Machine Learning

4771

Instructor: Tony Jebara

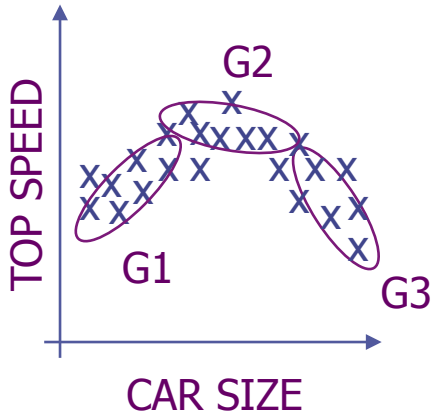
Topic 12

- Mixture Models and Hidden Variables
- Clustering
- K-Means
- Expectation Maximization

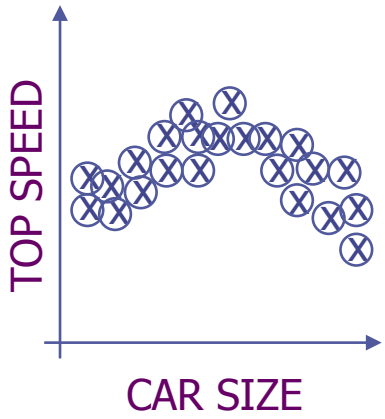
Mixtures for More Flexibility

- With mixtures (e.g. mixtures of Gaussians) we can handle more complicated (e.g. multi-bump, nonlinear) distributions.

  subpopulations: G1=compact car
  G2=mid-size car
  G3=cadillac



- In fact, if we have enough Gaussians (maybe infinite) we can approximate any distribution...

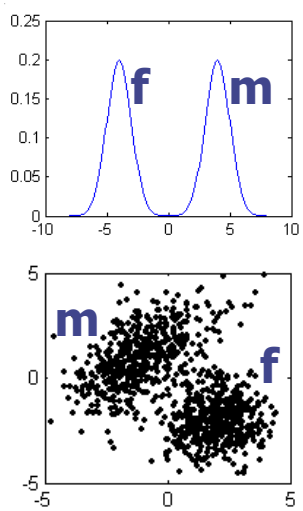


Mixtures as Hidden Variables

- Consider a dataset with K subpopulations but don't know which subpopulation each point belongs to

  I.e. looking at height of adult people, we see K=2 subpopulations: males & females

  I.e. looking at weight and height of people we see K=2 subpopulations: males & females



unsupervised

- Because of the 'hidden' variable (y can be 1 or 2), these distributions are not Gaussians but Mixture of Gaussians

$$p\left(\vec{x}\right) = \sum_y p(\vec{x}, y) = \sum_y p\left(y\right) p\left(\vec{x} \mid y\right) = \sum_y \pi_y N\left(\vec{x} \mid \vec{\mu}_y, \Sigma_y\right)$$

$$= \sum_{y=1}^{K} \pi_y \frac{1}{\left(2\pi\right)^{D/2} \sqrt{\left|\Sigma_y\right|}} \exp\left(-\frac{1}{2}\left(\vec{x} - \vec{\mu}_y\right)^T \Sigma_y^{-1} \left(\vec{x} - \vec{\mu}_y\right)\right)$$

probability that $\vec{x}_i$ occurs

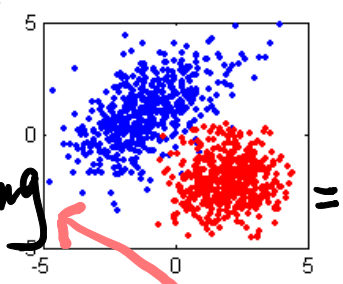
π_y: probability of y_i labels.

Unlabeled data → Clustering

- Recall classification problem:

  maximize the log-likelihood of data given models:

$$l = \sum_{n=1}^{N} \log p\left(\vec{x}_n, y_n \mid \pi, \mu, \Sigma\right)$$
$$= \sum_{n=1}^{N} \log \pi_{y_n} N\left(\vec{x}_n \mid \vec{\mu}_{y_n}, \Sigma_{y_n}\right)$$

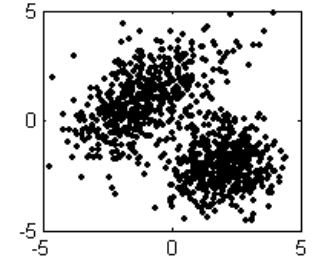


Supervised Learning (with label) ⇒ log likelihood = $\log \prod_{i=1}^{N} p(\vec{x}_n, y_n \mid \pi, u, \Sigma)$

$= \log \prod_{i=1}^{N} p(y_n \mid \pi) \cdot p(\vec{x}_n \mid y_n, \pi, u, \Sigma)$

- If we don't know the class treat it as a hidden variable

  maximize the log-likelihood with unlabeled data:



Unsupervised learning

$$l = \sum_{n=1}^{N} \log p\left(\vec{x}_n \mid \pi, \mu, \Sigma\right) = \sum_{n=1}^{N} \log \sum_{y=1}^{K} p\left(\vec{x}_n, y \mid \pi, \mu, \Sigma\right)$$
$$= \sum_{n=1}^{N} \log\left(\pi_1 N\left(\vec{x}_n \mid \vec{\mu}_1, \Sigma_1\right) + \ldots + \pi_K N\left(\vec{x}_n \mid \vec{\mu}_K, \Sigma_K\right)\right)$$

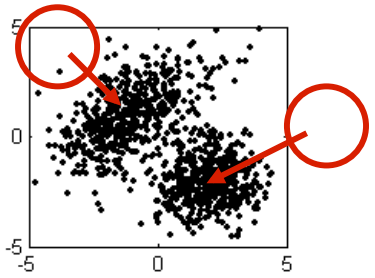
not $p(\vec{x}_n, y \mid \pi, u, \Sigma)$

Unsupervised (without label)

- Instead of classification, we now have a clustering problem

original log likelihood = $\log \prod_{i=1}^{N} p(\vec{x}_n \mid \pi, u, \Sigma) = \sum_{i=1}^{N} \log p(\vec{x}_n \mid \pi, u, \Sigma) = \sum_{i=1}^{N} \log \sum_{i=1}^{K} p(\vec{x}_n, y \mid \pi, u, \Sigma)$

K-Means Clustering



- K-means solves a Chicken-and-Egg problem:
  If knew classes, we can get model (max likelihood!)
  If knew the model, we can predict the classes (classifier!)
- Kmeans: guess a model, use it to classify the data, use classified data as labeled data to update the model, repeat.

- Assumes each point x has a discrete multinomial vector z

0) Input dataset $\{\vec{x}_1,\ldots,\vec{x}_N\}$

k = the number of classes

1) Randomly initialize means $\vec{\mu}_1,\ldots,\vec{\mu}_K$

2) Find closest mean for each point $\vec{z}_n(i) = \begin{cases} 1 & if\ i = \arg\min_j \left\|\vec{x}_n - \vec{\mu}_j\right\|^2 \\ 0 & otherwise \end{cases}$

i = arg min_j : j minimize

3) Update means $\vec{\mu}_i = \sum_{n=1}^{N} \vec{x}_n \vec{z}_n(i) \Big/ \sum_{n=1}^{N} \vec{z}_n(i)$

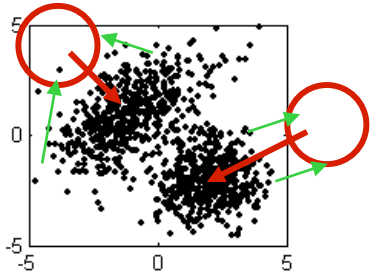
4) If any z has changed go to 2

$\vec{\mu}_i = \sum_{n \in \text{observations } i} \vec{x}_n$ / number of element in cluster

$z_n(i) = \begin{cases} 1 \\ 0 \end{cases}$

k=2 { $z_1(1), z_2(1), z_3(1), z_4(1), \ldots, z_N(1)$
$z_1(2), z_2(2), z_3(2), z_4(2), \ldots, z_N(2)$

K-Means Clustering



- Geometric, each point goes to closest Gaussian
- Recompute the means by their assigned points
- Essentially minimizing the following cost function:

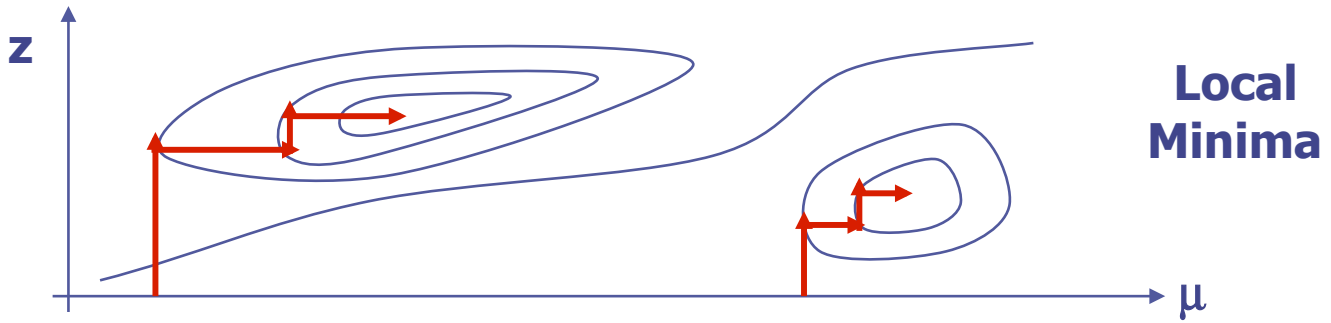
$$\min_{\mu}\min_{z} J\left(\vec{\mu}_1,\ldots,\vec{\mu}_K,\vec{z}_1,\ldots,\vec{z}_N\right)=\sum_{n=1}^{N}\sum_{i=1}^{K}\vec{z}_n\left(i\right)\left\|\vec{x}_n-\vec{\mu}_i\right\|^2$$

$$\vec{z}_n\left(i\right)=\begin{cases}1 & if\ i=\arg\min_j\left\|\vec{x}_n-\vec{\mu}_j\right\|^2\\ 0 & otherwise\end{cases}$$

$$\vec{\mu}_i=\frac{\sum_{n=1}^{N}\vec{x}_n\vec{z}_n\left(i\right)}{\sum_{n=1}^{N}\vec{z}_n\left(i\right)}$$

$\vec{z}_n(i)=0,1$

- Guaranteed to improve per iteration and converge
- Like Coordinate Descent (lock one var, maximize the other)
- A.k.a. Axis-Parallel Optimization or Alternating Minimization



40:00

Expectation-Maximization (EM)

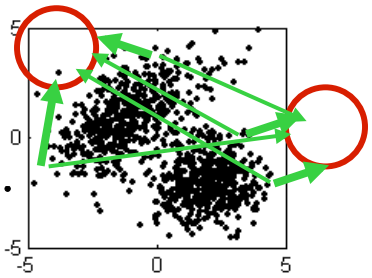
- EM is a soft/fuzzy version of K-Means (which does winner-takes-all, closest Gaussian Mean completely wins datapoint)

$$\vec{z}_n(i) = \begin{cases} 1 & if\ i = \arg\min_j \left\|\vec{x}_n - \vec{\mu}_j\right\|^2 = \arg\max_j N\left(\vec{x}_n \mid \vec{\mu}_j, I\right) = \arg\max_j p\left(\vec{x}_n \mid \vec{\mu}_j\right) \\ 0 & otherwise \end{cases}$$

- Instead, consider soft percentage assignment of datapoint

$$assign \propto \pi_j \frac{1}{(2\pi)^{D/2}} \exp\left(-\frac{1}{2}\left\|\vec{x}_n - \vec{\mu}_j\right\|^2\right)$$

$\theta = (\vec{D}_{(2)}, \vec{u})$

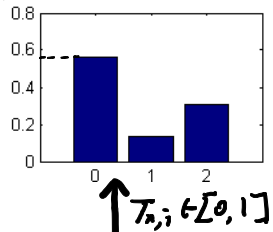


- EM is 'less greedy' than K-Means uses $\tau_{n,i} = p\left(\vec{z} = \vec{\delta}_i \mid \vec{x}_n, \theta\right)$ as shared responsibility for $\vec{x}_n$

$\tau_{n,i}$ = class i responsibility for Data n $= p(\vec{z} = \vec{\delta}_i \mid \vec{x}_n, \theta)$

$\theta = (\vec{D}_{(2)}, \vec{u})$

$\tau_{n,1}, \ldots, \tau_{n,K} =$



$\tau_{n,1} = p(\vec{z} = \vec{\delta}_1 \mid \vec{x}_n, \theta)$

$\sum_i \tau_{n,i} = 1$

$\tau_{n,i} \in [0,1]$

- Update for the means are then 'weighted' by responsibilities:

$$\mu_i = \frac{\sum_{n=1}^{N} \tau_{n,i} \vec{x}_n}{\sum_{n=1}^{N} \tau_{n,i}}$$

$\tau_{n,i} = p(z_n = \delta_i \mid x_n, \theta) = \frac{p(z_n=\delta_i, x_n \mid \theta)}{p(x_n \mid \theta)} = \frac{p(x_n \mid z_n=\delta_i, \theta)\cdot p(z_n=\delta_i \mid \theta)}{p(x_n \mid \theta)} = \frac{\mathcal{N}(x_n \mid \mu_i, \Sigma_i)\cdot \pi_i}{\sum_j \mathcal{N}(x_n \mid \mu_j, \Sigma_j)\pi_j}$

$p(x_n \mid z_n=\delta_i, \theta) = \mathcal{N}(x_n \mid \mu_i, \Sigma_i)$

$p(z_n=\delta_i \mid \theta) = \pi_i$

$p(x_n \mid \theta) = \sum_j p(x_n, z_n=\delta_j \mid \theta) = \sum_j p(x_n \mid \theta, z_n=\delta_j)\cdot p(z_n=\delta_j \mid \theta)$
$= \sum_j \mathcal{N}(x_n \mid \mu_j, \Sigma_j)\cdot \pi_j$

$p(x_n \mid \theta) = \sum_j p(x_n, z_n=\delta_j \mid \theta)$
$= \sum_j p(x_n \mid z_n=\delta_j, \theta)\cdot \pi_j$

Tony Jebara, Columbia University

Expectation-Maximization

- EM uses expected value of $\vec{z}_n(i)$ rather than max

$$\tau_{n,i} = E\left\{\vec{z}_n(i) \mid \vec{x}_n\right\} = p\left(\vec{z}_n = \vec{\delta}_i \mid \vec{x}_n, \theta\right)$$

- EM updates covariances, mixing proportions AND means...
- The algorithm for Gaussian mixtures:

EXPECTATION:

$$\tau_{n,i}^{(t)} = \frac{\pi_i N\left(\vec{x}_n \mid \vec{\mu}_i^{(t)}, \Sigma_i^{(t)}\right)}{\sum_j \pi_j N\left(\vec{x}_n \mid \vec{\mu}_j^{(t)}, \Sigma_j^{(t)}\right)}$$

MAXIMIZATION:

$$\vec{\mu}_i^{(t+1)} = \frac{\sum_n \tau_{n,i}^{(t)} \vec{x}_n}{\sum_n \tau_{n,i}^{(t)}} \qquad \pi_i^{(t+1)} = \frac{\sum_n \tau_{n,i}^{(t)}}{N}$$

$$\Sigma_i^{(t+1)} = \frac{\sum_n \tau_{n,i}^{(t)} \left(\vec{x}_n - \vec{\mu}_i^{(t+1)}\right)\left(\vec{x}_n - \vec{\mu}_i^{(t+1)}\right)^T}{\sum_n \tau_{n,i}^{(t)}}$$

硬币模型

- DEMO... like an iterative divide-and-conquer algorithm
- But, divide&conquer is not a guarantee. Can we prove EM?

https://blog.csdn.net/qq_33369979/article/details/103913637
https://blog.csdn.net/u010834867/article/details/90762296