HW3

Sunday, November 21, 2021 3:03 PM

Problem 4 (20pt): [Bayesian Network] Suppose we are given 5 random variables, A_1, A_2, B_1, B_2, B_3. These random variables have dependence relations as follows: B_1 depends on A_1, A_2 depends on B_1. B_2 depends on A_2 and A_1. B_3 depends on A_2. All 5 random variables are binary, i.e., $A_i, B_j \in \{0, 1\}, i = 1, 2; j = 1, 2, 3$.

(1) [5pt] Draw the corresponding bayesian network.

(2) [5pt] Based on the bayesian network in (1), write down the joint distribution $p(A_1, A_2, B_1, B_2, B_3)$.

(3) [5pt] How many independent parameters are needed to fully specify the joint distribution in (2).

(4) [5pt] Suppose we do not have any independence assumption, write down one possible factorization of $p(A_1, A_2, B_1, B_2, B_3)$, and state how many independent parameters are required to describe joint distribution in this case.

1) Draw the bayesian network.

2) Write Joint Distribution $p(A_1, A_2, B_1, B_2, B_3)$.

$$p(A_1, A_2, B_1, B_2, B_3) = p(B_3 | A_1, A_2, B_1, B_2) \cdot p(A_1, A_2, B_1, B_2)$$
$$= p(B_3 | A_1, A_2, B_1, B_2) \cdot p(B_2 | A_1, A_2, B_1) \cdot p(A_1, A_2, B_1)$$
$$= p(B_3 | A_1, A_2, B_1, B_2) \cdot p(B_2 | A_1, A_2, B_1) \cdot p(B_1 | A_1, A_2) \cdot p(A_1, A_2)$$
$$= p(B_3 | A_1, A_2, B_1, B_2) \cdot p(B_2 | A_1, A_2, B_1) \cdot p(B_1 | A_1, A_2) \cdot p(A_2 | A_1) \cdot p(A_1)$$

3) # of independent parameters needed to specify the joint distribution?

$p(B_3 \mid A_1, A_2, B_1, B_2)$	$\rightarrow 2^4 = 16$ parameters	32 total parameters needed
$p(B_2 \mid A_1, A_2, B_1)$	$\rightarrow 2^3 = 8$ parameters	
$p(B_1 \mid A_1, A_2)$	$\rightarrow 2^2 = 4$ parameters	
$p(A_2 \mid A_1)$	$\rightarrow 2^1 = 2$ parameters	
$p(A_1)$	$\rightarrow 2^0 = 1$ parameter	

4) One possible factorization & state # of independent parameters needed.

$$p(A_1, A_2, B_1, B_2, B_3) = p(B_3 | A_2) \cdot p(B_2 | A_1, A_2) \cdot p(A_2 | B_1) \cdot p(B_1 | A_1) \cdot p(A_1)$$

$p(B_3 \mid A_2)$	$\rightarrow 2^1 = 2$ parameters	11 total parameters needed
$p(B_2 \mid A_1, A_2)$	$\rightarrow 2^2 = 4$ parameters	
$p(A_2 \mid B_1)$	$\rightarrow 2^1 = 2$ parameters	
$p(B_1 \mid A_1)$	$\rightarrow 2^1 = 2$ parameters	
$p(A_1)$	$\rightarrow 2^0 = 1$ parameter	

Problem 2 (15pt): [K-means and gradient descent] Recall the loss function for k-means clustering with k clusters, sample points $x_1, x_2, ..., x_n$, and centers $\mu_1, \mu_2, ..., \mu_k$:

$$L = \sum_{j=1}^{k} \sum_{x_i \in S_j} ||x_i - \mu_j||^2$$

, where S_j refers to the set of data points that are closer to μ_j than to any other cluster mean.

(1) [5pt] Instead of updating μ_j by computing he mean, let's minimize L with *batch* gradient descent while holding the sets S_j fixed. Derive the update formula for μ_1 with learning rate ϵ.

(2) [2pt] Derive the update formula for μ_1 with *stochastic* gradient descent on a single sample point x_i. Use learning rate ϵ.

(3) [8pt] In this part, we will connect the batch gradient descent update equation with the standard k-means algorithm. Recall that in the update step of the standard algorithm, we assign each cluster center to be the mean of the data points closest to that center. It turns out that a particular choice of the learning rate ϵ (which may be different for each cluster) makes the two algorithms (batch gradient descent and the standard k-means algorithm) have identical update steps. Let's focus on the update for the first cluster, with center μ_1. Calculate the value of ϵ so that both algorithms perform the same update for μ_1.

1) Minimize L w/ batch gradient descent, holding sets S_i fixed. Derive update formula for μ_1 w/ learning rate ϵ.

$L = \sum_{i=1}^{k} \sum_{x_i \in S_i} ||x_i - \mu_i||^2$

$\rightarrow$ for μ_1: $L = \sum_{x_i \in S_1} ||x_i - \mu_1||^2$

We know L is a quadratic function of μ_1 where $x_i \in S_1$ are constants. For gradient descent, we need to calculate the gradient of L.

$\frac{\partial L}{\partial \mu_1} = 2 \sum_{n=1}^{N} (x_n - \mu_1) \cdot \frac{\partial L}{\partial \mu_1}(x_n - \mu_1) = -2 \sum_{n=1}^{N} (x_n - \mu_1)$ *Chain rule

+ because current is negative * gradient is negative

$\Rightarrow$ Update formula: $\mu_1 = \mu_1 + \epsilon [2 \sum_{n=1}^{N} (x_n - \mu_1)]$

where ϵ is a chosen learning rate (e.g. 0.001) & N is the total number of points assigned to cluster 1.

2) Derive update formula for μ_1 w/ stochastic gradient descent.

Similar to the above, we need to calculate the gradient of L. However, since we are now using SGD, only need to use one sample point x_i per update. Thus, we can remove the summation.

$\frac{\partial L}{\partial \mu_1} = -2(x_i - \mu_1)$ x_i is a single point in S_i where $i = 1$

$\Rightarrow$ Update formula: $\mu_1 = \mu_1 + \epsilon [2(x_i - \mu_1)]$

3) Calculate the value of ϵ so that both algorithms perform the same update for μ_1.

Recall that for K-means clustering, the formula to calculate the mean (aka centroid) for each cluster of points is:

$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$, where r_{nk} is a binary indicator that = 1 if the point x_n is assigned to cluster k & 0 otherwise.
$\Rightarrow \sum_n r_{nk}$ is just the number of points in cluster k, N

Since we are considering 1 cluster $k = 1$:

$\mu_1 = \frac{\sum_{n=1}^{N} x_n}{N}$

We can then just set this equal to the update formula we got in 1) & solve for ϵ.

Problem 3 (10pt): [Latent variable model and GMM] Consider the discrete latent variable model where the latent variable $\mathbf{z}$ use 1-of-K representation. The distribution for latent variable $\mathbf{z}$ is defined as:

$$p(z_k = 1) = \pi_k$$

where $\{\pi_k\}$ satisfy: $0 \leq \pi_k \leq 1$ and $\sum_{k=1}^{K} \pi_k = 1$. Suppose the conditional distribution of observation $\mathbf{x}$ given particular value for $\mathbf{z}$ is Gaussian:

$$p(\mathbf{x} | z_k = 1) = \mathcal{N}(x | \mu_k, \Sigma_k)$$

(1) [2pt] Write down the compact form of $p(\mathbf{z})$ and $p(\mathbf{x}|\mathbf{z})$.

(2) [3pt] Show that the marginal distribution $p(\mathbf{x})$ has the following form:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} | \mu_k, \Sigma_k)$$

(3) [5pt] If we want to find the MLE solution for parameters π_k, μ_k, Σ_k in such model, what algorithm should we use? Briefly describe its major difference compared to K-means algorithm.

1) Write compact form of $p(z)$ & $p(x|z)$.

The prior, $p(z)$, where z is a K-dimensional latent variable having 1-of-K representation (one element $z_k = 1$, rest = 0) and $\sum_k z_k = 1$, can be written as $p(z_k = 1) = \pi_k$.

*$\sum_{k=1}^{K} \pi_k = 1$

We get this from $p(z) = \prod_{k=1}^{K} \pi_k^{z_k}$.

Since every $z_k = 0$ except for one, the above will simplify:
e.g. $K = 3$, $z = (0\ 0\ 1) \Rightarrow z_k = 3$
$\Rightarrow \prod_{k=1}^{3} \pi_k^{z_k} = \pi_1^0 \cdot \pi_2^0 \cdot \pi_3^1 = \pi_3$

$\Rightarrow p(z) = \prod_{k=1}^{K} \pi_k^{z_k}$ *π_k = mixing coefficient

The conditional likelihood, $p(x|z)$, where x is the input data, can be written as $p(x | z_k = 1) = \mathcal{N}(x | \mu_k, \Sigma_k)$, a Gaussian distribution with mean μ_k & covariance Σ_k.

Using the same logic used to define $p(z)$ above since there can only be one $z_k = 1$:

$p(x|z) = \prod_{k=1}^{K} \mathcal{N}(x | \mu_k, \Sigma_k)^{z_k}$

2) Show that marginal distribution has the form: $p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x | \mu_k, \Sigma_k)$.

The marginal distribution of x, $p(x)$, can be obtained by summing the joint distribution over all possible values of z.

We know from the product (aka chain) rule of probabilities:

$p(x_n, z) = p(z) \cdot p(x_n | z)$

Both $p(z)$ & $p(x_n|z)$ were obtained in 1). We can use these with marginalization (sum rule) to get the marginal distribution, $p(x)$, by summing up the terms on z.

$p(x) = \sum_{k=1}^{K} p(z) p(x|n) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x | \mu_k, \Sigma_k)$

3) If MLE solution for π_k, μ_k, Σ_k wanted, which algorithm? Describe major difference compared to K-means algorithm.

$\mu_1 = \frac{\quad}{N}$

We can then just set this equal to the update formula we got in 1) & solve for ϵ.

$\mu_1 + \epsilon[2\sum_{n=1}^{N}(x_n - \mu_1)] = \frac{\sum_{n=1}^{N} x_n}{N}$

$\rightarrow \epsilon[2\sum_{n=1}^{N}(x_n - \mu_1)] = \frac{\sum_{n=1}^{N} x_n}{N} - \mu_1$

$\rightarrow \epsilon = \left[\frac{\sum_{n=1}^{N} x_n}{N} - \mu_1\right] \Big/ 2\sum_{n=1}^{N}(x_n - \mu_1)$

$\rightarrow \epsilon = \frac{1}{N}[\sum_{n=1}^{N} x_n - N\mu_1] \Big/ 2\sum_{n=1}^{N} x_n - 2N\mu$

$\rightarrow \epsilon = \frac{\sum_{n=1}^{N} x_n - N\mu_1}{2N\sum_{n=1}^{N} x_n - N\mu_1} \Rightarrow \boxed{\epsilon = \frac{1}{2N}}$

3) If MLE solution for $\pi_k, \mu_k, \& \Sigma_k$ wanted, which algorithm? Describe major difference compared to K-means algorithm.

Since z is a latent (hidden) variable, Expectation Maximization, EM, is needed to find z & the parameters $\pi_k, \mu_k, \& \Sigma_k$

The EM algorithm consists of 2 steps, the E step & M step. First, z, $\pi_k, \mu_k, \& \Sigma_k$ can be randomly initialized & the initial value of the log likelihood, $\ln p(x|\pi,\mu,\Sigma) = \sum_{n=1}^{N} \ln[\sum_{n=1}^{N} \pi_k N(x_k|\mu_k,\Sigma_k)]$, is computed.

E (Expectation) Step
Algorithm tries to infer value of z using responsibility $\gamma(z_{nk})$, aka the posterior $p(z|x)$, which is gotten from Bayes's rule.

To simplify things, the expectation $E(z_{nk}) = \frac{\pi_k N(x|\mu_k,\Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x|\mu_j,\Sigma_j)}$, the expected value of the indicator variable z_{nk} under posterior $p(z|x,\pi,\mu,\Sigma)$, can be calculated.

M (Maximization) Step
Algorithm updates parameter $\pi_k, \mu_k, \& \Sigma_k$ based on the guess of z.

This is done by maximizing $E_z[\ln p(X,z|\pi,\mu,\Sigma)]$:

$\pi_k = \frac{N_k}{N}$

$\mu_k = \frac{1}{N_k}\sum_{n=1}^{N} \gamma(z_{nk}) x_n$

$\Sigma_k = \frac{1}{N_k}\sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k)(x_n - \mu_k)^T$

The E & M steps repeat until convergence, such as when the parameters stop changing or the log likelihood stops changing.

The difference between K-means & this algorithm is that K-means uses hard assignment, where each data point is only assigned to 1 cluster. Alternatively, this algorithm uses soft assignment, meaning posterior probabilities are used to find how much a data point is associated with a specific cluster. This adds a measure of uncertainty. K-means can be thought of as a special case of this algorithm that uses the same & infinitely small variance for each Gaussian. K-means may not be as accurate but it converges faster.