67750: Advanced Practical Machine Learning 2017/8

MLE, GMM & EM

TA: Daniel Gissin

1 Maximum Likelihood Estimation

1.1 Estimating a Coin's Bias

Say we have a coin that when tossed has a chance of p to turn up heads. We want to estimate p.

So we toss the coin 1000 times, and we end up getting 379 heads. If you had to guess p and you didn't have a reason to assume it should be anything in particular, you would probably guess $p = \frac{379}{1000}$...

1.2 The Estimation Problem

Let's generalize the above problem. Say we have a sample S of N data points, and we want to learn the data's distribution D such that $S \sim D^N$. To do that, we need some prior knowledge:

We will assume that D is a member of a familiy of probability functions, parameterized byθ. We will then look for the D_θ which best explains the given data. Intuitively, we would like to find the distribution which gives our sample the highest probability mass. The Likelihood function helps us do just that - given our data points S and our parameter θ, the Likelihood function returns the probability mass of drawing the sample S, from D_θ:

$$L(S, \theta) = Pr(S \sim D_\theta^N) = \prod_{i=1}^{N} p(x_i; \theta)$$

So maximizing the likelihood function with respect to θ should give us the best distribution D_θ.

1.3 The Log-Likelihood Trick

The Likelihood function is multiplicative, which will make differentiating it quite annoying. Since the logarithm is a monotonically increasing function, it is usually easier to work with the Log-Likelihood function, which has the same argmax as the Likelihood function:

$$LL(S, \theta) = log(L(S, \theta)) = log(\prod_{i=1}^{N} p(x_i; \theta)) = \sum_{i=1}^{N} log(p(x_i; \theta))$$

$$\hat{\theta}_{MLE} = argmax_\theta(LL(X, \theta))$$

1.4 Returning to the Coin Toss

Let's solve our original problem using the MLE method. We have N coin tosses, and k turned out heads. Our family of distribution functions will be all Bernoulli distributions, $x_i \sim Ber(\theta)$, and our Likelihood function will be:

$$L(k, \theta) = \Pi_{i=1}^{k} \theta \cdot \Pi_{i=1}^{N-k} (1 - \theta) = \theta^k (1 - \theta)^{N-k}$$

Our Log Likelihood function will be:

$$LL(N, k, \theta) = k \cdot log(\theta) + (N - k) \cdot log(1 - \theta)$$

Taking the derivative w.r.t. θ gives us our MLE:

$$0 = \frac{\partial LL}{\partial \theta} = \frac{k}{\theta} - \frac{N - k}{1 - \theta} \rightarrow \hat{\theta}_{MLE} = \frac{k}{N}$$

Which is indeed what we would have guessed intuitively...

1.5 Caveats of the MLE

The MLE is a great tool for estimating the distribution that generated our data, but it has it's limitations. Specifically for our example, say we didn't have 1000 coin tosses but only had 3. An eighth of the time we would say that a fair coin will always turn out Heads...

This is a general problem with MLE, which shows poor results for small samples.

1.6 MLE for a Gaussian Model

Let's look at another canonical example of Maximum Likelihood Estimation - the 1-dimensional Gaussian.

We are given N data points, the height of randomly selected people in Israel, and we assume that the heights have a Gaussian distribution. We would like to find the most likely parameters for the mean and variance.

First, we will write down the Likelihood function for our data:

$$L(S, \mu, \sigma^2) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i - \mu)^2}{2\sigma^2}} = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} e^{-\frac{1}{2\sigma^2} \sum_{i=1}^{N} (x_i - \mu)^2}$$

We want to maximize the Likelihood for both variables - μ and σ^2.

For μ we don't even have to look at the Log Likelihood, since maximizing the Likelihood w.r.t. μ simply means minimizing the exponent, which is simple:

$$\min_{\mu} (\frac{1}{2\sigma^2} \sum_{i=1}^{N} (x_i - \mu)^2) \rightarrow \min_{\mu} (\sum_{i=1}^{N} (x_i - \mu)^2)$$

$$\frac{\partial}{\partial\mu}(\sum_{i=1}^{N}(x_i-\mu)^2)=0=2(\sum_{i=1}^{N}x_i-\sum_{i=1}^{N}\mu)\rightarrow\hat{\mu}=\frac{\sum_{i=1}^{N}x_i}{N}=\bar{x}$$

So as we might have expected, our mean turned out to be the empirical mean of our sample. Now we move on to the variance, so we'll use the Log Likelihood:

$$LL(S,\mu,\sigma)=-\frac{N}{2}log(2\pi)-Nlog(\sigma)-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2$$

Simply differentiate w.r.t. σ to get our MLE:

$$\frac{\partial LL}{\partial\sigma}=0=-\frac{N}{\sigma}+\frac{1}{\sigma^3}\sum_{i=1}^{N}(x_i-\mu)^2\rightarrow\hat{\sigma^2}=\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2=\frac{1}{N}\sum_{i=1}^{N}(x_i-\bar{x})^2$$

Also as we might have expected, we got the empirical varince as our MLE.

Finally, had we calculated the MLE of a multidimensional Gaussian, we would have gotten the following results (d is the dimension of our data):

$$p(x)=\frac{1}{\sqrt{(2\pi)^d det(\Sigma)}}e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

$$\hat{\mu}=\frac{\sum_{i=1}^{N}x_i}{N}=\bar{x}$$

$$\hat{\Sigma}=\frac{1}{N}\sum_{i=1}^{N}(x_i-\bar{x})(x_i-\bar{x})^T$$

2 The Gaussian Mixture Model

But what can we do if our data doesn't seem to look like any of the parameterized models we've learned?

For instance, we assumed that the heights of the people in Israel have a Gaussian distribution, but it would seem more likely that the men and the women have seperate Gaussian distributions. So actually, the distribution of heights is bimodal, and none of our standard distributions can express this kind of data very well...

The next natural step for us in expanding our family of distributions' expressivity would be to allow for a combination of distributions. In figure 1 we can see that if we define a family of distributions that contains three 1D Gaussians, then that could explain the data well. As for the heights example, if we allowed for a combination of two seperate Gaussians, then the data could be better explained.

We introduce a **Latend Random Variable** to our model, which decides which Gaussian is picked when we draw a sample. This means our model assumes that each data point is generated by first selecting a Gaussian out of the k possible Gaussians (using a multinomial distribution), and then sampling from the relevant Gaussian. For a single data point x:

$$P[X=x]=\sum_{y=1}^{k}P[Y=y]P[X=x|Y=y]$$

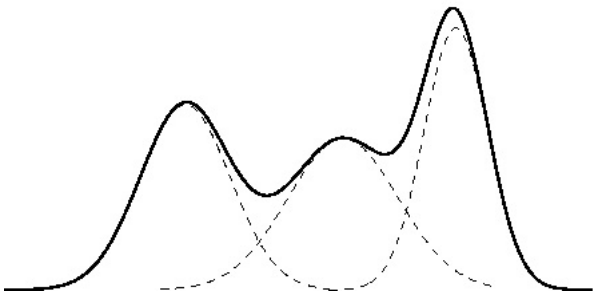

Figure 1: A Mixture of 3 1D Gaussians

We know how to calculate each of the k values, since they are each a product of a multinomial distribution and a Gaussian distribution:

$$P[Y = y] = \pi_y$$

$$P[X = x] = \sum_{y=1}^{k} \pi_y N(X = x|\mu_y, \Sigma_y) = \sum_{y=1}^{k} \pi_y \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_y|^{\frac{1}{2}}} exp(-\frac{1}{2}(x - \mu_y)^T \Sigma^{-1}(x - \mu_y))$$

So all we need to do now is find the optimal parameters for the multinomial and Gaussian distributions, which maximize the Log-Likelihood:

$$\hat{\theta}_{MLE} = argmax(\sum_{i=1}^{N} log(\sum_{y=1}^{k} \pi_y N(x_i; \mu_y, \Sigma_y))$$

Sadly, the summation inside the logarithm makes this intractable, which means we need a good optimization algorithm. Gradient Descent could work here, but calculating the gradient for each small step requires going over the entire data set, and we want something faster.

3 The Expectation Maximization Algorithm

3.1 Had we known the assignment of each point to each Gaussian

While the above optimization objective isn't tractable, had we known which Gaussian produced each sample, we could estimate each Gaussian seperately using MLE. We could simply define an indicator variable $z_{i,y}$ which equals 1 if data point x_i came from Gaussian y. Now we can simply estimate every Gaussian seperately:

$$\pi_y = \frac{1}{N} \sum_{i=1}^{N} z_{i,y}$$

$$\mu_y = \frac{\sum_{i=1}^{N} z_{i,y} x_i}{\sum_{i=1}^{N} z_{i,y}}$$

$$\Sigma_y = \frac{\sum_{i=1}^{N} z_{i,y}(x_i - \mu_y)(x_i - \mu_y)^T}{\sum_{i=1}^{N} z_{i,y}}$$

3.2 Had we known the Gaussian parameters

On the other hand, had we not known the assignments but had known the parameters of the Gaussians and latent variable, we would have been able to calculate the probability that each data point came from each Gaussian:

$$c_{i,y} = Pr(Y = y|x_i; \theta, \pi) = \frac{\pi_y N(x_i; \mu_y, \Sigma_y)}{\sum_{i=1}^{k} \pi_i N(x_i; \mu_i, \Sigma_i)}$$

If we define an indicator random variable $z_{i,y}$ like before, which takes the value of 1 if data point x_i came from Gaussian y, we can express the log likelihood of our data as a random variable:

$$LL(S, Z; \theta, \pi) = const + \sum_{i=1}^{N} \sum_{y=1}^{k} z_{i,y}(log(\pi_y) + log(N(x_i; \mu_y, \Sigma_y)))$$

Since the z's are multinomial random variables, we can calculate the expectation of the log likelihood easilly:

$$\mathbb{E}_z[LL(S, Z; \theta, \pi)] = const + \sum_{i=1}^{N} \sum_{y=1}^{k} c_{i,y}(log(\pi_y N(x_i; \mu_y, \Sigma_y)))$$

And now maximizing the expectation becomes tractable:

$$\pi_y = \frac{1}{N} \sum_{i=1}^{N} c_{i,y}$$

$$\mu_y = \frac{\sum_{i=1}^{N} c_{i,y} x_i}{\sum_{i=1}^{N} c_{i,y}}$$

$$\Sigma_y = \frac{\sum_{i=1}^{N} c_{i,y}(x_i - \mu_y)(x_i - \mu_y)^T}{\sum_{i=1}^{N} c_{i,y}}$$

3.3 Combining everything to form an iterative algorithm

Sadly, we don't know both the assignment of points to Gaussians and the Gaussian parameters. The EM algorithm decouples these two estimation tasks, each time thinking of one of them as known and maximizing the other. The E-step of the algorithm calculates the expectation of the assignment problem (the c's), and the M-step maximizes the expectation of the log likelihood (finding the parameters of the Gaussians and latent variable). This is done for a predefined amount of iterations, or until the log likelihood converges to the maximum.

This is very similar to K-Means' decoupling of the assignment of points to clusters from the calculation of the centroids, and indeed the EM algorithm with a Gaussian Mixture Model is often referred to as "soft K-Means".

3.4 Initialization

Note that like K-Means, EM is also susceptible to local optimum. We will learn later in this course of ways to initialize K-Means better, but for now you can initialize EM in whichever way you prefer. Just make sure your initialization tactic allows the algorithm to find a good optimum (for example, initializing two Gaussians in the same way isn't a good idea).

3.5 The Algorithm

Algorithm 1 The EM Algorithm for a Gaussian Mixture Model

1: Initialize Gaussian and Latent Variable Parameters
2: **while** not converged **do**
3: $\quad c_{i,y} = \frac{\pi_y N(x_i;\mu_y,\Sigma_y)}{\sum_{l=1}^{k} \pi_l N(x_i;\mu_l,\Sigma_l)}$
4: $\quad \pi_y = \frac{1}{N}\sum_{i=1}^{N} c_{i,y}$
5: $\quad \mu_y = \frac{\sum_{i=1}^{N} c_{i,y} x_i}{\sum_{i=1}^{N} c_{i,y}}$
6: $\quad \Sigma_y = \frac{\sum_{i=1}^{N} c_{i,y}(x_i-\mu_y)(x_i-\mu_y)^T}{\sum_{i=1}^{N} c_{i,y}}$
7: **end while**

4 Working in Log-Space

Working with likelihood functions can quickly cause numerical problems, since the probabilities are very, very small. This means that to have numerical stability when implementing EM, you will need to work in log-space when making your calculations. For instance, if you need to calculate the assignment of each point in the E-step:

$$c_{i,y} = Pr(Y = y|x_i;\theta,\pi) = \frac{\pi_y N(x_i;\mu_y,\Sigma_y)}{\sum_{i=1}^{k} \pi_i N(x_i;\mu_i,\Sigma_i)}$$

you will need to do it in log space, calculating each of the log probabilities $log(\pi_y N(x_i;\mu_y,\Sigma_y))$, then normalizing in log space using the logsumexp function, and finally taking the exponent to move back from log space. The logsumexp function basically does the following calculation in a numerically stable way:

$$logsumexp(x) = log(\sum_{i=1}^{N} e^{x_i})$$

You will be supplied with an example in the exercise and a function that normalizes a matrix in log space.