Lecture 14. Clustering, K-means, and EM

Prof. Alan Yuille

Spring 2014

Outline

1. Clustering
2. K-means
3. EM

1 Clustering

Task: Given a set of unlabeled data $D = \{x_1, ..., x_n\}$, we do the following:

1. Decompose them into classes $w_1, ..., w_M$, where M is unknown
2. Learn class models $p(x|w)$
3. Discovery of new classes (concepts)

We provide motivations in three perspectives.

The first one is dictionary-based method. Suppose we have a dictionary of k elements, $< w_1, ..., w_k >$. We want to assign each datapoint x to the k elements.

$$x = (\alpha_1, ..., \alpha_k), \sum_{i=1}^{k} \alpha_i = 1, \alpha_i \geq 0.$$

We call it "hard assignment" if $\alpha_i = 0$ or $\alpha_i = 1$. In this case, each datapoint is assigned to one closest element. And we call it "soft assignment" if $\alpha_i \in [0, 1]$. In this case, α_i represents the probability of datapoint being element i.

The second perspective is from dimension reduction. Recall we have data x of very high dimension D. And we want to use y to represent x, where the dimension of y is much smaller than D.

The third one is unsupervised learning. There are some hidden/latent/missing variables that are not specified by the training data. I.e., the training data is $\{x_1, ..., x_n\}$ and we assume that it is generated by a model $P(x|w)P(w)$, where the w are unknown and can be called latent/hidden/missing (Statisticians call them 'missing data', Neural Network researchers call them 'hidden units', Machine Learning researchers call them 'latent variables'). By contrast, in previous lectures, we assumed a model $p(x)$ which was either an exponential distribution or a non-parametric distribution.

We can think of this as learning a model $P(x, w)$ and we can assume that this model is of exponential form $P(x, w) = \frac{1}{Z[\lambda]} \exp\{\lambda \cdot \phi(x, w)\}$, i.e. an exponential distribution defined over the observer units x and the unobserved units w (latent/hidden/missing). This model can capture extremely complex models – stochastic grammars on natural language, hierarchical models of objects, probabilistic models of reasoning. But these models are harder to deal with than models without hidden variables. The models require: (i) learning the structure – how the hidden and observed units interact (see later lectures) – which is called 'structure induction' and which is very difficult, (ii) learning the parameters λ of the model if the structure is known – this is a non-convex optimization problem and so it will have multiple minima. We will describe the basic algorithm for doing this – the EM algorithm – later in this lecture. (Note: there is also a machine learning algorithm called latent SVM which is like an approximation to EM). Note, there are also non-parametric methods for learning $P(x, w)$.

2 K-means

Basic Assumption:
the data D is clustered round (unknown) mean values $m_1, ..., m_k$, see figure 1. This is an assumption about the 'structure of the model'.

Seek to associate the data to these means, and to simultaneously estimate the means. For now, we assume that the number of means is known – i.e. k is fixed.

Idea: decompose $D = \cup_{a=1}^{k} D_a$, where D_a is data associated with mean m_a

Mean of Fit $F(\{D_a\}) = \sum_{a=1}^{k} \sum_{x \in D_a} (x - m_a)^2$

Want to select $\{m_a\}$, and assignment $x \in D_a$ to minimize the fit $F(\cdot)$.

Assignment Variable:
$V_{ia} = 1$, if data x_i is assigned to m_a

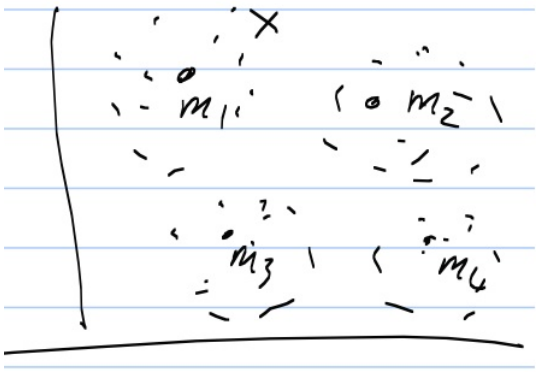

Figure 1: The k-means algorithm assumes that the data can be clustered round (unknown) mean values $m_1, ..., m_k$. Equivalently, that the data is generated by a mixture of Gaussians with fixed covariance and with these means.

$V_{ia} = 0$, otherwise

Constraint $\sum_{a=1}^{k} V_{ia} = 1, \forall i$,
(i.e. each datapoint x_i is assigned to only one class m_a). Note: don't need assignment variables yet.
Deterministic K-means:
1. Initialize a partition $\{D_a^0 : a = 1 \text{ to k}\}$ of the data
(E.G. randomly choose points x and put them into set, $D_1^0, D_2^0, ..., D_k^0$ - so that all datapoints are in exactly one set)
2. Compute the mean of each cluster D_a, $m_a = \frac{1}{|w_a|} \sum_{x \in D_a} x$
3. For i=1 to N, compute $d_a(x_i) = |x_i - m_a|^2$
Assign x_i to cluster D_{a^*}
s.t. $a^* = \arg\min\{d_a(x_i), ..., d_k(x_i)\}$
4. Repeat steps 2 & 3 until convergence.

Comment:
The k-mean algorithm is not guaranteed to converge to the best fit of $F(D)$

It will almost always converge to a fixed point.

Typically, you can improve k-means by giving it different initial conditions. Then select the solution which has best fit.

Evaluate solution with different values of k. If the number of means (clusters) is too large, then you will usually find some of the mean will be close together, see figure (2).

i.e do postprocessing to eliminate means which are too close to each other.

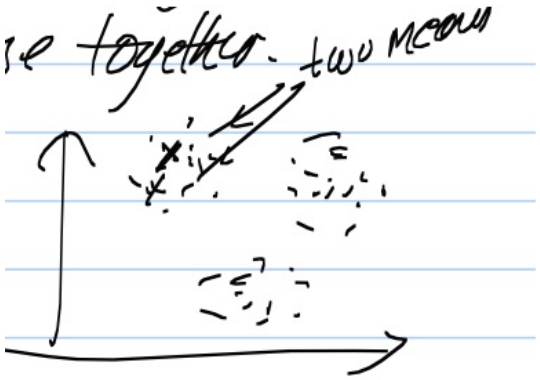

Figure 2: There are three clusters and four 'means'. In practice, two 'means' will usually be assigned to one cluster.

A "softer" version of k-means – the Expectation-Maximization (EM) algorithm. Assign datapoint x_i to each cluster with probability $(P_1, \ldots, P_k)$

1. Initialize a partition.

E.G. randomly chose k points as centres $m_1, m_2, \ldots, m_k$

2. For j=1 to N,

Compute distances $d_a(x_j) =\mid x_j - m_a \mid^2$

Compute the probability that x_j belongs to ω_a

$P_a(x_j) = \frac{1}{(2\pi\sigma_a^2)^{d/2}} e^{-\frac{1}{2\sigma_a^2}(x_j - m_a)^2}$

3. Compute the mean and variance for each cluster:

$m_a = \frac{1}{|D_a|} \sum_{x \in D_a} x P_a(x)$

$\sigma_a^2 = \frac{1}{|D_a|} \sum_{x \in D_a} (x - m_a)^2 P_a(x)$

Repeat steps 2 & 3 until convergence.

Comment:
K-means++ is an algorithm the choose k initial means. The implementation is freely available online.

3 EM algorithm

3.1 Generative model with latent variables

Modeling data with hidden variables

$P(x, h, \theta)$

* x : observed data

* h : hidden variable

*θ : model parameter

Example:

data is generated by a mixture of Gaussian distribution with unknown means and covariance.

The variable h indicates which Gaussian generated the data.

$P(x \mid h = a, \theta_a) = \frac{1}{(2\pi)|\Sigma_a|^{1/2}} e^{-\frac{1}{2}(x-\mu_a)\Sigma_a^{-1}(x-\mu_a)}$

Assume $P(h = a \mid \theta) = \frac{e^{\phi_a}}{\sum_c e^{\phi_c}}$, and a prior probability $P(\theta)$ on $\{\phi_a, \theta_a\}$

The joint distribution can be written as $P(x, h, \theta) = p(x|h, \theta)p(h|\theta)p(\theta)$, see figure 3.

Now perform MAP estimation of θ from data examples $D = \{x_1, .., x_n\}$

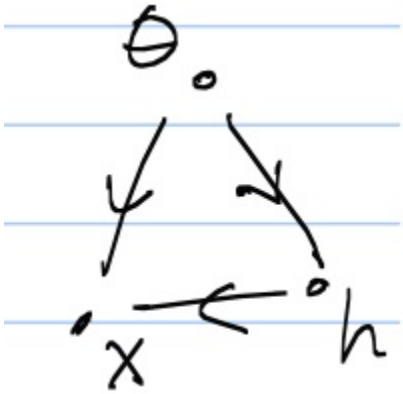

Figure 3: The data x is generated by hidden variable h by a probability model with unknown parameters θ.

$p(D|h, \theta) = p(\{x_i, ..., x_n\}|h, \theta) = \prod_{i=1}^{N} p(x_i|h, \theta)$ i.i.d. assumption.

$p(\theta|D) = \sum_h p(\theta, h|D)$

How to estimate $\hat{\theta}$ from D?

The most popular algorithm: Expectation Maximization (EM) algorithm.

Comment: standard k-means algorithm can be thought of as a limiting case of EM for mixture of Gaussians - where the covariance is fixed to be the identity matrix.

3.2 The EM Algorithm

$p(\theta|D) = \sum_h p(\theta, h|D)$

Define a new distribution $q(h)$

Minimize $F(\theta, q) = -\log p(\theta|D) + \sum_h q(h) \log \frac{q(h)}{p(h|\theta,D)}$

* Kullback-Leibler divergence: $\sum_h q(h) \log \frac{q(h)}{p(h|\theta,D)} \geq 0$

Note, the minimum occurs at $\hat{\theta} = \arg\min_\theta\{-\log p(\theta|D)\} = \arg\max_\theta p(\theta|w)$
and at $\hat{q}(h) = p(h|\hat{\theta}, D)$
(Because the Kullback-Liebler divergence attains its minimum at $\hat{q}(h) = p(h|\hat{\theta}, w)$).

We can re-express the Free Energy

$F(\theta, q) = -\log p(\theta|D) + \sum_h q(h) \log \frac{q(h)}{p(h|\theta,D)}$
$= -\sum_h q(h) \log p(\theta|D) + \sum_h q(h) \log q(h) - \sum_h q(h) \log p(h|\theta, D)$
$= \sum_h q(h) \log\{p(\theta|D)p(h|\theta, D)\} + \sum_h q(h) \log q(h)$

$F(\theta, q) = -\sum_h q(h) \log p(h, \theta|D) + \sum_h q(h) \log q(h)$

EM minimize $F(\theta, q)$ w.r.t. $\theta \& q$ alternatively. This is called 'coordinate descent', see figure (4). Here is an intuition for this algorithm. You live in a city with hills, like Seoul, and the streets are arranged in a horizontal grid of blocks. You start on a hill (at high altitude) and want to decrease your altitude. You can either walk in the North-South direction or in the East-West direction. You look in all directions and must choose to walk North, South, East, or West. You see that North and East are uphill, so you do not choose them. You see that you can walk South and decrease your altitude by only 10 meters before the street starts going uphill. You look West and see that you can decrease your altitude by 100 meters by walking in that direction (until that street also starts going uphill). So you – i.e. coordinate descent – chooses to walk West and stop when the street starts going uphill (i.e. you have lost 100 meters). The you stop, look again in the directions North, South, East, West and walk in the direction to maximize your decrease in altitude. The you repeat until you are at a place where all directions are uphill.

Note: coordinate descent will be important when the dimension of the space is very big. In two-dimensions, like the city example, the number of choices is small – so if you have 'moved' East-West last time then you have to move North-South the next time. In high-dimensions, there are an enormous number of choices. So the algorithm will often

choose only to 'move' in a small number of possible directions, and the other directions will be irrelevant. This will be important for AdaBoost learning.

Note: you may not be able to calculate how much you will decrease altitude by walking in one direction. This will depend on the specific problem.

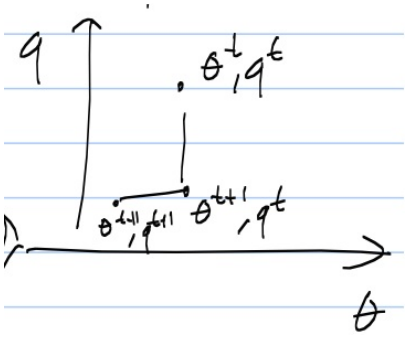

Figure 4: The coordinate descent algorithm. You start at θ^t, q^t. You fix q^t and decrease $F(\theta, q)$ by changing θ until you reach θ^{t+1}, where $\frac{\partial}{\partial\theta}F(\theta, q^t) = 0$. Then fix θ^{t+1} and decrease $F(\theta, q)$ by changing q until you reach q^{t+1}, where $\frac{\partial}{\partial q}F(\theta^{t+1}, q) = 0$.

Step 1: Fix q^t, set $\theta^{t+1} = \arg\min_\theta\{-\sum_h q(h)\log p(h, \theta|D)\}$
Step 2: Fix θ^t, set $q^{t+1}(h) = p(h|\theta^t, D)$

Iterate steps 1 & 2 until convergence.

Note: this algorithm is guaranteed to converge to a local minimum of $F(\theta, q)$, but there is no guarantee that it will converge to the global minimum. You can use multiple starting points – and pick the solution which has lowest value of $F(.,)$ – or you can use extra knowledge about the problem to have a good starting point, or starting points. Or you can use a stochastic algorithm.

3.3 Exponential EM Example

EM takes a simple form for exponential distributions.

$P(x, h \mid \lambda) = \frac{e^{\lambda\phi(x,h)}}{z[\lambda]}$

General exponential form, where the h are hidden variables / latent variables.

$P(x \mid \lambda) = \Sigma_h P(x, h \mid \lambda)$

Two steps of EM

Step 1:

$q^{t+1}(h) = P(h \mid x, \lambda^t) = \frac{P(x,h|\lambda^t)}{\Sigma_x P(x,h|\lambda^t)}$

Step 2:

$\lambda^{t+1} = \arg\min_\lambda \quad -\sum_h q^{t+1}(h)\{\log P(x, h \mid \lambda)\}$

$\lambda^{t+1} = \arg\min_\lambda \quad -\{\sum_h q^{t+1}\phi(x, h)\lambda + \log z[\lambda]\}$

Differentiating gives : $\sum_{x,h} P(x, h \mid \lambda^{t+1})\phi(x, h) = \sum_h q^{t+1}(h)\phi(x, h)$

3.4 Mixture of Gaussian Example

$P(y_i \mid V_{ia} = 1, \theta) = \frac{1}{(2\pi)^{d/2}\|\Sigma_a\|^{1/2}} e^{-\frac{1}{2}(y_i-\mu_a)^T\Sigma_a^{-1}(y_i-\mu_a)}$

datapoint y_i is generated by a^{th} model.

We can write :

$P(y_i \mid \{V_{ia}\}, \theta) = \frac{1}{(2\pi)^{d/2}} e^{-\sum_a V_{ia}\{\frac{1}{2}(y_i-\mu_a)^T\Sigma_a^{-1}(y_i-\mu_a)+\frac{1}{2}\log\|\sum_a\|^{1/2}\}}$

By i.i.d assumption,

$P(\{y_i\} \mid \{V_{ia}\}, \theta) = \prod_i P(y_i \mid \{V_{ia}\}, \theta) = \frac{1}{(2\pi)^{nd/2}} e^{-\sum_{ia} V_{ia}\{\frac{1}{2}(y_i-\mu_a)^T\Sigma_a^{-1}(y_i-\mu_a)+\frac{1}{2}\log\|\sum_a\|^{1/2}\}}$

Let the prior probability on the $\{V_{ia}\}$ be uniform unknown $P(\{V_{ia}\}) = \text{constant}$.

Then

$P(\{y_i\}, \{V_{ia}\} \mid \theta) = \frac{e}{Z}^{-\sum_{ia} V_{ia}\{\frac{1}{2}(y_i-\mu_a)^T\Sigma_a^{-1}(y_i-\mu_a)+\frac{1}{2}\log\|\sum_a\|^{1/2}\}}$

*Z : Normalization constant.

To apply EM to this problem,

E-step:

$Q(\theta \mid \theta^{(t)}) = \sum_v \{\log P(y, V \mid \theta)\} P(V \mid \theta^t)$

M-step: Determine $\theta^{(t+1)}$ by maximizing $Q(\theta \mid \theta^{(t)})$

$P(\{V_{ia}\} \mid \theta) = \prod_i P_i(V_{ia} \mid \theta)$.

$P_i(V_{ia} \mid \theta) = \frac{e^{-V_{ia}\{\frac{1}{2}(y_i-\mu_a)^T\Sigma_a^{-1}(y_i-\mu_a)-\frac{1}{2}\log|\sum_a|\}}}{\sum_b e^{-V_{ib}\{\frac{1}{2}(y_i-\mu_b)^T\Sigma_b^{-1}(y_i-\mu_b)-\frac{1}{2}\log|\sum_b|\}}}$

Note : $\sum_a P_i(V_{ia} = 1 \mid \theta) = 1$. Also $P_i(V_{ia} \mid \theta)$ is biggest for the model μ_a, Σ_a closest to the data point.

Let $\sum_{V_{ia}} V_{ia} P_i(V_{ia} \mid \theta) = q_{ia}$

E-step: $Q(\theta \mid \theta^{(t)}) = -\sum q_{ia}\{\frac{1}{2}(y_i - \mu_a)^T\Sigma_a^{-1}(y_i - \mu_a) + \frac{1}{2}\log \mid \sum_a|\}$

M-step: minimize w.r.t. μ_a&$\sum_a$ gives
$\mu_a = \frac{1}{\sum_i q_{ia}}(\sum_i q_{ia} y_i) \leftarrow$ weighted sum
$\sum_a = \frac{1}{\sum_i q_{ia}} \sum_i q_{ia}(y_i - \mu_a)(y_i - \mu_a)^T$

Hence, for this case (mixture of Gaussians) we can perform the E and M steps analytically.