K-means Clustering

Use a greedy algorithm to find an approximation to minimizing

$$\sum_{j=1}^{k} \sum_{i} \left\| x_i - \mu_j \right\|^2$$

clusters
points
distance
from
point to
centroid
Choose k

Randomly choose k examples as initial centroids

while true:
    create k clusters by assigning each example to closest centroid
    compute k new centroids by averaging examples in each cluster
    if centroids don’t change:
        break

What is complexity of one iteration?

k*n*c, where n is number of points and c time required to compute the distance between a pair of points
Example

From Perez et al
Pick Random Centroids
Assign Points to Closest Centroid
Find New Centroids
Assign Points to Closest Centroid
Find New Centroids
Assign Points to Closest Centroid
Find New Centroids
Assign Points to Closest Centroid
Find New Centroids
Observe that assignment doesn’t change
General Issues with K-Means

Result can depend upon initial centroids
Number of iterations
Even final result
Greedy algorithm can find different local optimas

Choosing the “wrong” k can lead to nonsense results
Consider k = 3
How to Choose K

A priori knowledge about application domain
There are two kinds of people in the world: $k = 2$
There are five different types of bacteria: $k = 5$

Search for a good k
Try different values of $k$ and evaluate quality of results
How to Deal with Initial Centroids

Try multiple sets of randomly chosen initial centroids
Select “best” result
Dealing with Dependence on Initial Centroids

Best = kMeans(points)
for t in range(numTrials):
    C = kMeans(points)
    if badness(C) < badness(best):
        best = C

Measuring badness

\[
\text{variance}(c) = \sum_{x \in c} (\text{mean}(c) - x)^2
\]

\[
\text{badness}(C) = \sum_{c \in C} \text{variance}(c)
\]
Cardiac Example

Generate many patients with 4 features each
- Heart rate in beats per minute
- Number of past heart attacks
- Age
- ST elevation (binary)

Generate outcomes based on features
- Probabilistic, not deterministic

Cluster into two clusters, and examine purity of clusters relative to outcomes
<table>
<thead>
<tr>
<th>Patient ID</th>
<th>Age</th>
<th>Gender</th>
<th>Duration</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>P000</td>
<td>89</td>
<td>1</td>
<td>0</td>
<td>66</td>
</tr>
<tr>
<td>P001</td>
<td>59</td>
<td>0</td>
<td>0</td>
<td>72</td>
</tr>
<tr>
<td>P002</td>
<td>73</td>
<td>0</td>
<td>0</td>
<td>73</td>
</tr>
<tr>
<td>P003</td>
<td>56</td>
<td>1</td>
<td>0</td>
<td>65</td>
</tr>
<tr>
<td>P004</td>
<td>75</td>
<td>1</td>
<td>1</td>
<td>68</td>
</tr>
<tr>
<td>P005</td>
<td>68</td>
<td>1</td>
<td>0</td>
<td>56</td>
</tr>
<tr>
<td>P006</td>
<td>73</td>
<td>1</td>
<td>0</td>
<td>75</td>
</tr>
<tr>
<td>P007</td>
<td>72</td>
<td>0</td>
<td>0</td>
<td>65</td>
</tr>
<tr>
<td>P008</td>
<td>73</td>
<td>1</td>
<td>0</td>
<td>64</td>
</tr>
<tr>
<td>P009</td>
<td>73</td>
<td>0</td>
<td>0</td>
<td>58</td>
</tr>
<tr>
<td>P010</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>75</td>
</tr>
<tr>
<td>P011</td>
<td>79</td>
<td>0</td>
<td>0</td>
<td>31</td>
</tr>
<tr>
<td>P012</td>
<td>81</td>
<td>0</td>
<td>0</td>
<td>58</td>
</tr>
<tr>
<td>P013</td>
<td>89</td>
<td>1</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>P014</td>
<td>81</td>
<td>0</td>
<td>0</td>
<td>70</td>
</tr>
</tbody>
</table>
Cluster
  update
  computeCentroid

tryKMeans

kMeans
Test k-means (k = 2)
Fraction of positives = 0.312
Cluster of size 280 with fraction of positives = 0.286
Cluster of size 220 with fraction of positives = 0.345

Test k-means (k = 3)
Fraction of positives = 0.312
Cluster of size 155 with fraction of positives = 0.361
Cluster of size 197 with fraction of positives = 0.259
Cluster of size 148 with fraction of positives = 0.331

Test k-means (k = 4)
Fraction of positives = 0.312
Cluster of size 126 with fraction of positives = 0.294
Cluster of size 130 with fraction of positives = 0.338
Cluster of size 149 with fraction of positives = 0.289
Cluster of size 95 with fraction of positives = 0.337
Not Impressive

Pretty close to what we would get by a random assignment
What’s going wrong?
What’s the Problem?

Features have very different means and variances
- Heart rate in beats per minute
- Number of past heart attacks
- Age
- ST elevation (binary)

If we use Euclidean distance metric, relevance of number of past heart attacks and ST elevation will be minimized because of relative size

Does this seem like a good idea?

Rescale all features to have same mean and variance

$$x' = \frac{x - \mu_x}{\sigma_x}$$
def scaleAttrs(vals):
    vals = pylab.array(vals)
    mean = sum(vals)/float(len(vals))
    sd = stdDev(vals)
    vals = vals - mean
    return vals/sd

What is the new mean?
What is the new standard deviation?
Let’s try it
Look at Code, and then Run It
Test k-means (k = 2) with scaling
Fraction of positives = 0.312
Cluster of size 254 with fraction of positives = 0.102
Cluster of size 246 with fraction of positives = 0.528

Test k-means (k = 3) with scaling
Fraction of positives = 0.312
Cluster of size 288 with fraction of positives = 0.021
Cluster of size 53 with fraction of positives = 0.792
Cluster of size 159 with fraction of positives = 0.679

Test k-means (k = 4) with scaling
Fraction of positives = 0.312
Cluster of size 125 with fraction of positives = 0.04
Cluster of size 53 with fraction of positives = 0.792
Cluster of size 159 with fraction of positives = 0.679
Cluster of size 163 with fraction of positives = 0.006
Try Scaling Cardiac Attributes

Are you more impressed?
How might we use this to risk stratify a previously unseen patient?
Find cluster with nearest centroid
Special case of k-nearest neighbors