CS 584 – Machine Learning

Topic: Clustering

Mustafa Bilgic

http://www.cs.iit.edu/~mbilgic

https://twitter.com/bilgicm

Motivation

- There is no supervision – i.e., there is no target variable
- Find groups / clusters in the data

Various Approaches

- K-Means
- Expectation Maximization (EM)
  - EM is a general approach that can be used for many other tasks, one of which is clustering
- Hierarchical clustering
- DBScan
- Spectral clustering
- Mean-shift
- … and many more

WE'LL COVER

- K-Means
- EM for mixture of Gaussians

K-MEANS

- Basic idea: given data, find k prototype vectors, and assign each data point to its closest prototype vector, such that the sum of the distances to the closest prototype vectors is minimized
- Let $D = \left\{x_j\right\}_{j=1}^{N}$ be the data and let $\{m_i\}_{i=1}^{k}$ be the prototype vectors. Objective:
  - minimize $\sum_{j=1}^{N}\left(\min_{i} distance\left(x_j, m_i\right)\right)$
  - A typical distance metric is Euclidan distance

K-Means

- Initialize k centers $\{m_i\}_{i=1}^{k}$
- Repeat until convergence
  - Assign each data point x_j to the closest cluster center
  - Re-compute each center $\{m_i\}_{i=1}^{k}$ as the mean of the points that are assigned to that center
- See OneNote for an illustration

EM for Mixture of Gaussians

- Basic idea: very much like the Bayes classifier where we have P(C) and p(x | C) and we use that for computing P(C | x), except, in this case, classes are not given to us. So, we take an iterative approach
- Repeat until convergence
  - Assign data points to classes/centers/components, <u>probabilistically</u> (i.e., compute P(C | x) using P(C) and p(x | C)) – **Expectation**
  - Re-estimate class/center/component parameters based on the <u>probabilistic</u> assignments to classes/centers (i.e., recompute P(C) and p(x | C) using the probabilistic assignments) – **Maximization**

Kmeans

assign
determinist

re-estim
means

NOTATION

- $P(C_i)$: prior probability of component C_i
  - A multinomial distribution
- $P(x_j|C_i)$: probability that component C_i generated x_j
  - A Gaussian distribution with mean μ_i and variance σ_i^2
    - This can be easily generalized to multivariate Gaussian distribution with mean μ_i and co-variance Σ_i
- $P(C_i|x_j)$: probability that x_j belongs to component C_i
  - Via Bayes rule, $P(C_i|x_j) \propto P(x_j|C_i)P(C_i)$
- n_i: Expected number of instances that belong to component C_i
  - $n_i = \sum_{j=1}^{N} P(C_i|x_j)$

C_1	C_2	C_3
0.8	0.1	0.1
0.2	0.7	0.1

EM

- E step: compute cluster/component assignments
  - $P(C_i|x_j) \propto P(x_j|C_i)P(C_i)$
- M step: recompute parameters
  - $n_i = \sum_{j=1}^{N} P(C_i|x_j)$
    - Weighted counts, where weights are $P(C_i|x_j)$
  - $P(C_i) = \frac{n_i}{N}$
  - $P(x_j|C_i) \sim \mathrm{N}(\mu_i, \sigma_i^2)$
    - $\mu_i = \frac{\sum_{j=1}^{N}\left(P(C_i|x_j)x_j\right)}{n_i}$
      - Weighted mean, where weights are $P(C_i|x_j)$
    - $\sigma_i^2 = \frac{\sum_{j=1}^{N}\left(P(C_i|x_j)(x_j - m_i)^2\right)}{n_i}$
      - Weighted variance, where weights are $P(C_i|x_j)$

x	C_1	C_1
1	0.1	0.9
2	0.8	0.2
5	0.7	0.3
7	0.6	0.4
	n_1	n_2

Scikit-Learn

- http://scikit-learn.org/stable/modules/clustering.html