ML Homework #3

Yoke Kai Wen, 2020280598

December 27, 2020

1 Clustering: Mixture of Multinomials

1.1 MLE for multinomial

$$P(\boldsymbol{x}|\boldsymbol{\mu}) = \frac{n!}{\prod_i x_i!}\prod_i \mu_i^{x_i} = n!\prod_i \frac{\mu_i^{x_i}}{x_i!} \tag{1}$$

Log-likelihood:

$$\begin{aligned} l(\boldsymbol{\mu}) &= log(P(\boldsymbol{x}|\boldsymbol{\mu})) \\ &= log(n!\prod_i \frac{\mu_i^{x_i}}{x_i!}) \\ &= log(n!) + log(\prod_i \frac{\mu_i^{x_i}}{x_i!}) \\ &= log(n!) + \sum_{i=1}^{d} log(\frac{\mu_i^{x_i}}{x_i!}) \\ &= log(n!) + \sum_{i=1}^{d} x_i log(\mu_i) - \sum_{i=1}^{d} x_i! \end{aligned} \tag{2}$$

Form Lagrangian with constraint $\sum_i \mu_i = 1$:

$$L(\boldsymbol{\mu}, \lambda) = l(\boldsymbol{\mu}) + (1 - \sum_{i=1}^{d} \mu_i) \tag{3}$$

Find $argmax_{\boldsymbol{\mu}} L(\boldsymbol{\mu}, \lambda)$

$$\begin{aligned} \frac{\partial}{\partial \mu_i} L(\boldsymbol{\mu}, \lambda) = \frac{\partial}{\partial \mu_i} l(\boldsymbol{\mu}) + \frac{\partial}{\partial \mu_i}(1 - \sum_{i=1}^{d} \mu_i) = 0 \\ \frac{x_i}{\mu_i} + \lambda(-1) = 0 \\ \mu_i = \frac{x_i}{\lambda} \\ \sum_{i=1}^{d} \mu_i = \sum_{i=1}^{d} \frac{x_i}{\lambda} \end{aligned} \tag{4}$$

Result for μ_i can be further simplified:

$$\begin{aligned} &LHS = 1, \quad RHS = \frac{1}{\lambda}\sum_{i=1}^{d} x_i = \frac{n}{\lambda} \quad (given) \\ &1 = \frac{n}{\lambda} \quad \longrightarrow \quad \lambda = n \\ &\mu_i = \frac{x_i}{n} \end{aligned} \tag{5}$$

1.2 EM for mixture of multinomials

Let $\boldsymbol{x}_d$ represent 1xW feature vector of document d.
Rewriting probability distribution:

$$\begin{aligned} &P(\boldsymbol{x}_d|c_d = k) = P(\boldsymbol{x}_d|\boldsymbol{\mu}_k) = n_d! \prod_{\omega=1}^{W} \frac{\mu_{\omega k}^{T_{d\omega}}}{T_{d\omega}!} \\ &\sum_{\omega=1}^{W} T_{d\omega} = n_d \\ &\sum_{\omega=1}^{W} \mu_{\omega k} = 1, \quad \forall k \\ &\sum_{k=1}^{K} \pi_k = 1 \end{aligned} \tag{6}$$

Expectation step is similar to GMM, we also introduce a q distribution for latent variable:

$$\begin{aligned} q(c_d = k) = P(c_d = k|\boldsymbol{x}_d, \boldsymbol{\pi}, \boldsymbol{\mu}) = \frac{\pi_k P(\boldsymbol{x}_d|\boldsymbol{\mu}_k)}{\sum_{j=1}^{k} \pi_j P(\boldsymbol{x}_d|\boldsymbol{\mu}_j)} &= \frac{\pi_k n_d! \prod_{\omega=1}^{W} \frac{\mu_{\omega k}^{T_{d\omega}}}{T_{d\omega}!}}{\sum_{j=1}^{K} \pi_j n_d! \prod_{\omega=1}^{W} \frac{\mu_{\omega j}^{T_{d\omega}}}{T_{d\omega}!}} \\ &= \frac{\pi_k \prod_{\omega=1}^{W} \mu_{\omega k}^{T_{d\omega}}}{\sum_{j=1}^{K} \pi_j \prod_{\omega=1}^{W} \mu_{\omega j}^{T_{d\omega}}} \end{aligned} \tag{7}$$

EM algorithm defines lower bound for log-likelihood by applying Jensen's inequality.

$$\begin{aligned} log(P(\boldsymbol{x}_d|\boldsymbol{\pi}.\boldsymbol{\mu}) = log(\sum_{k=1}^{K} q(c_d = k) \frac{P(\boldsymbol{x}_d, c_d = k|\pi_k, \boldsymbol{\mu}_k)}{q(c_d = k)}) &\geq \sum_{k=1}^{K} q(c_d = k) log(\frac{P(\boldsymbol{x}_d, c_d = k|\pi_k, \boldsymbol{\mu}_k)}{q(c_d = k)}) \\ &= L_i(\boldsymbol{\pi}, \boldsymbol{\mu}.\boldsymbol{q}) \end{aligned} \tag{8}$$

Maximisation step with respect to $\boldsymbol{\mu}, \boldsymbol{\pi}$:

First, include all documents in the lower bound for the log-likelihood:

$$
\begin{aligned}
L(\boldsymbol{\pi}, \boldsymbol{\mu}.\boldsymbol{q}) &= \sum_{d=1}^{D} L_i(\boldsymbol{\pi}, \boldsymbol{\mu}.\boldsymbol{q}) \\
&= \sum_{d=q}^{D} \sum_{k=1}^{K} q(c_d = k) log(\frac{P(\boldsymbol{x}_d, c_d = k | \pi_k, \boldsymbol{\mu}_k)}{q(c_d = k)}) \\
&= \sum_{d=1}^{D} \sum_{k=1}^{K} q(c_d = k) log(P(\boldsymbol{x}_d, c_d = k) P(\boldsymbol{x}_d | c_d = k, \pi_k, \boldsymbol{\mu}_k)) - \sum_{d=1}^{D} \sum_{k=1}^{K} q(c_d = k) log(q(c_d = k))
\end{aligned}
\tag{9}
$$

Last term in $L(\boldsymbol{\pi}, \boldsymbol{\mu}.\boldsymbol{q})$ can be dropped during maximisation because it does not contain $\boldsymbol{\pi}, \boldsymbol{\mu}$ terms, we can call it *const*. Then formulate Lagrangian optimisation problem from $L(\boldsymbol{\pi}, \boldsymbol{\mu}.\boldsymbol{q})$ with constraints $\sum_{\omega=1}^{W} \mu_{\omega k} = 1, \forall k$ and $\sum_{k=1}^{K} \pi_k = 1$.

$$
L' = \sum_{d=1}^{D} \sum k = 1^{K} q(c_d = k) log(P(c_d = k) | \pi_k, \boldsymbol{\mu}_k) - \lambda_1([\sum_{\omega=1}^{W} \mu_{\omega k}] - 1) - \lambda_2([\sum_{k=1}^{K} \pi_k] - 1) + const
\tag{10}
$$

Then we differentiate and set differential to zero, first with respect to $\mu_{\omega k}$: $\frac{\partial L'}{\partial \mu_{\omega k}} = 0$.

$$
\begin{aligned}
&\frac{\partial}{\partial \mu_{\omega k}} [(\sum_{d=1}^{D} q(c_d = k) log(\pi_k n_d! \prod_{\omega=1}^{W} \frac{\mu_{\omega k}^{T_{d\omega}}}{T_{d\omega}!})) - \lambda_1 \mu_{\omega k}] = 0 \\
&\frac{\partial}{\partial \mu_{\omega k}} [(\sum_{d=1}^{D} q(c_d = k) log(\pi_k n_d!)) + (\sum_{d=1}^{D} q(c_d = k) \sum_{\omega=1}^{W} log(\frac{\mu_{\omega k}^{T_{d\omega}}}{T_{d\omega}!})) - \lambda_1 \mu_{\omega k}] = 0 \\
&\frac{\partial}{\partial \mu_{\omega k}} [(\sum_{d=1}^{D} q(c_d = k) \sum_{\omega=1}^{W} [T_{d\omega} log(\mu_{\omega k}) - log(T_{d\omega}!)]) - \lambda_1 \mu] = 0 \\
&\frac{\partial}{\partial \mu_{\omega k}} [(\sum_{d=1}^{D} q(c_d = k) T_{d\omega} log(\mu_{\omega k})) - \lambda_1 \mu_{\omega k}] = 0 \\
&\frac{\sum_{d=1}^{D} q(c_d = k) T_{d\omega}}{\mu_{\omega k}} - \lambda_1 = 0 \\
&\mu_{\omega k} = \frac{\sum_{d=1}^{D} q(c_d = k) T_{d\omega}}{\lambda_1}
\end{aligned}
\tag{11}
$$

Simplify expression for $\mu_{\omega k}$:

$$\begin{aligned}
&\sum_{\omega=1}^{W} \mu_{\omega k} = \frac{\sum_{\omega=1}^{W}\sum_{d=1}^{D} q(c_d = k) T_{d\omega}}{\lambda_1} \\
&LHS = 1, \quad \sum_{\omega=1}^{W} T_{d\omega} = n_d \quad (given) \\
&\lambda_1 = \sum_{d=1}^{D} q(c_d = k) n_d \\
&\mu_{\omega k} = \frac{\sum_{d=1}^{D} q(c_d = k) T_{d\omega}}{\sum_{d=1}^{D} q(c_d = k) n_d}
\end{aligned} \tag{12}$$

Now, we differentiate with respect to π_k, then $\frac{\partial L'}{\partial \pi_k} = 0$.

$$\begin{aligned}
&\frac{\partial}{\partial \pi_k}[(\sum_{d=1}^{D} q(c_d = k) log(\pi_k n_d! \prod_{\omega=1}^{W} \frac{\mu_{\omega k}^{T_{d\omega}}}{T_{d\omega}!})) - \lambda_2 \pi_k] = 0 \\
&\frac{\partial}{\partial \pi_k}[(\sum_{d=1}^{D} q(c_d = k) log(\pi_k)) + (\sum_{d=1}^{D} q(c_d = k) \sum_{\omega=1}^{W} log(n_d! \frac{\mu_{\omega k}^{T_{d\omega}}}{T_{d\omega}!})) - \lambda_2 \pi_k] = 0 \\
&\frac{\partial}{\partial \pi_k}[\sum_{d=1}^{D} q(c_d = k) log(\pi_k) - \lambda_2 \pi_k] = 0 \\
&\frac{\sum_{d=1}^{D} q(c_d = k)}{\pi_k} - \lambda_2 = 0 \\
&\pi_k = \frac{\sum_{d=1}^{D} q(c_d = k)}{\lambda_2}
\end{aligned} \tag{13}$$

Simplify expression for π_k:

$$\begin{aligned}
&\sum_{k=1}^{K} \pi_k = \frac{\sum_{k=1}^{K}\sum_{d=1}^{D} q(c_d = k)}{\lambda_2} \\
&LHS = 1, \quad \sum_{k=1}^{K} q(c_d = k) = 1 \quad (given) \\
&\lambda_2 = \sum_{d=1}^{D} 1 = D \\
&\pi_k = \frac{\sum_{d=1}^{D} q(c_d = k)}{D}
\end{aligned} \tag{14}$$

Therefore, the update rules are:
For the Expectation step:

$$q(c_d = k) = \frac{\pi_k \prod_{\omega=1}^{W} \mu_{\omega k}^{T_{d\omega}}}{\sum_{j=1}^{K} \pi_j \prod_{\omega=1}^{W} \mu_{\omega j}^{T_{d\omega}}}$$

For the Maximisation step:

$$\mu_{\omega k} = \frac{\sum_{d=1}^{D} q(c_d = k) T_{d\omega}}{\sum_{d=1}^{D} q(c_d = k) n_d}$$

$$\pi_k = \frac{\sum_{d=1}^{D} q(c_d = k)}{D}$$

2 PCA

2.1 Minimum Error Formulation

Task: Given that actual data point

$$x_n = \sum_{i=1}^{P} (x_n^T \mu_i) \mu_i$$

and low dimensional approximation

$$\widetilde{x_n} = \sum_{i=1}^{d} z_{ni} \mu_i + \sum_{i=d+1}^{P} b_i \mu_i$$

and error

$$J = \frac{1}{N} \sum_{n=1}^{N} ||x_n - \widetilde{x_n}||^2$$

prove that the best approximation to minimise the error J is: $z_{ni} = x_n^T \mu_i$ for i=1..d, $b_i = \overline{x_n}^T \mu_i$ for i= $(d+1)..p$
First, reformulate J to simplify the vector norm so that differentiation later will be easier.

$$\begin{aligned} J &= \frac{1}{N} \sum_{n=1}^{N} (x_n - \widetilde{x_n})^T (x_n - \widetilde{x_n}) \\ &= \frac{1}{N} \sum_{n=1}^{N} (x_n^T x_n - 2 x_n^T \widetilde{x_n} + \widetilde{x_n}^T \widetilde{x_n}) \end{aligned} \tag{15}$$

Simplify expressions for $x_n^T x_n$, $x_n^T \widetilde{x_n}$ and $\widetilde{x_n}^T \widetilde{x_n}$ by using orthonormal property of μ_i, i.e.

$$\mu_i^T \mu_j = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \tag{16}$$

$$\begin{aligned} x_n^T x_n &= (\sum_{i=1}^{P} (x_n^T \mu_i) \mu_i)^T (\sum_{i=1}^{P} (x_n^T \mu_i) \mu_i) \\ &= \sum_{i=1}^{P} (x_n^T \mu_i)^2 \end{aligned} \tag{17}$$

$$\begin{aligned} x_n^T \widetilde{x_n} &= (\sum_{i=1}^{P} (x_n^T \mu_i) \mu_i)^T (\sum_{i=1}^{d} z_{ni} \mu_i + \sum_{i=d+1}^{P} b_i \mu_i) \\ &= \sum_{i=1}^{d} (x_n^T \mu_i) z_{ni} + \sum_{i=d+1}^{P} (x_n^T \mu_i) b_i \end{aligned} \tag{18}$$

$$\begin{aligned}\widetilde{x_n}^T\widetilde{x_n} &= (\sum_{i=1}^{d} z_{ni}\mu_i + \sum_{i=d+1}^{P} b_i\mu_i)^T(\sum_{i=1}^{d} z_{ni}\mu_i + \sum_{i=d+1}^{P} b_i\mu_i) \\ &= \sum_{i=1}^{d} z_{ni}^2 + \sum_{i=d+1}^{P} b_i^2\end{aligned} \tag{19}$$

Then, J can be rewritten as:

$$J = \frac{1}{N}\sum_{n=1}^{N}(\sum_{i=1}^{P}(x_n^T\mu_i)^2 - 2\sum_{i=1}^{d}(x_n^T\mu_i)z_{ni} - 2\sum_{i=d+1}^{P}(x_n^T\mu_i)b_i + \sum_{i=1}^{d} z_{ni}^2 + \sum_{i=d+1}^{P} b_i^2) \tag{20}$$

Then, differentiate J with respect to z_{ni} and b_i to get the best approximation to minimise J.
Differentiating with respect to z_{ni}, $\frac{\partial J}{\partial z_{ni}} = 0$:

$$\begin{aligned}&\frac{1}{N}(-2x_n^T\mu_i + 2z_{ni}) = 0 \\ &z_{ni} = x_n^T\mu_i \quad (proven)\end{aligned} \tag{21}$$

Differentiating with respect to b_i, $\frac{\partial J}{\partial b_i} = 0$:

$$\begin{aligned}&\frac{1}{N}(-2x_n^T\mu_i + 2b_i) = 0 \\ &\sum_{n=1}^{N} b_i = \sum_{n=1}^{N} x_n^T\mu_i \\ &Nb_i = \sum_{n=1}^{N} x_n^T\mu_i \\ &b_i = \frac{1}{N}\sum_{n=1}^{N} x_n^T\mu_i = \overline{x_n}^T\mu_i \quad (proven)\end{aligned} \tag{22}$$

3 Deep Generative Models: Class-conditioned VAE

3.1 Derive algorithm for CVAE

Algorithm for CVAE is same as VAE, except with class condition y.
MLE learning:

$$max_\theta log(P_\theta(x|y))$$

But $P_\theta(x|y)$ is conditional marginal likelihood of x, and is hard to compute, since we only have the joint likelihood $P_\theta(x, z|y) = P_\theta(x|z, y)P(z|y)$, but computing $P_\theta(x|y) = \int P_\theta(x, z|y)dz$ is intractable.
So instead of maximising $log(P_\theta(x|y))$ directly, we choose to optimise the variational lower bound of it, and introduce the class conditioned variational posterior $q_\phi(z|x, y)$.

$$\begin{aligned}log(P_\theta(x|y)) &\geq log(P_\theta(x|y)) - KL(q_\phi(z|x, y)||P_\theta(z|x, y)) \\ &= E_{q_\phi(z|x,y)}[log(P_\theta(x, z|y)) - log(q_\phi(z|x, y))] \\ &= L(\theta, \phi)\end{aligned} \tag{23}$$

3.2 Implement CVAE with ZhuSuan on MNIST dataset

I modified the `vae.py` example code in Zhusuan to include the class condition. The CVAE python script `cvae_mnist2.py` is in the attached zip file. The main modifications to the example code is the concatenation of the class variable y with the latent variable z in the generator (decoder) and concatenation of the class variable y with the observed variable x in the encoder as seen in the code fragments below.

```
@zs.meta_bayesian_net(scope="gen", reuse_variables=True)
def build_gen(y, x_dim, z_dim, y_dim, n):
    bn = zs.BayesianNet()
    # z ~ N(0,I)
    z_mean = tf.zeros([n, z_dim])
    z = bn.normal("z", z_mean, std=1., group_ndims=1)

    # Concatenate z and y
    z = tf.concat(axis=1, values=[z,y])

    # x_logits = f_NN(z)
    h = tf.layers.dense(z, 500, activation=tf.nn.relu)
    h = tf.layers.dense(h, 500, activation=tf.nn.relu)
    x_logits = tf.layers.dense(h, x_dim)

    x_mean = bn.deterministic("x_mean", tf.sigmoid(x_logits))

    # x ~ Bernoulli(x_logits)
    bn.bernoulli("x", x_logits, group_ndims=1, dtype=tf.float32)
    return bn


@zs.reuse_variables(scope="q_net")
def build_q_net(x, y, z_dim, y_dim):
    bn = zs.BayesianNet()
    # concatenate x and y
    x = tf.concat(axis=1, values=[x,y])
    h = tf.layers.dense(x, 500, activation=tf.nn.relu)
    h = tf.layers.dense(h, 500, activation=tf.nn.relu)
    z_mean = tf.layers.dense(h, z_dim)
    z_logstd = tf.layers.dense(h, z_dim)
    bn.normal("z", z_mean, logstd=z_logstd, group_ndims=1)
    return bn
```

3.3 Visualise CVAE generations

The CVAE model trained on MNIST produced the following visualisations at the first epoch (Figure 1) and at the 30th epoch (Figure 2).

Figure 1: CVAE generations in first epoch

Figure 2: CVAE generations in 30th epoch