CS281A/Stat241A: Statistical Learning Theory

K-means and Expectation Maximization

Lecturer: Michael I. Jordan *Scribes: Ling Huang and Matthew Caesar*

1 K-means

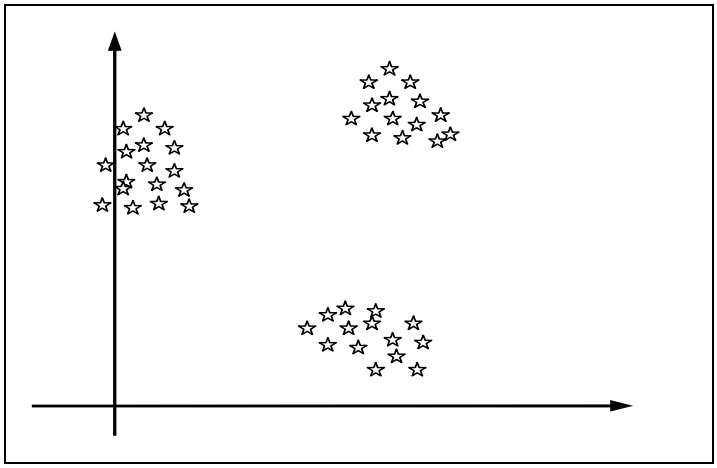

Figure 1: Example data set with three clusters.

In K-means we have a bunch of data that we think of as lying in clusters (an example is shown in Figure 1. K-means is an algorithm that takes those clusters and replaces each cluster with a centroid. You need to decide in advance how many means (i.e., how many clusters) you want. The K-means algorithm takes the following inputs:

$$data : \{x_n\}_{n=1}^{N} \tag{1}$$

$$means : \{\mu_i\}_{i=1}^{K} \tag{2}$$

You start off by initializing the μ_i's to random values (or you should do things more intelligently if possible – e.g., choosing each mean to be the value of a randomly chosen data point). K-means consists of two steps:

- The first is the assignment step. For every data point n, I set the indicator value z_n^i to either 1 or 0. I assign it to 1 if the current position of the ith mean is the closest one to data point n. That is,

$$z_n^i = \begin{cases} 1 & if \;\; i = \arg\min(||x_n - \mu_i||^2 \\ 0 & otherwise \end{cases} \tag{3}$$

  Another way to think about this step is that it partitions the space into Voronoi cells which have centroids defining them.

- The next step updates the means. For each μ_i, we set it to the weighted sum of all the data in its cell. This sets μ_i to the weighted mean of all the data points that we assigned to the ith centroid in the previous step. This causes the means to readjust to new locations.

$$\mu_i = \frac{\sum_n z_n^{i(t)} x_n}{\sum_n z_n^{i(t)}} \tag{4}$$

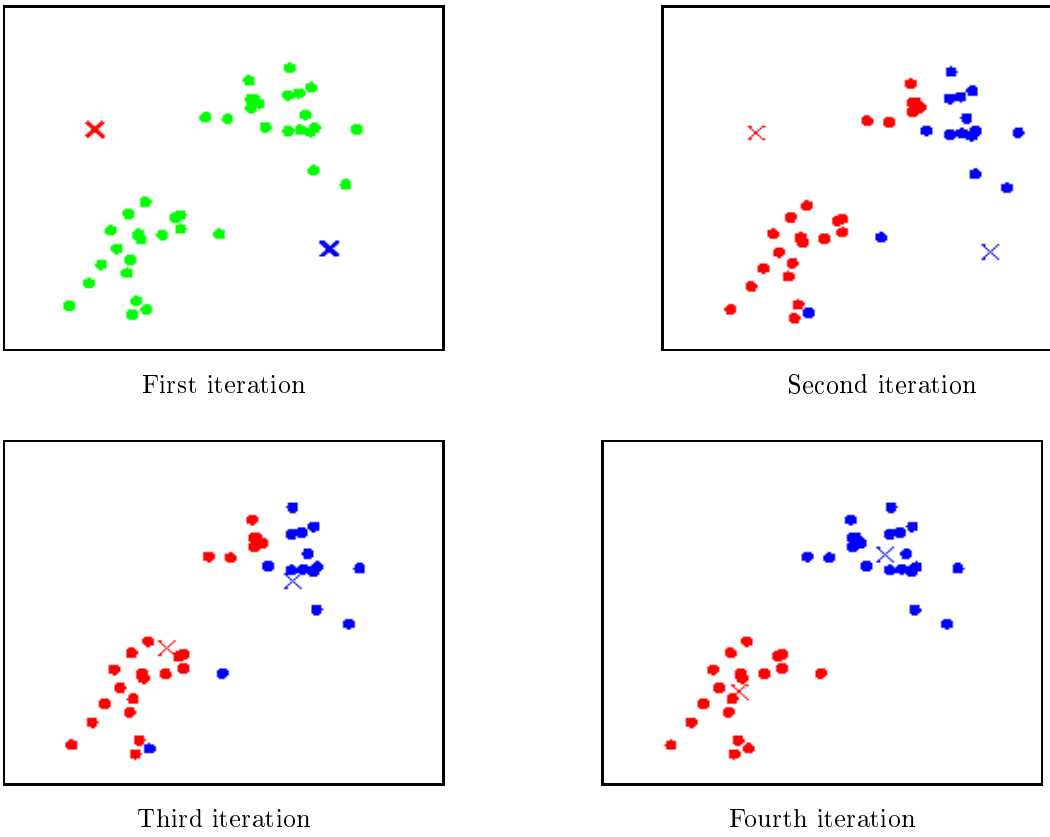

Figure 2: An example run of the K-means algorithm.

We then repeat the two above steps, the means hop around, and they eventually converge, as shown in Figure 2. You guess your initial values of your means, you see which data points are closest to those means, then you move the means even closer to the data points they're already close to.

It's a very short hop from there to an algorithm for mixture models. This is called the EM algorithm for Gaussian mixtures. EM is a general principle, hardly even an algorithm. The algorithm below is a special case of EM for Gaussian mixtures.

As a heuristic argument, we will take K-means and "soften it up". Instead of making z's be 1 or zeros, we will make them "soft" by measuring the distance to the means and take the exponential of that. We will

turn the z variable into a random variable. Instead of calculating z, I'm going to take the expected value of z conditional on the data.

- *(E Step)* The first steps assigns all data points x_n to the means z_i in a soft way. If a mean is close to a particular data point, then that data point has a large weight for you. In particular, we calculate τ. This is called the E-step:

$$\tau_n^{i(t)} = \frac{\pi_i^{(t)} N(x_n|\mu_i^{(t)}, \Sigma_i^{(t)})}{\sum_j \pi_j^{(t)} N(x_n|\mu_j^{(t)}, \Sigma_j^{(t)})} \tag{5}$$

$$N(x_n|\mu_i^{(t)}, \Sigma_i^{(t)})) \Rightarrow e^{-\frac{1}{2}(x_i-\mu_i)^T \Sigma^{-1}(x_i-\mu_i)} \tag{6}$$

  where t is the iteration number, N is the Gaussian distribution.

  We will show that this has the interpretation as a conditional expectation of the random variable z_n^i conditional on the data point x_n, i.e., the above is equal to $E[z_n^i = 1 \mid x_n, \theta^{(T)}]$.

- *(M Step)* We then calculate μ_i as the sum over the data points, of how close they are to the ith mean, times a weighting factor τ_n^i. τ_n^i is the posterior probability that data point x_n is closest to the ith mean. We then normalize by the weighting factor.

$$\mu_i^{(t+1)} = \frac{\sum_n \tau_n^{i(t)} x_n}{\sum_n \tau_n^{i(t)}} \tag{7}$$

  Note this is the same as the second step of the K-means algorithm, with τ substituted for z_n^i.

You can use this same algorithm for covariance matrices. You have the same weighted sum, but instead of x_n you have $(x_n - \mu)(x_n - \mu)^T$. That is, you first find all the data close to a given mean, then use that data to form a covariance estimate.

EM is a little slower than the K-means, since the z's are not just 1 or 0. In practical problems, K-means algorithm is often appropriate since it is quick and easy to implement. In K-means, the stopping condition is that the Voronoi cells don't change between steps. In EM, the stopping condition is that the log likelihood doesn't change between steps. Or, you can run until you get a "good enough" answer.

2 Gaussian Mixtures

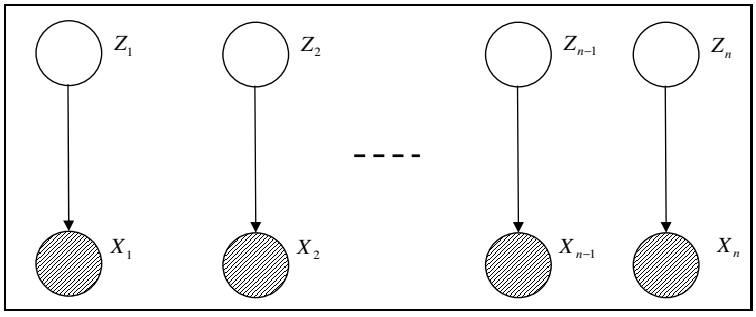

Figure 3: Graphical model of a Gaussian mixture.

Looking at a simple Gaussian mixture model will give us some insights into the EM algorithm. Figure 3 shows the graphical model. We have a unobserved data z_n and observed data x_n. We have $P(z_n^i = 1) = \pi_i$, and the x_n are selected from a Gaussian distribution conditional on $z_n^i i$, i.e. $(x_n, z_n^i)\ N(\mu_i, \sigma_i)$.

Let's calculate the expected value of the indicator variable:

$$E(z_n^i | x_n, \theta^{(t)}) = p(z_n^i = 1 | x_n, \theta^{(t)}) \tag{8}$$

The first step is obtained by observing that z is an indicator variable, and the probability of an indicator variable is the probability that it's equal to 1, that is, $E(z = 1) = 1 * p(z = 1) + 0 * p(z = 0)$. The second step is obtained by Bayes' rule, and the third is obtained by definition.

$$E(z = 1) = 1 * p(z = 1) + 0 * p(z = 0) \tag{9}$$

$$= \frac{p(z_n^i | \theta^{(t)}) p(x_n | z_n^i, \theta(t))}{N} \tag{10}$$

$$= \frac{\pi_i^{(t)} N(x_n | \mu_i^{(t)}, \Sigma^{(t)})}{N} \tag{11}$$

Hence, the E-step is just probabilistic inference. That is, we're just running a probabilistic inference algorithm to estimate the unknown variable z given the observed values x. Note that each z_n node is connected to only one x_n node, and hence is independent of all the other z_ns and x_ns.

3 General recipe for EM

The following is a general recipe for EM:

- Write down the *expected complete log likelihood* l_c. The expected complete log likelihood is the log likelihood based on inferring the values of unobserved random variables, as discussed below.
- Take derivatives to maximize l_c with respect to the parameters. This step is typically trivial (just calculus).

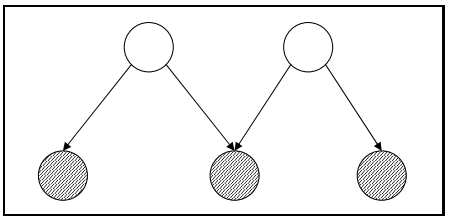

Figure 4: We start with a linear model in which certain nodes are observed (shaded nodes).

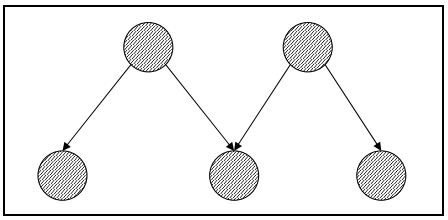

Figure 5: We infer unobserved nodes and treat them as if they were actually observed.

The basic idea is as follows. Consider the arbitrary graphical model shown in Figures 4 and 5. Several nodes are observed, and several are not. We run our elimination algorithm on the graph to calculate the probabilities of the unobserved nodes (i.e. simply do inference), and then fill them in with their estimated values from doing that calculation. Now, we have a fully observed graph, as shown in Figure /reffig:znobs We

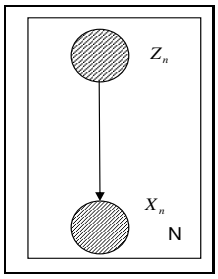

Figure 6: The Gaussian mixture graphical model, where we pretend all nodes are observed.

have previously learned how to handle this case: we simply find the values that maximize the log likelihood of this new graph.

We now proceed to derive the EM equations based on this general recipe for the Gaussian mixture model presented above:

$$\prod_n p(x_n, z_n|\theta) = \prod_n p(z_n|\theta)p(x_n|z_n, \theta) \tag{12}$$

The complete log likelihood breaks apart. z_n is a multinomial random variable, so we can write its probability as follows:

$$l_c(\{x_n, z_n\}_{n=1}^N|\theta) = \log[\prod_n p(x_n, z_n|\theta)] = \sum_n \log p(z_n|\theta) + \sum_n \log p(x_n|z_n, \theta) \tag{13}$$

$$p(z_n|\theta) = \prod_j \pi_j^{z_n^j} \tag{14}$$

$$p(x_n|z_n, \theta) = \prod_j N(x_n|\mu_j, \Sigma_j)^{z_n^j} \tag{15}$$

$$l_c = \sum_n \sum_j z_n^j \log \pi_j + \sum_n \sum_j z_n^j \log N(x_n|\mu_j, \Sigma_j) \tag{16}$$

recall the z_n^i indicator variables are used to pick out the θ_j for which $z_n^j = 1$.

This should look familiar to you – when we did classification, we did exactly this (recall the naive Bayes model). We had a log likelihood split into two terms: one with a prior, and one with a μ and σ. If you take derivatives of this quantity with respect to π, μ, and σ, you will get an estimation polynomial with π as are the proportions and μ, and σ will be the means and covariances within the respective classes.

However, we don't really know z_i – it's a random variable! Hence, l_c is a random variable as well. We are working with the expected complete log likelihood, which breaks apart into two terms that are easy to maximize over. So, we need some way to go uphill on a random function. There are several ways to do this, one way is to go up on the *average* of l_c, where *average* means "conditional expectation of $\{z_n\}$ with respect to $\{x_n\}$".

Note: you could also use a random sample instead of taking the average, this algorithm is called "data augmentation". There's another algorithm called EIP that goes back and forth between taking a sample, and then updating the parameters as if you had complete data. Note: We won't necessarily reach the global optimum if there is local optima.

We calculate the expected value of l_c as follows: we want to calculate $E(.|x_1, x_2, \ldots, x_n, \theta^{(t)})$. As shorthand, we will write this as $< . >_{\{x_n\},\theta^{(t)}}$

so,

$$E(.|x_1, ..., x_N, \theta^{(t)}) =< . >_{x_n,\theta^{(t)}} \tag{17}$$

$$< l_c >_{\{x_n\},\theta^{(t)}} = \sum_n \sum_j < z_n^j >_{\{x_n\},\theta^{(t)}} \log \pi_j + \sum_n \sum_j < z_n^j >_{\{x_n\},\theta^{(t)}} \log N(x_n|\mu_j, \Sigma_j) \tag{18}$$

Note that the expectation only affects the z_ns, so it comes all the way inside, past the summations. We then need to compute the expected value of z_n, given all the parameters, which we do as follows:

$$< z_n^j >_{\{x_n\},\theta^{(t)}} = E(z_n^j|x_1, ..., x_N, \theta^{(t)}) = p(z_n^j|x_1, ..., x_n, \theta(t)) = \tau_n^j(t) \tag{19}$$

Now, we have a final solution for l_c. By the graphical model d-seperation property, the expected value of z_n given all the data is simply $< z_n >_{x_n}$. We will also need to calculate the value $< z_n x_n >$, but we are conditioning on x_n, so we can find this value as well. The right way to think about this is that we're calculating the expected values of the sufficient statistics conditional on the data.

I won't do the M-step because it's trivial. We have the expected value of the log likelihood, so we just take derivatives and set it equal to zero. That is, take derivatives of $< l_c >$ with respect to π, μ, and σ_j, and you will get the following results:

$$\mu_i^{(t+1)} = \frac{\sum_n \tau_n^{i(t)} x_n}{\sum_n \tau_n^{i(t)}} \tag{20}$$

$$\Sigma_i^{(t+1)} = \frac{\sum_n \tau_n^{i(t)} (x_n - \mu_i^{(t+1)})(x_n - \mu_i^{(t+1)})^T}{\sum_n \tau_n^{i(t)}} \tag{21}$$

$$\pi_i^{(t+1)} = \frac{\sum_n \tau_n^{i(t)}}{N} \tag{22}$$

μ_i is like a "soft" count of, for each data point, how likely it is to be from the ith class. It is a "soft proportion" of the number of data points from the ith class.

4 Expectation Maximization II

EM has a long history – the first algorithms that did this sort of thing arose back in the '50s. In the 70s people began to realize that there was some central principle. People continue to reinvent it today.

Books tend to present this algorithm in two steps, an "expectation" step, and a "maximization" step. A better way to think about this is that it is really a coordinate ascent algorithm. The expectation step should really be thought of as a maximization step as well, so we're really doing maximization along two different variables, i.e. coordinate ascent.

- E-step: compute the expected complete log likelihood. We will show that we can think of this as a "maximization" in the appropriate variable.

- M-step: maximize the expected complete log likelihood.

To do this we make the following definitions:

- X: totality of observable variables (the data). The algorithm is not limited to iid data, so the X's don't necessarily form a sequence.
- Z: totality of latent variables (hidden variables)
- θ: totality of all parameters

Let's consider the probability model $p(x, z|\theta)$, that is, the probability of all the all the observed and unobserved variables conditional on the parameters. The complete log likelihood is the likelihood of the parameters you would have if you could see all the data, i.e. $l_c(\theta|x, z)$, which is equal to $log(p(x, z|\theta)$ by definition. To be clear, when working with the log likelihood l, recall that $l(\theta|x) = log(p(x|\theta) = \log \sigma_z p(x, z|\theta)$. So our goal is to find a $\theta^* \in \arg\max_\theta l(\theta|x)$, that is, to maximize the log likelihood.

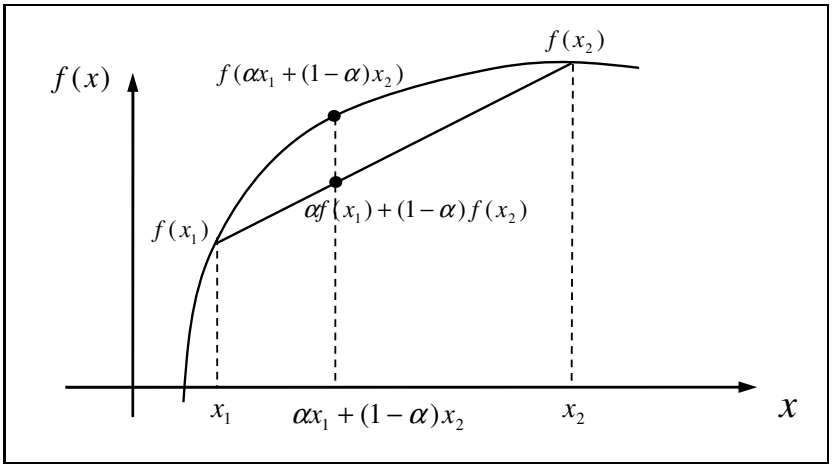

Figure 7: Jensen's inequality.

4.1 Jensen's inequality

To do this we need to know Jensen's inequality [2], which is a consequence of convexity. Later I will derive it from conjugate duality, but you can look it up in an elementary calculus text. As shown in Figure 7. First, note that any chord of a concave function lies below the function. In particular, we can consider calculating the average along the chord, and the average along the function itself. The average along the chord will certainly be less than that along the function (since the function is concave. We can write this out as follows:

$$f(\sum_{i=1}^{K} \alpha_i x_i) \geq \sum_{i=1}^{K} \alpha_i f(x_i) \tag{23}$$

4.2 Application of Jensen's inequality

Now I will use Jensen's inequality to show that the Kullback-Liebler (KL) divergence $D(p||q)$ is greater than or equal to zero, and is equal to zero iff $p = q$. Here, we will use Jensen's inequality where the function f is equal to the log function, which is convex. Recall that the KL divergence is defined as:

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)} \tag{24}$$

We start with the definition of log likelihood:

$$l(\theta|x) = \log p(x|\theta) = \log \sum_z p(x, z|\theta) \tag{25}$$

I will now multiply and divide by an arbitrary distribution on z conditional on x, which I will call $q(z|x)$. For each fixed x, $q(z|x)$ is like the α_is in Jensen's inequality.

$$\log \sum_z q(z|x) \frac{p(x, z|\theta}{q(z|x)} \tag{26}$$

We would like to bring the log inside past the sum. We can do this with Jensen's inequality at the expense of a bound.

$$\geq \sum_z q(z|x) \log \frac{p(x, z|\theta)}{q(z|x)} = \sum_z q(z|x) \log p(x, z|\theta) - \sum_x q(z|x) \log q(z|x) \tag{27}$$

The first term is the expected complete log likelihood with respect to some unknown distribution q. The second term is the entropy of the q distribution. I will call the first term (the expected complete log likelihood) $L(q, \theta)$.

The EM algorithm is now simply coordinate ascent:

- E-step We calculate the q distribution - so, we calculate the maximum over all possible q distributions of the L function, where the second argument (q) is being held fixed.
- M-step Then, we maximize over the other argument (θ) of the L function.

We can draw this as a 3-D picture. We have a direction q and a direction θ. The EM algorithm first goes uphill in the q direction, and then goes uphill in the θ direction.

We can show that the above "maximization-maximization" algorithm is the same as the EM algorithm presented at the beginning of lecture. Let's consider E-step and the M-step separately. To show the M-step is the same, we want to maximize $L(q^{(t)}, \theta)$ with respect to θ. But this is equivalent to maximizing $\sigma_z q(z|x) \log p(x, z|\theta)$ with respect to θ, which is the determinant of the M-step given earlier. Hence we're done.

We will show that the E-step is the same in the next lecture, that is, we will show that the E-step is equivalent to $q^{(t+1)}(z|x) = p(z|x, \theta^{(t)})$

The technique above of introducing a parameter and then maximizing over it is called "Lagrangian relaxation," [1] or "variational calculus".

References

[1] C. Lemarechal, "Lagrangian Relaxation," to be published in Lecture Notes in Computer Science. *http* : *//www.optimizationonline.org/DB_HTML/*2001/03/298.*html*

[2] "Convexity and Jensen's inequality," *http* : *//www.engineering.usu.edu/classes/ece/*7680/*lecture*2/*node*5.*html*. Defines Jensen's inequality and gives an example application to information theory.

[3] "The Expectation-Maximization Algorithm," *http* : *//www.owlnet.rice.edu/* ~ *msmiley/elec*431/*em.htm*. Gives a simple overview of the EM algorithm, and an example application to computer vision.