Week 9: Expectation Maximization

Instructor: Sergey Levine

1 EM Recap

Last week, we saw how we could represent clustering with a probabilistic model. In this model, called a Gaussian mixture model, we model each datapoint $\mathbf{x}_i$ as originating from some cluster, with a corresponding cluster label y_i distributed according to $p(y)$, and the corresponding distribution for that cluster given by a multivariate Gaussian:

$$p(\mathbf{x}|y=k) = \frac{1}{\sqrt{(2\pi)^D|\Sigma_k|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_k)^T\Sigma_k^{-1}(\mathbf{x}-\mu_k)\right).$$

Our goal is to maximize the (log) likelihood of the data, as usual, which is given by

$$\mathcal{L} = \prod_{i=1}^N p(\mathbf{x}_i),$$

but because we don't know the cluster labels y_i associated with each datapoint $\mathbf{x}_i$, we must marginalize them out:

$$\mathcal{L} = \prod_{i=1}^N p(\mathbf{x}_i) = \prod_{i=1}^N \sum_{k=1}^K p(\mathbf{x}_i, y_i = k) = \prod_{i=1}^N \sum_{k=1}^K p(\mathbf{x}_i|y_i=k)p(y_i=k).$$

The resulting objective cannot be optimized in closed form, and while we could use gradient ascent, we saw a much more convenient and effective algorithm called expectation maximization (EM):

Algorithm 1 EM

```
1: Initialize means and covariances
2: while not converged do
3:     E-step: estimate $w_{ik}$ for each datapoint $i$ and each cluster $k$
4:     M-step: fit $\mu_k$ and $\Sigma_k$ using the weighted MLE fit
5: end while
```

In EM, we explicitly estimate a distribution $p(y_i = k|\mathbf{x}_i) = w_{ik}$ during the E-step, and then maximize the expected log-likelihood in the M step with respect to the model parameters. The expected log-likelihood is given by

$$\hat{\mathcal{L}} = \sum_{i=1}^{N} \sum_{k=1}^{K} w_{ik} \log p(\mathbf{x}_i, y_i = k).$$

The expected log-likelihood is *not* the same as the marginal (log) likelihood! However, it is easy to see that EM maximizes the expected log-likelihood if we decouple w_{ik} and $p(\mathbf{x}_i, y_i = k)$ in this way. So then the question is: does EM also maximize the marginal likelihood?

2 What Does EM Optimize?

To understand what EM optimizes, we'll introduce a little bit of notation for clarity. When we update w_{ik}, we set $w_{ik} = p(y_i = k|\mathbf{x}_i)$. However, in the E-step, we hold w_{ik} fixed, even though we change $p(\mathbf{x}_i, y_i)$. To make it a bit clearer which model is being used in each step, we'll condition the probability on the parameter values $\theta = \{\mu_1, \Sigma_1, \ldots, \mu_K, \Sigma_K\}$, where θ denotes the old parameters during a single iteration, and θ' denotes the new parameters we get at the end of the M-step. Therefore, we have $w_{ik} = p(y_i = k|\mathbf{x}_i, \theta)$. The expected log-likelihood is therefore given by

$$\hat{\mathcal{L}}(\theta') = \sum_{i=1}^{N} \sum_{k=1}^{K} p(y_i = k|\mathbf{x}_i, \theta) \log p(\mathbf{x}_i, y_i = k|\theta') = \sum_{i=1}^{N} E_{y_i \sim p(y_i|\mathbf{x}_i, \theta)}[\log p(\mathbf{x}_i, y_i|\theta')],$$

hence the name expected log-likelihood.

We'll also introduce a very useful relation called Jensen's inequality. The particular form of Jensen's inequality for logarithms tells us that

$$\log \left[\sum_{k=1}^{K} w_k a_k \right] \geq \sum_{k=1}^{K} w_k \log a_k,$$

when $w_k > 1$ and $\sum_k w_k = 1$: that is, the log of a weighted sum is greater than or equal to the weighted sum of logs.

With these definitions, we can show that one iteration of EM actually optimizes the marginal log-likelihood $\mathcal{L}$, where we use $\mathcal{L}(\theta)$ to denote the log-likelihood under the old parameters, and $\mathcal{L}(\theta')$ under the new parameters, which is what we are trying to find. To recap, the marginal log-likelihood is given by

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \log p(\mathbf{x}_i|\theta) = \sum_{i=1}^{N} \log \left[\sum_{k=1}^{K} p(\mathbf{x}_i, y_i = k|\theta) \right]$$

We would like an algorithm that *increases* the marginal log-likelihood at each iteration, since that's the log-probability of the data (while the expected log-likelihood is just some nonsense we made up). So, specifically, we would like to have

$$\mathcal{L}(\theta') > \mathcal{L}(\theta),$$

so we want to maximize $\mathcal{L}(\theta') - \mathcal{L}(\theta)$. We can write the difference between the new and old log-likelihood as following (we will omit the sum over i for convenience, but it's always there):

$$\begin{aligned}
\mathcal{L}(\theta') - \mathcal{L}(\theta) &= \log\left[\sum_{k=1}^{K} p(\mathbf{x}_i|y_i = k, \theta')p(y_i = k|\theta')\right] - \log p(\mathbf{x}_i|\theta) \\
&= \log\left[\sum_{k=1}^{K} p(\mathbf{x}_i|y_i = k, \theta')p(y_i = k|\theta')\frac{p(y_i = k|\mathbf{x}_i, \theta)}{p(y_i = k|\mathbf{x}_i, \theta)}\right] - \log p(\mathbf{x}_i|\theta) \\
&= \log\left[\sum_{k=1}^{K} p(y_i|\mathbf{x}_i, \theta)\frac{p(\mathbf{x}_i|y_i = k, \theta')p(y_i = k|\theta')}{p(y_i = k|\mathbf{x}_i, \theta)}\right] - \log p(\mathbf{x}_i|\theta).
\end{aligned}$$

Now we'll apply Jensen's inequality to the first part of this equation. Jensen's inequality tells us that the log of a weighted sum is greater than or equal to the weighted sum of the log of the thing being summed. Here, we'll choose $p(y_i = k|\mathbf{x}_i, \theta)$ as the weight. We know that these sum to 1, because they are probabilities, so we have

$$\begin{aligned}
\mathcal{L}(\theta') - \mathcal{L}(\theta) &= \log\left[\sum_{k=1}^{K} p(y_i = k|\mathbf{x}_i, \theta)\frac{p(\mathbf{x}_i|y_i = k, \theta')p(y_i = k|\theta')}{p(y_i = k|\mathbf{x}_i, \theta)}\right] - \log p(\mathbf{x}_i|\theta) \\
&\geq \sum_{k=1}^{K} p(y_i = k|\mathbf{x}_i, \theta)\log\frac{p(\mathbf{x}_i|y_i = k, \theta')p(y_i = k|\theta')}{p(y_i = k|\mathbf{x}_i, \theta)} - \log p(\mathbf{x}_i|\theta).
\end{aligned}$$

Now note that the last part doesn't depend on k, so we can insert a sum over k to get

$$\sum_{k=1}^{K} p(y_i = k|\mathbf{x}_i, \theta)\left[\log\frac{p(\mathbf{x}_i|y_i = k, \theta')p(y_i = k|\theta')}{p(y_i = k|\mathbf{x}_i, \theta)} - \log p(\mathbf{x}_i|\theta)\right] =$$
$$\sum_{k=1}^{K} p(y_i = k|\mathbf{x}_i, \theta)\left[\log p(\mathbf{x}_i|y_i = k, \theta')p(y_i = k|\theta') - \log p(y_i = k|\mathbf{x}_i, \theta)p(\mathbf{x}_i|\theta)\right],$$

where the last part comes from the fact that we can rewrite products of logs as sums (and ratios as differences). Note that the part of this equation that depends on θ' (the first part) is exactly the expected log-likelihood! This means that the M-step simply optimizes a bound on the difference between the new log-likelihood and the previous one. So if we can make this quantity positive, we know that we'll improve the marginal log-likelihood.

This turns out to be very simple to show: if we have $\theta = \theta'$, then the bound is exactly zero. So if we maximize the bound with respect to θ', we know that the improvement must be at least zero, since the trivial setting of $\theta = \theta'$ already achieves zero. This means that EM will never make the objective worse, and it will improve the objective unless the algorithm has converged and $\theta = \theta'$.

For notational convenience and to avoid keeping around old and new parameter values θ and θ', oftentimes people use $q(y_i = k)$ to denote $p(y_i = k|\theta)$ (the probability of a label under the old parameter values). We'll use this notation below as well.

3 EM for Missing Data

So far, we saw how EM could be used for probabilistic clustering with Gaussian mixture models. However, EM is a general algorithm for optimizing probabilistic models in missing data settings: that is, settings where some entries in y_i or $\mathbf{x}_i$ are not available during training. In the case of clustering, the hypothesis space corresponds to Gaussian mixture models, and the missing data corresponds to all of the labels y_i. In the general case of EM, the step computes a joint distribution over all missing variables for each datapoint, and the M-step maximizes the corresponding expected log-likelihood. Note that we do not distinguish at this point which variables are $\mathbf{x}$ and which are y.

First, let's introduce this idea for formally. Let $\mathcal{D} = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$. If we have labels y, let's assume for now that they are just stored in the last coordinate of $\mathbf{x}$. Then, for every datapoint, let I_i denote the indices that are known, and let J_i denote the indices that are missing. So we know $\mathbf{x}^i_{I_i}$, but do not know $\mathbf{x}^i_{J_i}$. The E-step of EM then constructs distributions of the form $q(\mathbf{x}^i_{J_i}) = p(\mathbf{x}^i_{J_i} | \mathbf{x}^i_{I_i})$ by using inference (Bayes rule). The M-step maximizes the expected log-likelihood, given by

$$\hat{\mathcal{L}} = \sum_{i=1}^{N} \sum_{\mathbf{x}^i_{J_i}} q(\mathbf{x}^i_{J_i}) \log p(\mathbf{x}^i_{I_i}, \mathbf{x}^i_{J_i})$$

We can show that this optimizes the marginal likelihood by the same procedure as before. To summarize the full algorithm:

Algorithm 2 EM – general version

```
1: Initialize model parameters $\theta$
2: while not converged do
3:     E-step: estimate $q(\mathbf{x}^i_{J_i}) \leftarrow p(\mathbf{x}^i_{J_i} | \mathbf{x}^i_{I_i}, \theta)$ based on $\theta$
4:     M-step: fit $\theta \leftarrow \arg\max_\theta \sum_{i=1}^{N} \sum_{\mathbf{x}^i_{J_i}} q(\mathbf{x}^i_{J_i}) \log p(\mathbf{x}^i_{I_i}, \mathbf{x}^i_{J_i} | \theta)$
5: end while
```

4 Example of Missing Data

To understand why working with missing data can be important, let's consider a simple example. We'll go back to a simplified version of the disease example from earlier in the quarter, and we'll try to fit a classifier with naïve Bayes, using EM to deal with missing data. Here is another patient dataset:

patient	fever? x_1	cough? x_2	disease? y
A	1	?	1
B	0	?	0
C	1	0	?
D	0	1	?

This dataset has missing data: we don't know whether patients A and B have a cough, and we don't have a diagnosis for patients C and D. What if we get a new patient who has a cough, and we don't have a thermometer on hand to diagnose them?

Question. Can we diagnose this patient?

Answer. Yes, because we know from the data that fever is anticorrelated with cough (for some reason)! In fact, if we use EM, we actually figure this out. Let's assume that we initialize all factors to be uniformly distributed, so that initially:

$$p(x_2 = 1|y = 0) = \frac{1}{2} \qquad p(x_2 = 1|y = 1) = \frac{1}{2}.$$

Then, during the E-step, we'll fill in a probability of $\frac{1}{2}$ for every assignment to x_2 and y. Then, during the next M-step, we'll fit $p(y = 1) = \frac{1}{2}$, $p(x_1 = 1|y = 1) = 1$, $p(x_1 = 1|y = 0) = 0$, and $p(x_2 = 1|y = 1) = 0 = \frac{1}{2}$.

During the next E-step, we'll then assign a higher probability that $x_2 = 0$ for patient A and a higher probability that $x_2 = 1$ for patient B! Similarly, we'll assign a higher probability that $y = 1$ for C and $y = 0$ for D. At convergence, we'll be certain that $p(x_2 = 1|y = 1) = 0$ and $p(x_2 = 1|y = 1) = 1$, meaning that only patients without a cough have the disease, all without ever having seen a diagnosed patient with a cough.

Note that this is all despite the feature independence assumption of naïve Bayes. Why? Well, in naïve Bayes, the features are only independent if y is known. Since y is not known for patients C and D, the features are in fact not independent!