the column-wise mean of that subset of features
3. Compute the euclidean distance from each data point in features to this mean. Try to do this in one line with broadcasting!

Breakout (numpy practice)

galvanize

Use these arrays:

```
features = np.array([[ 3,  4],
                     [ 2,  2],
                     [ 5,  4],
                     [ 6,  9],
                     [-1,  0]])

labels = np.array([1, 0, 1, 0, 1])
```

1. `subset = features[labels == 1]`
2. `mean = subset.mean(axis = 0)`
3. `np.linalg.norm(features - mean, axis = 1)`

Evaluating K-Means

galvanize

How do we measure clustering performance / effectiveness?

Quantify how similar items are within a cluster

Minimize Intra-Cluster Variance or Within Cluster Variance (WCV)

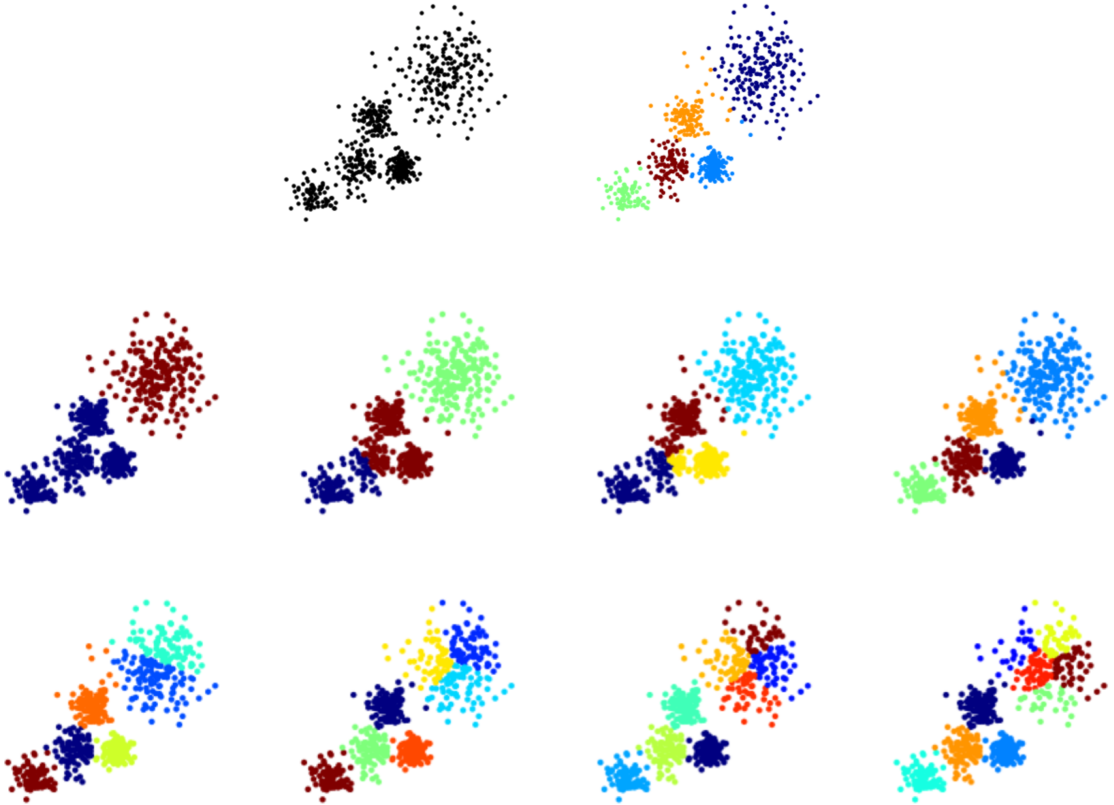
$$\min\left\{\sum_{k=1}^{K} WCV(C_k)\right\}$$

Where WCV for the kth cluster is the sum of all the pairwise Euclidean distances

$$WCV(C_k) = \frac{1}{|C_k|}\sum_{i,i' \in C_k}\sum_{j=1}^{p}(x_{ij} - x_{i'j})^2$$

Centroid of cluster C_k

K?



Choosing k

galvanize

Choosing k requires *a-priori* information:

- business logic (e.g. identify low, medium and high propensity customers)

- domain knowledge (e.g. there are k equilibrium states resultant from the phenomena)

Or a *heuristic*:

- Elbow plot

- Silhouette score

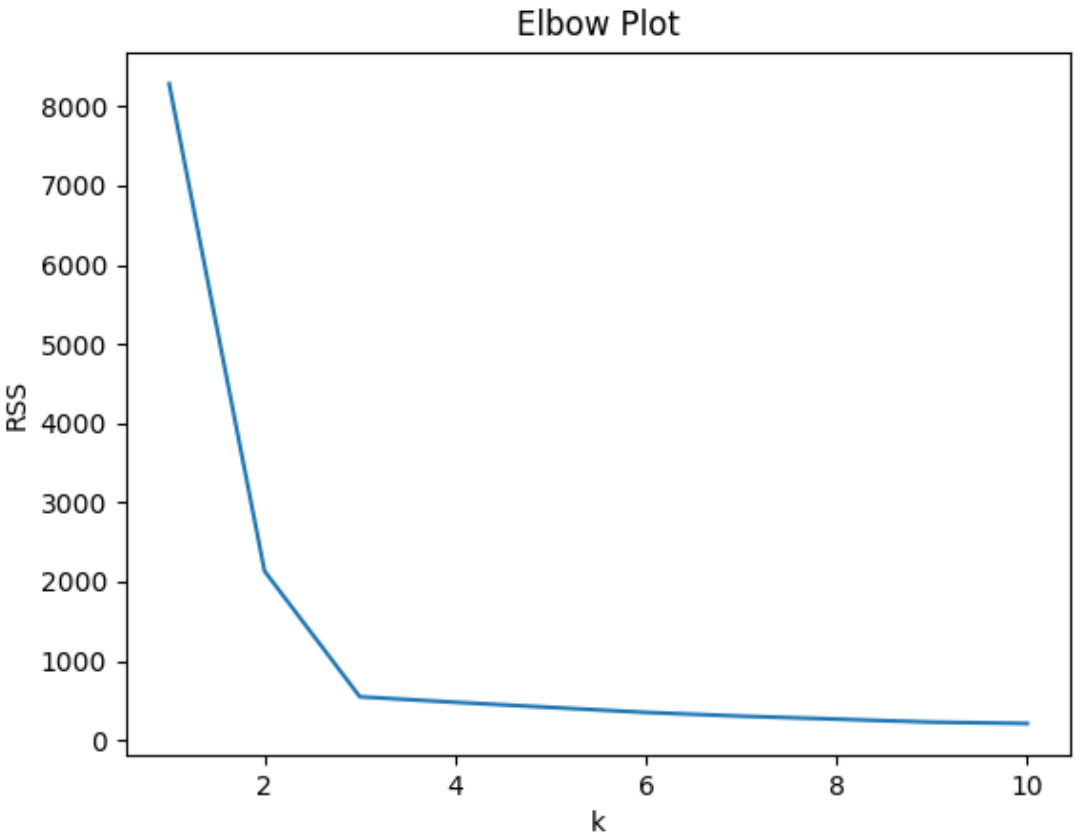
- GAP statistic and other methods that you don't need to know today

Elbow Plot

Elbow method - look for value of k that drastically reduces the residual sum of squares within the clusters

$$RSS = \sum_{k=1}^{K} \sum_{(i)=k} \left|\left| x_i - \bar{x}_k \right|\right|^2$$

Look for inflection point or value of k where improvement diminishes



Silhouette Score

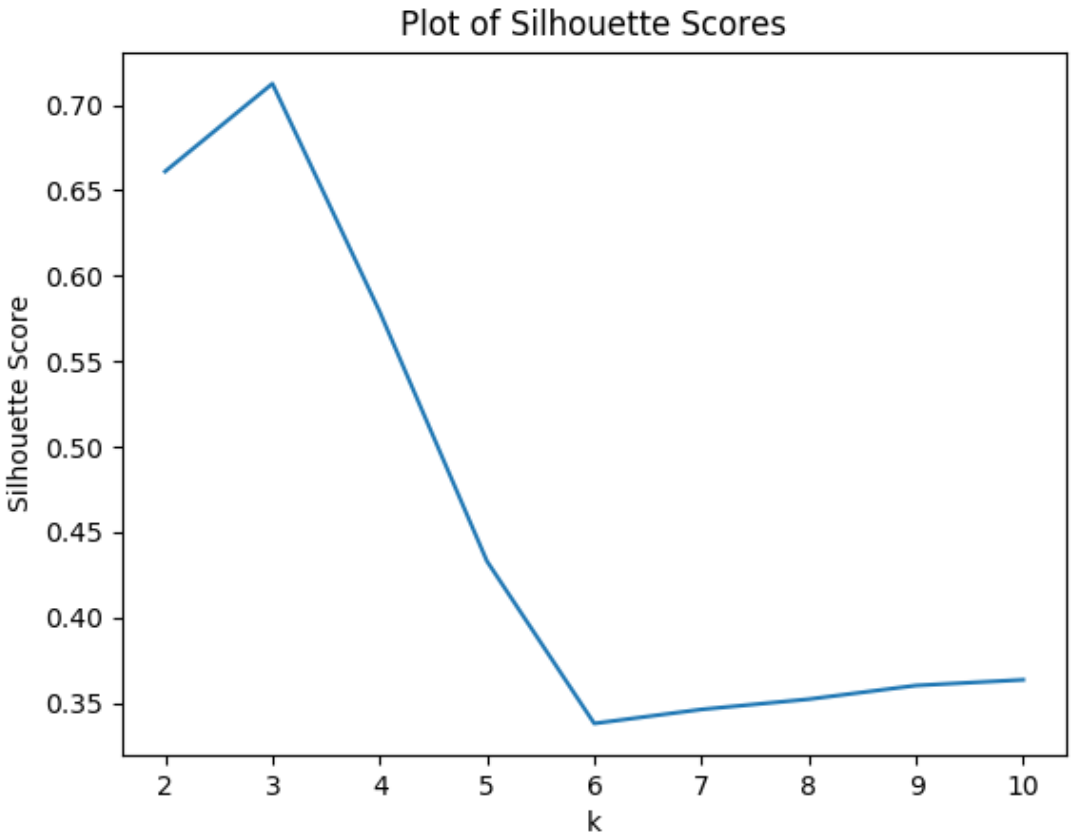
The Silhouette Coefficient is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample

(b - a) / max(a, b)

*only defined for 2 <= k < n

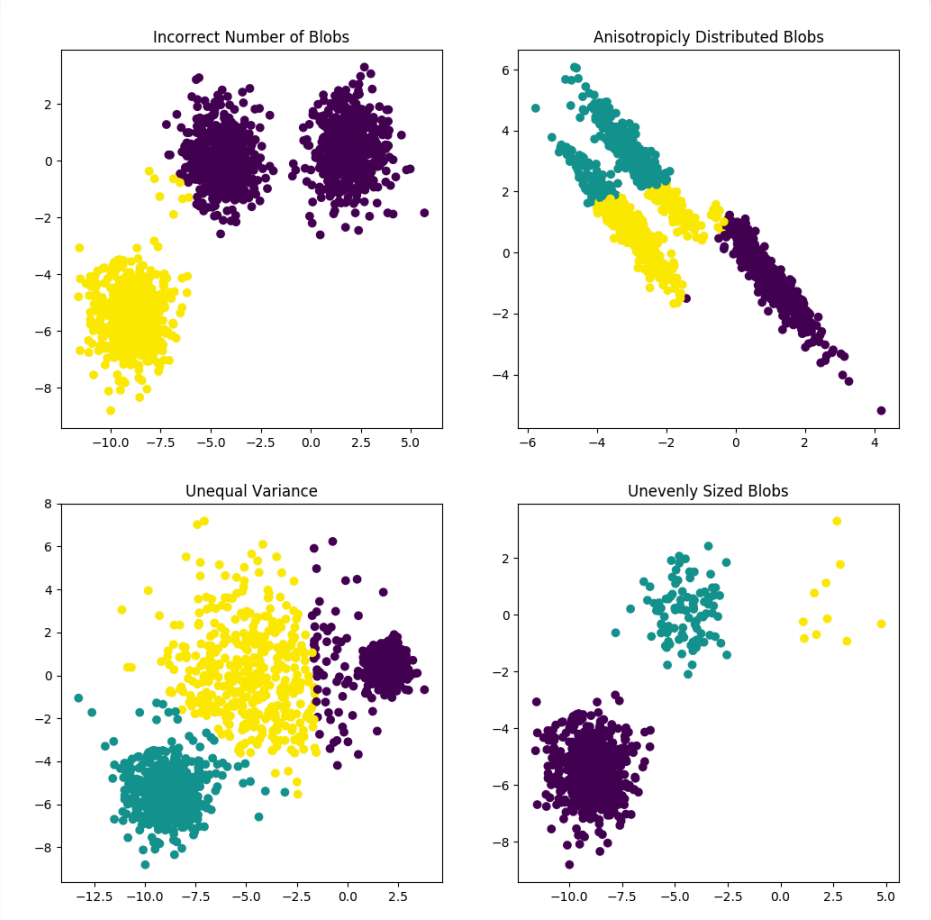
Values range from -1 to 1, with 1 being optimal and -1 being the worst

http://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html#sklearn.metrics.silhouette_score



K-Means Assumptions

- Picked the “correct” k
- Clusters have equal variance
- Clusters are isotropic (variance spherical)
- Clusters do NOT have to contain the same number of observations



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

Practical Considerations

galvanize

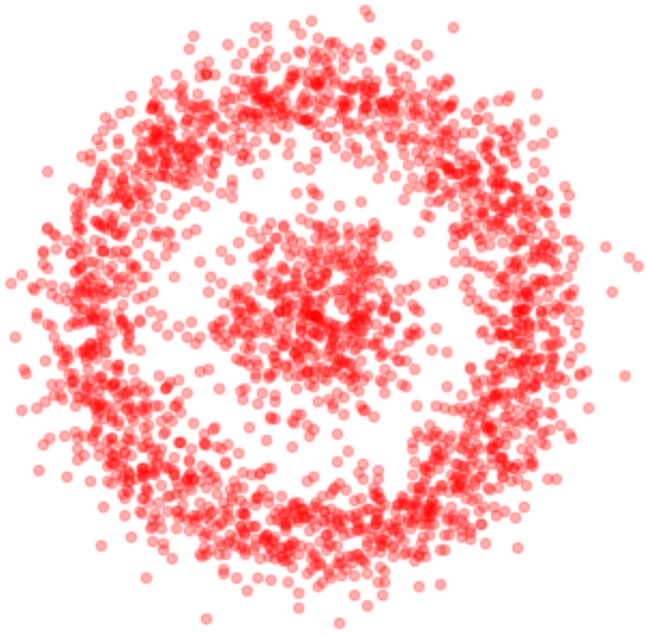
- K-means is not deterministic -- falls into local minima. Remedy by reinitializing multiple times and take the version with the lowest within-cluster variance (sklearn does multiple initializations by default)
- Susceptible to curse of dimensionality
- One hot encoded categorical can overwhelm - look into k-modes (https://pypi.python.org/pypi/kmodes/)
- Try MiniBatchKMeans for large datasets (finds local minima, so be careful)

DBScan Algorithm

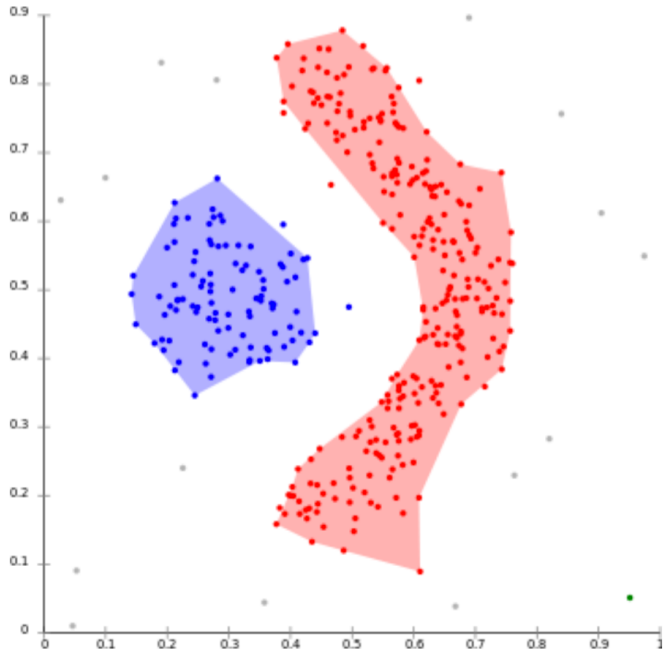
galvanize

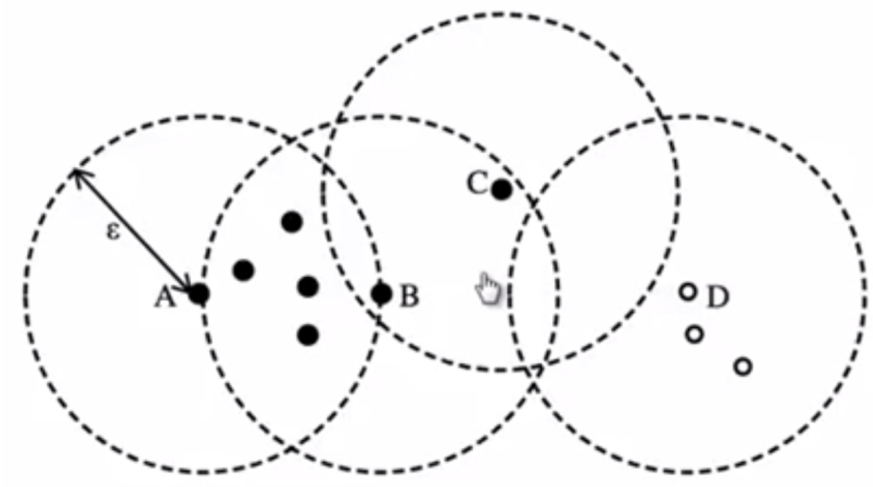
- Also computes clusters using distance metric, but decides the number of clusters for you
- With K-Means, you choose k - this is your **main** hyper-parameter
- With DBScan, your main hyper-parameter is **eps,** which is the maximum distance between two points that are allowed to be in the same ‘neighborhood’

DBSCAN

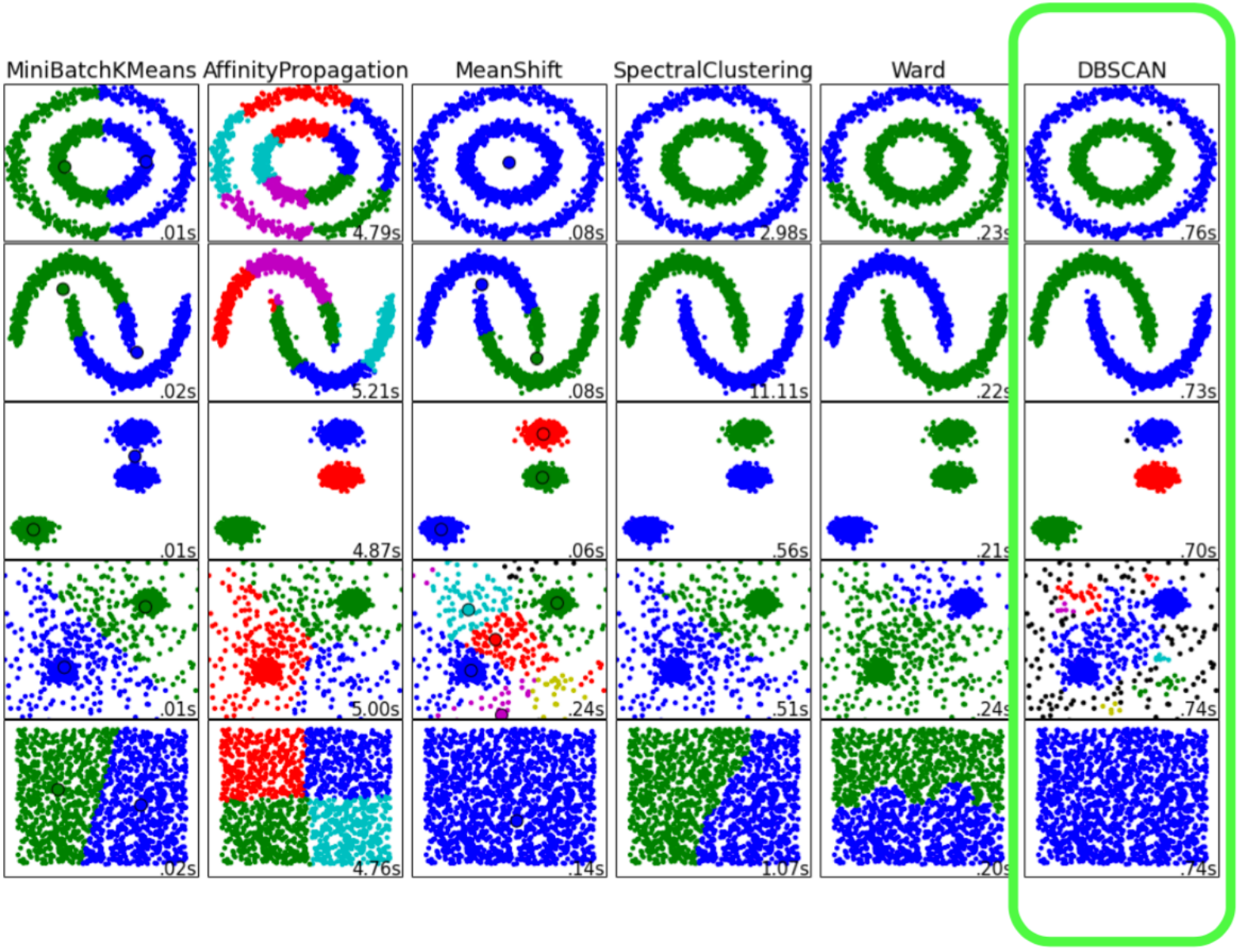


- Specify connection distance ϵ and minimum number of points for core m
- A cluster is all connected *core points*
- All other points are *noise*





DBSCAN


MiniBatchKMeans
AffinityPropagation
MeanShift
SpectralClustering
Ward
DBSCAN
.01s
4.79s
.08s
2.98s
.23s
.76s
.02s
5.21s
.08s
11.11s
.22s
.73s
.01s
4.87s
.06s
.56s
.21s
.70s
.01s
5.00s
.24s
.51s
.24s
.74s
.02s
4.76s
.14s
1.07s
.20s
.74s

Real world use cases

galvanize

- Customer segmentation
- Product segmentation
- Image segmentation
- Anomaly detection
- Social network analysis
- and many more…

Learning Objectives: Recap

galvanize

- Introduce unsupervised learning, compare to supervised
- Introduce clustering concept
- Enumerate clustering use-cases
- Define K-means algorithm:
  - How to code
  - Centroid initialization
  - Stopping Criteria
  - Evaluating the algorithm (metrics)
  - How to choose k

galvanize

Questions?

For your morning assignment, you will be coding the KMeans algorithm from scratch, and testing it on the iris dataset. I have included some (optional) starter code with the notes. You should be using numpy and vectorizing your calculations - there should only be one for loop, in the ‘fit’ method.