Generative Models & EM

Understanding EM

*Notes by Viet Vu (2021)

1. Generative Model: Unknown parameters, known parameters, latents, and observed data

1.1. How to classify parameters and variables

The parameters and variables of any inference model can be broken down to the following:

- **Unknown parameters**: parameters that we **do not know** the exact values of, and they are used to generate variables in the model.
- **Known parameters**: parameters that we **know** the exact values of, and they are used to generate variables in the model.
- **Latents**: these variables are *generated* from the parameters (unknown/known), but we do not observe them. We also say that these variables are hidden.
- **Observed data**: these variables are *generated* from the latent variables and the parameters, and we **observe** them (in other words, knowing their values).

Example 1 (RNA-Seq) Let's say that there are M transcripts, and N reads.

- The nucleotide abundances $\nu_1, \nu_2, \cdots, \nu_M$

are **unknown parameters**. Recall that in Week 2, Kallisto attempts to estimate these parameters using the reads. However, they are *not* variables, because they do not change from experiment to experiment: we do not know the values of $\boldsymbol{\nu}$, but they are **fixed**.

- The lengths of the transcripts, $L_1, L_2, \cdots, L_M$ are **known parameters**. The lengths are very integral to the identities of the transcripts, and it is logical that we know them. For example, the length of transcript 1 is 1500 (bps), and the length of transcript 2 is 2340 (bps).

- To make every read R_n $(1 \leq n \leq N)$, we have to know which transcript it comes from, G_n, the starting point of the read on that transcript, S_n, and the orientation O_n. We call G_n, S_n, and O_n **latents**: we do not know what the variables are by just looking at R_n, but $\{G_n, S_n, O_n\}$ generate R_n.

- The reads, R_n $(1 \leq n \leq N)$ are **observed data**. The output of the RNA-Seq experiment is the reads.

- Implicitly, we also observe the nucleotide sequences of the transcripts. However, we do not explicitly specify this here because these sequences are not used for inference purposes.

Example 2 (K-Means, soft) Let's say that there are N data points, and K groups.

- The centroids of K groups, $\mu_1, \mu_2, \cdots, \mu_K$, are **unknown parameters**. If they were known, we would not have gone through such hardship in Week 5 trying to estimate them!

- The variance of each group (same for all groups), σ^2, is a **known parameter**. In Week

5, we also call $\beta = \frac{1}{2\sigma^2}$ the *stiffness* of the model.

- The group identity of each data point, z_n, is a **latent**. We do not know which group each data point belongs to, but the group identity contributes to the generation of that data point.
- The data points $x_1, x_2, \cdots, x_N$, are **observed data**.

1.2. Generative Model

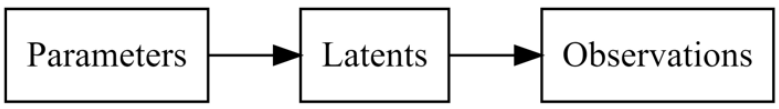
The generative model specifies the relationships between unknown parameters, known parameters, latents, and observed data.

In detail, a generative model explains the following:

- Which parameters are known and which parameters are unknown
- How the latents are generated from the parameters
- How the observed data are generated from the parameters and the latents

For our purposes, we can also think of the latents as **group assignments** of the data points. Using the parameters, we can assign each data point n to a group z_n (latent), and the data point x_n will be generated using the parameters unique to the group z_n.

The most general flowchart of a generative model is as follows:



Example (K-Means, soft) The generative model of soft K-Means answers the above questions:

- The centroids, $\mu_1, \cdots, \mu_K$, are unknown parameters, and the variance of all groups, σ^2, is a known parameter.
- For the n-th point, its group z_n is randomly chosen, with equal probabilities, among all groups:

$$P(z_n = j) = \frac{1}{K}$$

- With $z_n = j$, the data point x_n is generated from a Normal distribution with mean μ_j:

$$x_n|(z_n = j) \sim \mathcal{N}(\mu_j, \sigma^2)$$

- We can see that the latent z_n specifies the group that the n-th point belongs to, and when we already know the group, x_n will be generated by parameters unique to that group (in K-Means, it is from the centroid corresponding to group z_n).

Question: Can you specify how the generative model in lecture answer these questions about the RNA-Seq experiment as well?

2. Expectation-Maximization (EM)

\subsection{2.1. The goal of the game} In this setting, we have access to the observed data. For example, in the RNA-Seq experiment, we observe the reads $\{R_n\}_{n=1}^N$. However, we do not have access to both the unknown parameters and the latents.

Our purpose is that, from the observed data, we should have reliable estimates for both the unknown parameters and the latents.

2.2. General scheme for EM

We would notate z_n as the latent (group assignment) of the n-th data point, and x_n the n-th data point. The unknown parameters are denoted as a vector $\boldsymbol{\theta}$ (**only for this section**).

1. **Initialization**: make a guess for the unknown parameters

2. In the RNA-Seq example, we would make a guess for the nucleotide abundances vector, $\boldsymbol{\nu}$. Since the elements of $\boldsymbol{\nu}$ must sum to 1, we sample a random probability vector (check out the Dirichlet distribution in lecture).

3. In the K-Means example, we would make a guess for the centroids (this is covered in Week 5)

4. **Expectation Step**: infer the latents, given the observed data and the guessed parameters

5. In words, given a point x_n and guessed parameters $\boldsymbol{\theta}$, what is the probability that the point belongs to group k? For example, given a read R_n and nucleotide abundances $\boldsymbol{\nu}$, what is the probability that this read belongs to transcript k?

6. Mathematically, we calculate

   $$P(z_n = k | x_n, \boldsymbol{\theta})$$

   for all points n from 1 to N, and for all groups k from 1 to K.

7. At the expectation step, we always use Bayes' Rule to calculate these probabilities:

   $$q_{nk} = P(z_n = k | x_n, \boldsymbol{\theta}) = \frac{P(x_n | z_n = k, \boldsymbol{\theta}) P(z_n = k | \boldsymbol{\theta})}{\sum_{j=1}^{K} P(x_n | z_n = j, \boldsymbol{\theta}) P(z_n = j | \boldsymbol{\theta})}$$

8. The probabilities $P(z_n = j | \boldsymbol{\theta})$ and $P(x_n | z_n = j, \boldsymbol{\theta})$ are specified by the generative model.

- The first probability represents how to assign a group for point n, given the guessed parameters
- The second probability represents how to generate the observed data point n, given the group z_n and the guessed parameters.
- Remind yourself of the three questions that a generative model answers from above!

9. **Maximization Step**: given the probabilities of the latents $\{z_n\}_{n=1}^{N}$, estimate the parameters $\boldsymbol{\theta}^{\text{ML}}$ through maximum likelihood.

10. We choose to maximize the following function:

$$\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \sum_{k=1}^{K} q_{nk} \log P(x_n | z_n = k, \boldsymbol{\theta}) + q_{nk} \log P(z_n = k | \boldsymbol{\theta})$$

11. We will not do derivatives this week, so we will have to circumvent this step to find maximum likelihood estimates.

12. For the purposes of this week, the only parameters we will have to infer are the nucleotide abundances, ν. To get the maximum-likelihood estimate for $\boldsymbol{\nu}$, we need to have the estimated counts of reads to transcript k, $\hat{c}_k$ for all $1 \leq k \leq M$ (there are M transcripts, or M groups to assign reads to). The formula for the estimated counts is

$$\hat{c}_k = \sum_{n=1}^{N} q_{nk}$$

and the estimated nucleotide abundance for transcript k is

$$\hat{\nu}_k = \frac{\hat{c}_k}{N}$$

where N is the number of reads.

13. **Iterate**: We let $\boldsymbol{\theta}^{\mathrm{ML}}$ be our next guessed parameters, and go back to the expectation step.

14. **Convergence Criterion**: Somehow we have to stop the iterations; we cannot iterate forever. Here we introduce the log-likelihood of the data:

$$\mathcal{L} = \sum_{n=1}^{N} \log P(x_n|\boldsymbol{\theta}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \log P(x_n|z_n = k, \boldsymbol{\theta}) + \log P(z_n = k|\boldsymbol{\theta})$$

15. We have to **calculate using logarithms**, because there will be 100000 reads! The likelihood would be underflowed to 0 on any computer, so we have to resort to taking the logarithm of the likelihood, or the log-likelihood.

16. We iterate until $\mathcal{L}$ stops changing. A better strategy is to stop changing $\mathcal{L}$ when the change is too small:

$$|\mathcal{L}^{\mathrm{new}} - \mathcal{L}^{\mathrm{old}}| < \epsilon$$

    The threshold ϵ is usually set to anywhere around 0.01.

2.3. Strategies for implementing EM

- **Initialization**: The unknown parameters $\boldsymbol{\theta}$ can be a list, or an NumPy array. NumPy arrays are recommended.

- **Expectation Step**: Encode $\{q_{nk}\}$ ($1 \leq n \leq N, 1 \leq k \leq K$) in a list of lists, or $N \times K$ matrix.

    - For every n, make a list (array) of K elements, where the j-th element is $P(x_n|z_n = j, \boldsymbol{\theta})P(z_n = j|\boldsymbol{\theta})$. Then, normalize the list (array); in other words, divide each element of the list (array) by the sum of that list (array).

The output of this will be $\{q_{nk}\}_{k=1}^{K}$ for a fixed value of n.

- **Maximization Step**: No particular strategies; follow the formulas in Section 2.2, part 3.

- **Convergence Criterion**: To calculate the log-likelihood of the data, we should not loop from $n = 1$ to N as the formula in Section 2.2, part 5 suggests. Note that if $x_i = x_j$, then

$$P(x_i|\boldsymbol{\theta}) = P(x_j|\boldsymbol{\theta})$$

In other words, if the i-th observation and the j-th observation are the same, then their probabilities will be the same. Note that

$$\mathcal{L} = \sum_{n=1}^{N} \log P(x_n|\boldsymbol{\theta})$$

and so do we need to go through x_j in our for loop if we already know $x_i = x_j$?

Hence, when doing the problem set, you should consider this when computing the log-likelihood. Try your best not to resort to finding $\mathcal{L}$ by a for loop with N iterations; this would take too much time.
