Assignment #4

Due: 11:59pm EST, 4/13

Homework 4: Latent Variable Models, EM Algorithms, and PCA

Please type your solutions after the corresponding problems using this LaTeX template, and start each problem on a new page.

Please submit the **writeup PDF to the Gradescope assignment 'HW4'**. Remember to assign pages for each question.

Please submit your **LaTeX file and code files to the Gradescope assignment 'HW4 - Supplemental'**.

Problem 1 (Expectation-Maximization for Gamma Mixture Models)

In this problem we will explore expectation-maximization for a Categorical-Gamma Mixture model.

Let us suppose the following generative story for an observation x: first one of K classes is randomly selected, and then the features x are sampled according to this class. If

$$z \sim \text{Categorical}(\boldsymbol{\theta})$$

indicates the selected class, then x is sampled according to the class or "component" distribution corresponding to z. (Here, $\boldsymbol{\theta}$ is the mixing proportion over the K components: $\sum_k \theta_k = 1$ and $\theta_k > 0$. In other words, θ_k refers to the probability that a given observation x is of class k). In this problem, we assume these component distributions are gamma distributions with shared shape parameter but different rate parameters:

$$x|z \sim \text{Gamma}(\alpha, \beta_k).$$

In an unsupervised setting, we are only given a set of observables as our training dataset: $\mathcal{D} = \{x_n\}_{n=1}^N$. The EM algorithm allows us to learn the underlying generative process (the parameters $\boldsymbol{\theta}$ and $\{\beta_k\}$) despite not having the latent variables $\{z_n\}$ corresponding to our training data.

1. **Intractability of the Data Likelihood** We are generally interested in finding a set of parameters β_k that maximizes the likelihood of the observed data:

$$\log p(\{x_n\}_{n=1}^N; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K).$$

If we expand the data likelihood to include the necessary sums over observations x_n and to marginalize out the latents, $\mathbf{z}_n$, we get that

$$\log p(\{x_n\}_{n=1}^N; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) = \sum_{n=1}^N \log\left(\sum_{k=1}^K \text{Gamma}(x_n; \alpha, \beta_k)\theta_k\right).$$

Steps have been left out of the derivation for brevity, but feel free to walk through it as an additional challenge! Referencing the expression above, why is optimizing this likelihood directly intractable?

2. **Complete Data Log Likelihood** Now, assume that we know the latents $\mathbf{z}_n$. In other words, assume that we know which class each x belongs to. We can define the complete dataset $\mathcal{D} = \{(x_n, \mathbf{z}_n)\}_{n=1}^N$ as one that includes these latents $\mathbf{z}_n$. Note that we can now write the complete data log likelihood as:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) &= \log p(\mathcal{D}; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) \\ &= \sum_{n=1}^N \sum_{k=1}^K z_{nk} \log \text{Gamma}(x_n \alpha, \beta_k) + \sum_{n=1}^N \sum_{k=1}^K z_{nk} \log \theta_k.\end{aligned}$$

Again, steps have been left out of the derivation for brevity, but feel free to walk through it as an additional challenge! Why is optimizing this loss now computationally tractable if we know $\mathbf{z}_n$? Please explain how what we have seen in the first two parts motivates the EM algorithm.

(Continued on next page.)

Problem 1 (cont.)

3. **Expectation Step** Now, we will move on to the E step of the EM algorithm. We will start by introducing a mathematical expression for $\mathbf{q}_n$, the posterior over the hidden component variables $\mathbf{z}_n$ conditioned on the observed data x_n with fixed parameters. That is:

$$\mathbf{q}_n = \begin{bmatrix} p(\mathbf{z}_n = \mathbf{C}_1 | x_n; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) \\ \vdots \\ p(\mathbf{z}_n = \mathbf{C}_K | x_n; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) \end{bmatrix}.$$

   Write down and simplify the expression for $\mathbf{q}_n$. Note that because the $\mathbf{q}_n$ represents the posterior over the hidden categorical variables $\mathbf{z}_n$, the components of vector $\mathbf{q}_n$ must sum to 1. The main work is to find an expression for $p(\mathbf{z}_n | x_n; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K)$ for any choice of $\mathbf{z}_n$; i.e., for any 1-hot encoded $\mathbf{z}_n$. With this, you can then construct the different components that make up the vector $\mathbf{q}_n$.

4. **Maximization Step** Using the $\mathbf{q}_n$ estimates from the Expectation Step, derive an update for maximizing the expected complete data log likelihood in terms of $\boldsymbol{\theta}$ and $\{\beta_k\}_{k=1}^K$.

   (a) Derive an expression for the expected complete data log likelihood using $\mathbf{q}_n$. Use the expression given in part 2 as a starting point.

   (b) Find an expression for $\boldsymbol{\theta}$ that maximizes this expected complete data log likelihood. You may find it helpful to use Lagrange multipliers in order to enforce the constraint $\sum \theta_k = 1$. (Hint: Page 7 of the Math Review contains information on Lagrange Multipliers). Why does this optimal $\boldsymbol{\theta}$ make intuitive sense?

   (c) Find an expression for β_k that maximizes the expected complete data log likelihood. Why does this optimal β_k make intuitive sense?

5. Finally, implement your solution (final expressions) in code and attach the final plot below.

   You will receive no points for code not included below.

Solution

1.

2.

3.

4. (a)

   (b)

   (c)

5. Plot:

   Code:

```
def e_step(theta, betas):
    q = 'not implemented'
    return q


def m_step(q):
    theta_hat = 'not implemented'
    beta_hats = 'not implemented'
    return theta_hat, beta_hats


def run_em(theta, betas, iterations=1000):
    theta = 'not implemented'
    betas = 'not implemented'
    return theta, betas
```

Problem 2 (Principal Component Analysis on MNIST Data)

For this problem you will implement PCA from scratch on the first 6000 images of the MNIST dataset. Your job is to apply PCA on MNIST and discuss what kind of structure is found. Implement your solution in `p2.ipynb` and attach the final plots below.

Recall the derivation of PCA covered in Section 6, where the principal components are associated with the largest eigenvalues of the covariance matrix:

(a) Derive the empirical covariance matrix, which is defined as $\Sigma_X = \frac{1}{N}X^T X$ for data matrix X. *This is true only if X is centered: i.e. its columns have mean 0.*

(b) Decompose the data matrix with Singular Value Decomposition (SVD) into $X = USV^T$. One possible way to do this is with `np.linalg.svd`.

(c) In this decomposition, S is a diagonal matrix with entries s_i which are related to eigenvalues λ_i of Σ_X via the equation $\lambda_i = s_i^2/N$ (why is this true?) and V is an orthonormal matrix with columns as the eigenvectors of Σ_X.

(d) For PCA, you want to compute the first $k < d$ principal components. Make sure your eigenvalues are sorted correctly before extracting the corresponding first k eigenvectors!

You will receive no points for using third-party PCA implementations (i.e. `scikit-learn`), but you may use packages for calculating the singular value decomposition of a matrix.

You will receive no points for code not included below.

1. Compute the PCA. Plot the eigenvalues corresponding to the most significant 500 components in order from most significant to least. Make another plot that describes the cumulative proportion of variance explained by the first k most significant components for values of k from 1 through 500. How much variance is explained by the first 500 components? Describe how the cumulative proportion of variance explained changes with k.

   Reminder: Center the data before performing PCA.

2. Plot the mean image of the dataset and plot an image corresponding to each of the first 10 principal components.

   How can we interpret these first 10 principal component images? Do they perform well as a clustering technique? Why or why not?

3. Compute the reconstruction error on the data set using the mean image of the dataset. Then compute the reconstruction error using the first 10 principal components. (For consistency in grading, define the reconstruction error as the squared L2 norm averaged over all data points.)

4. Suppose you took the original matrix of principal components that you found U and multiplied it by some rotation matrix R. Why doesn't this change the reconstruction error? What parts of the interpretation of the principal components change, and what stays the same?

5. Instead of this SVD-based derivation of PCA, we could also train a network to learn PCA transformations. In words, how could an autoencoder be implemented to perform PCA? Briefly describe how you might design the autoencoder layers, activation functions, and loss function. In that design, how would you extract the principal components?

Solution

Plots:

Code:

```
def pca(x, n_comps=500):
    top_eigvals = 'not implemented'
    top_pcomps = 'not implemented'
    return top_eigvals, top_pcomps


def calc_cfvs(eigvals):
    cum_frac_vars = 'not implemented'
    return cum_frac_vars


def calc_errs(x, pcomps):
    err_mean = 'not implemented'
    err_pcomp = 'not implemented'
    return err_mean, err_pcomp
```

1.
2.
3.
4.
5.

Problem 3 (First-order EM Algorithm and Gradient Descent)

In this problem you will investigate the connection between the EM algorithm and gradient descent. As in Problem 1, suppose that the observations $\{x_n\}_{n=1}^N$ are generated according to a mixture model with K classes:

$$
\begin{aligned}
z_n &\sim \mathrm{Cat}(\boldsymbol{\pi}), \\
x_n \mid z_n &\sim p(x_n \mid z_n, \boldsymbol{\theta}),
\end{aligned}
$$

with $\sum_{k=1}^K \pi_k = 1$ and $\boldsymbol{\theta} = (\boldsymbol{\pi}, \mathbf{w}_1, \ldots, \mathbf{w}_K)$ denoting the vector containing all parameters, including the class probabilities and each individual class's parameters (we really only need $\mathbf{w}_{z_n}$ from $\boldsymbol{\theta}$ for each x_n, but this makes the notation easier). Here, $p(\cdot \mid z_n, \boldsymbol{\theta})$ is some arbitrary probability distribution.

1. Let $\boldsymbol{\theta}^t$ denote the current values of the parameters at iteration t. Write out the ELBO function $\mathrm{ELBO}(\boldsymbol{\theta}, q|\boldsymbol{\theta}^t)$, where $q(z_n)$ is the posterior distribution $p(z_n \mid x_n, \boldsymbol{\theta}^t)$. You should expand the expectation over the latent variable z_n into a sum. Be careful to indicate which probabilities should be conditioned on $\boldsymbol{\theta}$ versus $\boldsymbol{\theta}^t$.

Recall that in the M-step of the EM algorithm, we set $\boldsymbol{\theta}^{t+1} := \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ \mathrm{ELBO}(\boldsymbol{\theta}, q|\boldsymbol{\theta}^t)$, where q is once again the posterior distribution. However, in situations where no analytical solution exists, we resort to gradient ascent:

$$
\boldsymbol{\theta}^{t+1} := \boldsymbol{\theta}^t + \eta \nabla_{\boldsymbol{\theta}} \mathrm{ELBO}(\boldsymbol{\theta}, q|\boldsymbol{\theta}^t)\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^t}
$$

for some learning rate $\eta > 0$. In other words, we calculate the gradient of the ELBO at $\boldsymbol{\theta} = \boldsymbol{\theta}^t$ and then take a step in the direction of the gradient to iteratively improve $\boldsymbol{\theta}$. Note that only one gradient is taken per M-step; we do not wait for convergence before calculating the ELBO again. This is known as the *first-order* EM algorithm.

2. Show that $\nabla_{\boldsymbol{\theta}} \mathrm{ELBO}(\boldsymbol{\theta}, q|\boldsymbol{\theta}^t)\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^t} = \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \{x_n\}_{n=1}^N)\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^t}$, where $\ell(\cdot)$ denotes the observed data log-likelihood. That is, show that the gradient of $\mathrm{ELBO}(\boldsymbol{\theta}, q|\boldsymbol{\theta}^t)$ evaluated at $\boldsymbol{\theta} = \boldsymbol{\theta}^t$ is exactly equal to the gradient of the log-likelihood evaluated at $\boldsymbol{\theta} = \boldsymbol{\theta}^t$. This means that the first-order EM algorithm is equivalent to gradient ascent on the observed data log-likelihood, which by now is a very familiar algorithm!

3. For a given value of $\boldsymbol{\theta}^t$, are the gradients $\nabla_{\boldsymbol{\theta}} \mathrm{ELBO}(\boldsymbol{\theta}, q|\boldsymbol{\theta}^t)$ and $\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \{x_n\}_{n=1}^N)$ equal at all values of $\boldsymbol{\theta}$ (not just $\boldsymbol{\theta} = \boldsymbol{\theta}^t$)? Justify your answer using your proof from part 2 by either generalizing your answer for other values of $\boldsymbol{\theta}$ or by explaining how your proof fails for other values of $\boldsymbol{\theta}$.

Now, we will make this problem more concrete by setting $K = 2$ and having the the conditional distributions $p(\cdot \mid z_n, \boldsymbol{\theta})$ be (univariate) Normal. Thus, $z_n \in \{1, 2\}$ and $\boldsymbol{\theta} = (\pi_1, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2)$, with $p(x_n \mid z_n, \boldsymbol{\theta}) = \mathcal{N}(\mu_{z_n}, \sigma_{z_n}^2)$. We only need π_1 because $\pi_2 = 1 - \pi_1$.

4. In the context of this Gaussian mixture model, why do we not need to use the first-order EM algorithm? In other words, why is it "easy" to solve for $\underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ \mathrm{ELBO}(\boldsymbol{\theta}, q|\boldsymbol{\theta}^t)$? In particular, you should explain why components in $\boldsymbol{\theta}$ may be maximized separately and why such maximization is possible. You may find writing out $\mathrm{ELBO}(\boldsymbol{\theta}, q|\boldsymbol{\theta}^t)$ to be helpful, but no math is necessary for this part. **Your response should not be longer than five sentences**.

(Continued on next page.)

Problem 3 (cont.)

5. One well-known clustering algorithm is the k-means algorithm, also known as Lloyd's algorithm. This algorithm proceeds as follows.

   0. Randomly initialize k centers $\mu_1, \dots, \mu_k$.
   1. Assign each observation x_n to its nearest center:
   $$z_n = \operatorname*{argmin}_k \|x_n - \mu_k\|_2^2.$$
   2. Recalculate the k centers:
   $$\mu_j = \frac{1}{N_j} \sum_{n=1}^{N} \mathbf{1}\{z_n = j\} x_n$$
   for $j = 1, \dots, k$, where $N_j = \sum_{n=1}^{N} \mathbf{1}\{z_n = j\}$ (i.e. the number of observations belonging to the jth cluster).
   3. Repeat steps 1 and 2 until convergence.

   Explain how the k-means algorithm is a special case of the EM algorithm. In particular, what must the true values of $\pi_1, \mu_1, \sigma_2^2, \mu_2$, and σ_2^2 be in order for the two algorithms to be equivalent? You may use the update expressions for Gaussian mixture models in the M-step, given in lecture, without proof.

Solution

1.
2.
3.
4.
5.

Problem 4 (Impact Question: Fair Clustering and Subgroup Parity)

Prompt: A national bank in the US provides education loans to students to attend top private schools such as Harvard, MIT, etc. These loans are offered on the basis of a clustering algorithm, where the bank feeds in student attributes such as past education, financial credentials, scholastic achievements, etc. as input. The algorithm clusters these students and offers differing loan amounts and interest rates to each cluster. The bank hopes that the clustering algorithm will successfully group applicants based on potential.

Fairness Constraint: In fairness literature, subgroup parity is defined as equality among subgroups, i.e. no group of individuals should be discriminated against or overtly preferred by an algorithm. You have been roped in as a machine learning specialist to ensure that this clustering algorithm is not violating the bank's ethical goal of preserving subgroup parity. Given that people of color, women and minorities would have likely faced adversity and socioeconomic disparity which would be reflected in their attributes, one unfair clustering outcome could be grouping all underrepresented members in one cluster. Keeping this in mind, answer the following questions. You might find this paper useful, when formulating your answers.

1. Disregarding the fairness constraint for now, choose an appropriate clustering algorithm for this task and explain your choice. There could be multiple correct answers, as long as the reasoning for the choice is sound.

2. Give any two group fairness metrics (with their mathematical formulations) that exist in literature that can be used to ensure that the clustering algorithm respects the fairness constraint?

3. Give any two individual-level fairness metrics (with their mathematical formulations) given in literature.

4. Can you think of a scenario in the education loan dispatchment, where a group-level metric is in direct conflict with an individual-level metric? Concretely, let us consider the following toy dataset, where the features are: *GPA*, *Age*, *Gender*.

   GPA (4.0), Age, Gender:

   (a) 4.0, 22, M

   (b) 4.0, 22, M

   (c) 3.91, 22, M

   (d) 3.88, 22, M

   (e) 3.88, 22, F

   (f) 3.72, 22, F

   (g) 3.75, 22, F

   (h) 3.76, 22, F

   Let's say group-level fairness requires you to enforce a minimum of 50% females in each cluster. This could give a cluster of the following individuals:

   (a) 4.0, 22, M

   (b) 4.0, 22, M

   (c) 3.88, 22, F

   (d) 3.72, 22, F

   However, given that all data points differ only in a single attribute, i.e. GPA, what would individual-level fairness prescribe?

5. Describe a real-life scenario of your choice (in at most 200-300 words), where the ground-truth definition of a fair outcome is fuzzy and needs to be dealt with on a case-by-case basis. Feel free to be creative.

Solution

1.
2.
3.
4.
5.

Name

Collaborators and Resources

Whom did you work with, and did you use any resources beyond cs181-textbook and your notes?

Calibration

Approximately how long did this homework take you to complete (in hours)?