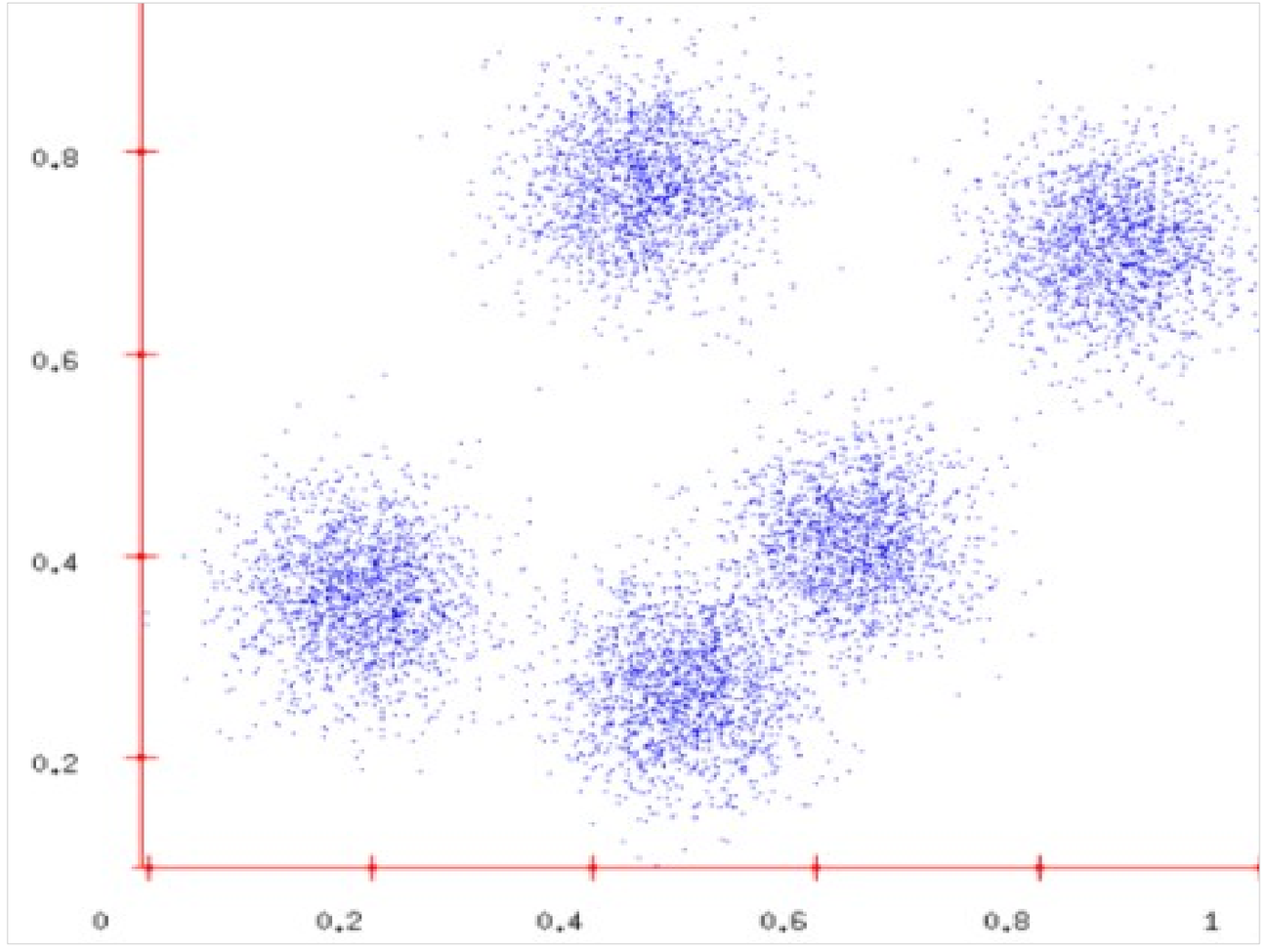
K Means

Vladimir Kulyukin
Department of Computer Science
Utah State University

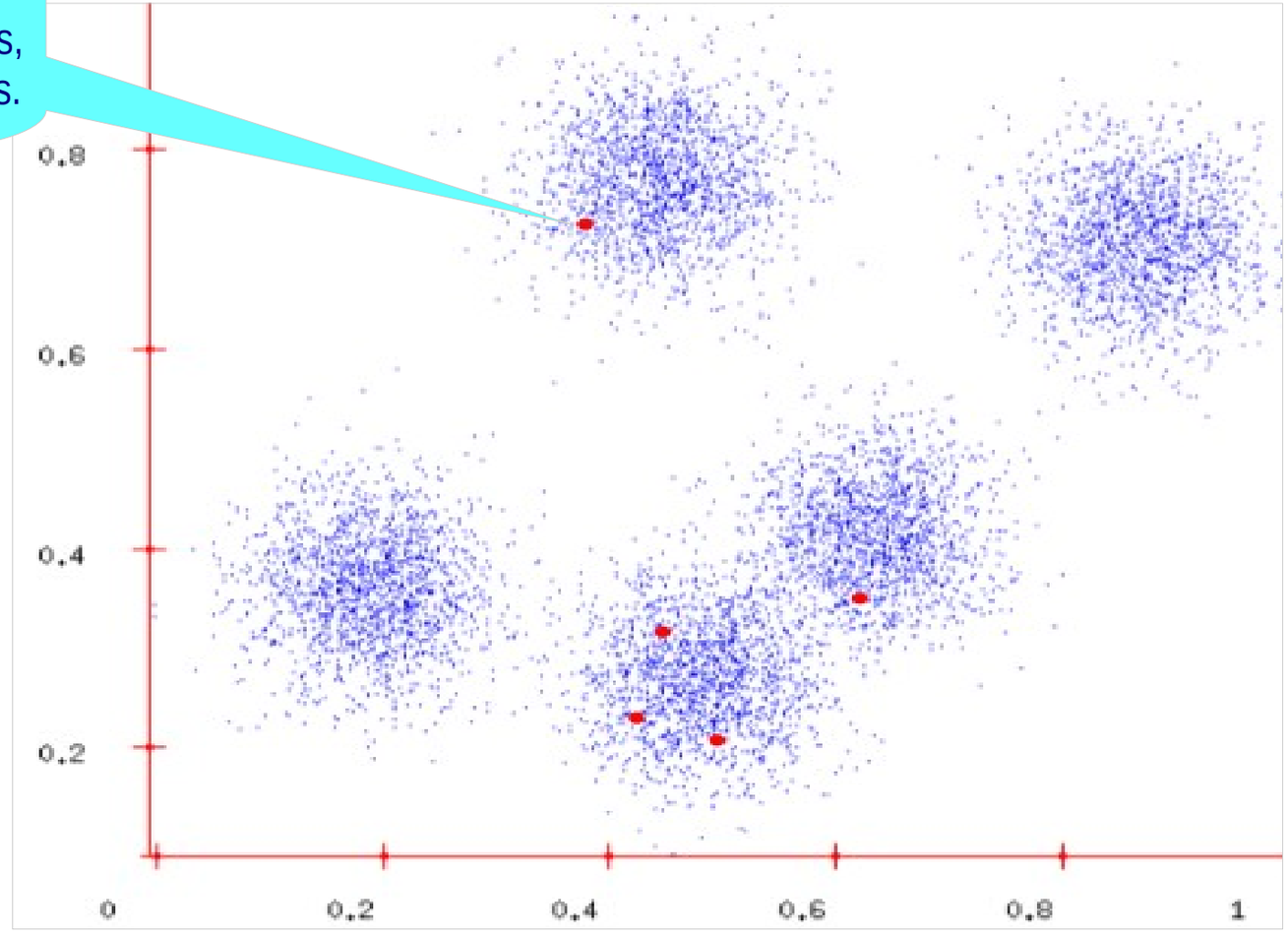
K-Means Illustrated: Step 1: Get Data



We have some 2D data to cluster

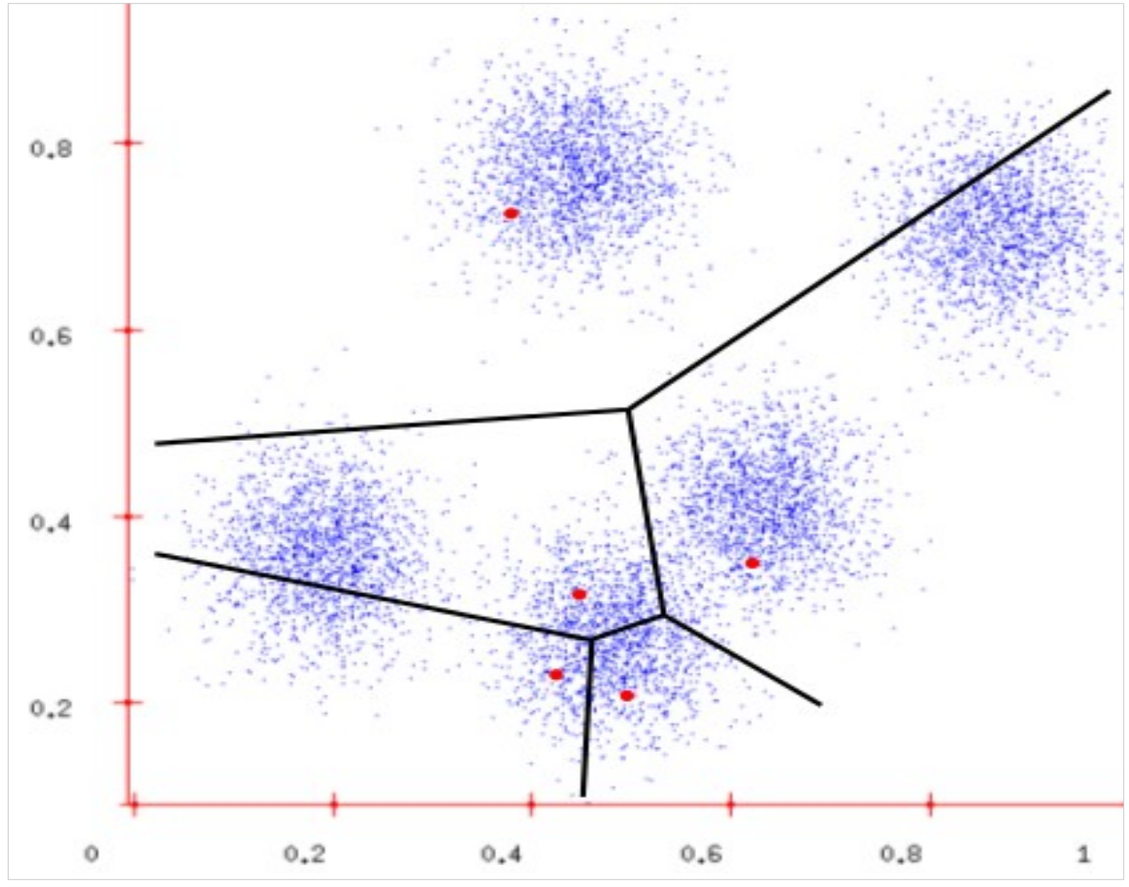
K-Means Illustrated: Step 2: Choose Centroids

Red dots are 2D centroids,
i.e. feature vectors. In n-dim spaces,
centroids are n-dimensional vectors.



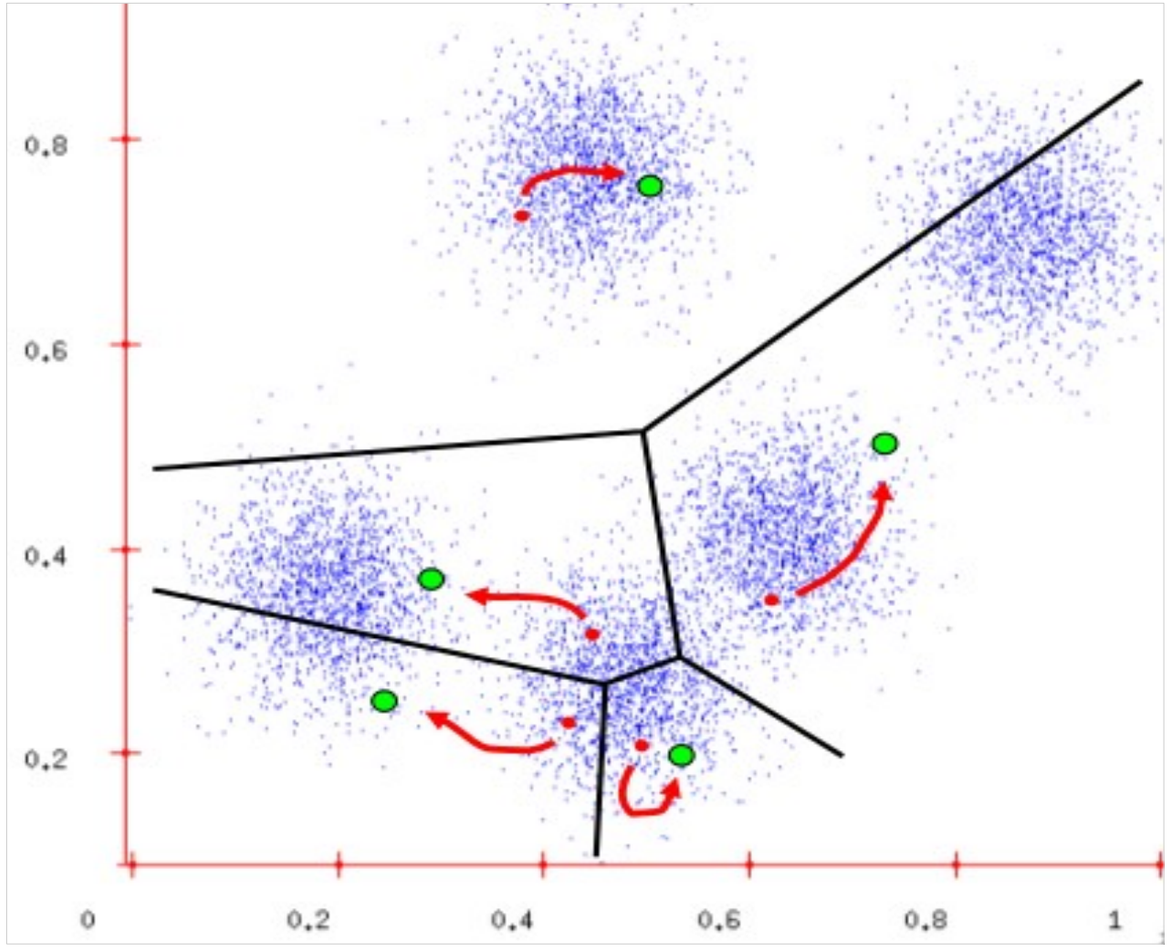
Pick a number of clusters you want and guess a centroid for each cluster; *guessing* typically refers to *random selection*

K-Means Illustrated: Step 3: Cluster Assignment



Each data item (i.e., feature vector) is assigned to a cluster on the basis of its closeness (e.g., euclidean distance) to a centroid; these centroids are referred to as *means* (this explains the means part of K-Means)

K-Means: Step 4: Adjust Centroids



Each centroid is adjusted by moving it to the center (mean) of its cluster; if centroid adjustment is small, stop; else recurse to step 3

K-Means in Pseudocode

1) Choose number of clusters you want to have (K)
2) Randomly choose K centroids (i.e., K feature vectors)
3) Assign all data items to one of the K clusters on the basis of their closeness to the randomly selected centroids
4) Adjust each centroid by moving it to the center of its cluster by setting the value of each feature to the mean of the values of that feature in all data items in the cluster
5) Note the amount of adjustment, i.e., the distance each centroid moves (this is called *inertia*)
6) If the amount of adjustment is greater than a threshold, go to step 3; this means the clusters are still unstable
7) If the amount of adjustment is smaller than a threshold, return

Problem

We know how to find posts related to a user's post by matching the feature vector of the user's post to the feature vectors of all posts in the database. This, of course, does not scale: as the number of posts grows, finding relevant posts becomes more and more time consuming. What we can do is 1) cluster the posts*; 2) find the cluster (or a small number of clusters) closest to the user's post; 3) look for related posts ***only within*** the closest cluster (or clusters).

* I am ignoring, for the sake of simplicity, how and where the clusters are stored. This is a cloud computing problem and, as such, has very little to do with scientific computing.

Getting Posts from Six Newsgroups

```
import sklearn.datasets
## select six neutral groups for training
groups = ['comp.graphics', 'comp.os.ms-windows.misc',
                'comp.sys.ibm.pc.hardware',  'comp.sys.mac.hardware',
                'comp.windows.x', 'sci.space']
## create train data
train_data = sklearn.datasets.fetch_20newsgroups(subset='train',
                                                 categories=groups)
```

source in find_post_with_kmeans.py

Clustering with K-Means

```
## 5) construct a tfidf vectorizer and build feature matrix of train_data
vectorizer = StemmedTfidfVectorizer(min_df=10,
                                    stop_words='english', decode_error='ignore')
train_data_feat_mat = vectorizer.fit_transform(train_data.data)

from sklearn.cluster import Kmeans
## 6) choose number of clusters
num_clusters = 50
## 7) create k-means clustering object
km = KMeans(n_clusters=num_clusters, verbose=1, random_state=3)
## 8) cluster the feature data
clustered_data = km.fit(train_data_feat_mat))
```

source in find_post_with_kmeans.py

K-Means: Displaying Clustering Stats

```
# after the clustering is done, each cluster has its own cluster label in [0, 49].
print('km.labels_=%s' % km.labels_)
print('len(km.labels_)=%d' % len(km.labels_))
# there are 3529 samples
# the number of samples is the same as the number of labels
# each label is an integer 0 <= i <= 49. or 0 <= i <= n-1
# where n is the number of clusters.
print('max label = %d' % max(km.labels_))
print('min label = %d' % min(km.labels_))
```

source in find_post_with_kmeans.py

K-Means: Classifying New Post

```
new_post = \
    """Disk drive problems. Hi, I have a problem with my hard disk.
After 1 year it is working only sporadically now.
I tried to format it, but now it doesn't boot any more.
Any ideas? Thanks.
"""


new_post_vec = vectorizer.transform([new_post]).getrow(0).toarray()
km_predicted_labels = km.predict(new_post_vec)
print('len(km_predicted_labels)=%d' % len(km_predicted_labels))

## get the top cluster label for the new post
top_new_post_label = km.predict(new_post_vec)[0]

print(top_new_post_label)
```

K-Means: Getting Similar Posts

computing distances b/w new post vector and the post feature vectors

```
posts_in_same_cluster = (km.labels_ == top_new_post_label).nonzero()[0]

similar_posts = [ ]
for i in posts_in_same_cluster:
    dist = sp.linalg.norm(new_post_vec - train_data_feat_mat[i])
    similar_posts.append((dist, train_data.data[i]

similar_posts.sort(key=lambda post: post[0])
```

So, what's the reduction of the search space? ## what is the reduction of the search space?

let's find out:

```
>>> len(train_data.filenames)
3529
>>> len(posts_in_same_cluster)
122
```

so, instead of matching aginst 3529 feature vectors, we are matching only against 122, i.e., about 3% of the available feature vectors.

Accessing Similar Posts

similar_posts is a list of 2-tuples of the form (dist, post_text).

accessting distance of top post

printing top post

```
>>> similar_posts[0][0]
1.038302580270046 5

>>> print(similar_posts[0][1])
From: Thomas Dachsel <GERTHD@mvs.sas.com>
Subject: BOOT PROBLEM with IDE controller
Nntp-Posting-Host: sdcmvs.mvs.sas.com
Organization: SAS Institute Inc.
Lines: 25
...
```

References

W. Richert & L. Coelho. “Building ML Systems with Python”, Ch. 3, Pack, 2013.

A. Moore. “K-Means & Hierarchical Clustering.” CMU, 2001.