COMPSCI 671D Fall 2019
Homework 4

1 EM Algorithm for Topic Modeling (35 points)

In this question we will try to design an algorithm for discovering the abstract topics that occur in a set of documents. In the problem, we have a set of M abstract documents in the universe, $\mathcal{D}$, which are mixtures of latent topics. We also have a dictionary of words, $\mathcal{W}$, with size N. Intuitively, $\mathcal{W}$ is the list of all unique words that occur at least once within $\mathcal{D}$. Using these two sets we can denote the number of times word w_j occurs in document d_i as $n(w_j, d_i)$. It follows that we can represent a document d_i as a vector of size N, each entry of which corresponds to how many times word w_j appears in it. The topics z are latent variables in a topic set $\mathcal{Z}$ sized K. Intuitively, if a document is about a certain topic, it will include some particular words with higher probability. For example, "music", "show" and "film" appear frequently in documents about Arts while "school", "student" and "teachers" usually occur if the document is about Education. Meanwhile, a document can be a mixture of several topics, e.g., a document can be about arts education for high school students. Inspired by the intuition, a reasonable approach to model the generative process is as follows.

$$\Pr(d_i, z_k, w_j) = \Pr(d_i) \Pr(z_k|d_i) \Pr(w_j|z_k)$$

which means the topics only depend on the documents and the words only depend on topics. For computational convenience, let $\Pr(d_i)$ be a uniform distribution and $\Pr(z_k|d_i)$ and $\Pr(w_j|z_k)$ be Multinomial distributions. i.e.

$$\Pr(d_i) = \frac{1}{M}$$

$$\Pr(z_k|d_i) = \alpha_{ik},\ \sum_{k=1}^{K} \alpha_{ik} = 1$$

$$\Pr(w_j|z_k) = \beta_{kj},\ \sum_{j=1}^{N} \beta_{kj} = 1.$$

We are interested in learning α_{ik} and β_{kj} because they are substantively interesting: α_{ik} represents the proportion of topic k that makes up document i, and β_{kj} the probability of seeing word j under topic k. Therefore, a single word w_j in a document d_i is generated in this way: (1) Choose a topic z_k from the topic distribution $Pr(z_k|d_i)$, the probability of choosing topic k is α_{ik}; (2) Choose a word from the vocabulary distribution $p(w_j|z_k)$ in this topic, the probability is β_{kj}. We want to learn these parameters by maximizing the likelihood of the data using the EM algorithm.

a) For our set of N documents and W word of choices, write down the log-likelihood of the model above, i.e. $\log(p(D = d_i, W = w_j|\alpha, \beta))$.

Remember that in the EM (Expectation-Maximization) algorithm, we can figure out the parameters and the hidden variables by iteratively computing (1) the distribution of latent variables using the old parameters and (2) find new parameters that maximize the likelihood using the old latent variable distribution. So, you can do the following:

b) E-step: Derive the distribution of latent variable $p(z_k|w_j, d_i, \alpha^{old}, \beta^{old})$ by fixing the old parameters.

c) M-step: Find the α^{new} and β^{new} that optimize the log likelihood using $p(z_k|w_j, d_i, \alpha^{old}, \beta^{old})$ as the distribution of z.

You would iterate these until convergence to get the final parameters in practice.

Just so you know, the model you have just been working with in this problem is a very famous and very popular model. :)

2 Neural Networks (65 points)

In this problem you will get the chance to construct a neural network with architecture $784 - H - H - 1$, where H is the number of hidden nodes you choose. This neural network can be used for binary classification, such 0, 1 digits classification for MNIST. The model can be represented as

$$f(\mathbf{x}; \theta) : \mathbb{R}^{784} \to [0, 1]$$

where $\theta = \{\mathbf{W}_1 \in \mathbb{R}^{784\times H}, \mathbf{b}_1 \in \mathbb{R}^H, \mathbf{W}_2 \in \mathbb{R}^{H\times H}, \mathbf{b}_2 \in \mathbb{R}^H, \mathbf{w}_3 \in \mathbb{R}^H, b_3 \in \mathbb{R}, \}$. Explicitly, $f(\mathbf{x})$ is

$$\mathbf{h}_1 = \sigma\left(\mathbf{W}_1^\top \mathbf{x} + \mathbf{b}_1\right)$$

$$\mathbf{h}_2 = \sigma\left(\mathbf{W}_2^\top \mathbf{h}_1 + \mathbf{b}_2\right)$$

$$f(\mathbf{x}) = \sigma\left(\mathbf{w}_3^\top \mathbf{h}_2 + b_3\right)$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$, and it is an element-wise operator, which means if $\mathbf{x} = (x^1, x^2, ..., x^d) \in \mathbb{R}^d$, we have $\sigma(\mathbf{x}) = (\sigma(x^1), \sigma(x^2), ..., \sigma(x^d))$.

The loss function for this model (or the negative log-likelihood) is the usual one:

$$\mathcal{L}(\theta) = -\sum_{i=1}^{N}(y_i \log f(\mathbf{x}_i) + (1 - y_i)\log(1 - f(\mathbf{x}_i)))$$

a) Backpropogation

Derive the gradients $\nabla_{\mathbf{w}_3}\mathcal{L}(\theta)$, $\nabla_{b_3}\mathcal{L}(\theta)$, $\nabla_{\mathbf{W}_2}\mathcal{L}(\theta)$, $\nabla_{\mathbf{b}_2}\mathcal{L}(\theta)$, $\nabla_{\mathbf{W}_1}\mathcal{L}(\theta)$, $\nabla_{\mathbf{b}_1}\mathcal{L}(\theta)$ by backpropagation.

Hints: The problem would be much easier if you define $\mathbf{z}_l = \mathbf{W}_l^T \mathbf{h}_{l-1} + \mathbf{b}_l$, and derive the relationship between $\nabla_{\mathbf{z}_l}\mathcal{L}(\theta)$ and $\nabla_{\mathbf{z}_{l-1}}\mathcal{L}(\theta)$. Also, you can give the final result in a recursive way as long as it is correct and consistent.

b) Weight Initialization

Neural Networks are normally trained by gradient descent based optimization algorithms, for which we have to give initial values for our parameters (weight and bias). The bias parameters are usually initialized as 0's, while the weight parameters are usually initialized by independently sampling from $N(0, \sigma^2)$.

(i) What are the potential problems if we choose σ to be either too small or too large?

(ii) One nice property we would like our weights to have is that by proper initialization, we want the values before activation to be similar in distribution for each layer. It turns out we can achieve this by choosing σ properly. Now suppose we use ReLU activation in our neural network and the inputs are normalized. We want to choose the proper σ_l for the l-th hidden layer such that

$$Var(z_l) = Var(z_{l-1}).$$

We assume the neurons in each layer are mutually independent and share the same distribution and we use z_l to represent the random variable that generates $\mathbf{z}_l$, where $\mathbf{z}_l = \mathbf{W}_l^T \mathbf{h}_{l-1} + \mathbf{b}_l$ and $\mathbf{h}_{l-1} = \max(0, \mathbf{z}_{l-1})$ and $\mathbf{W}_l \in \mathbb{R}^{n_{l-1} \times n_l}$.

Find σ_t such that $Var(z_l) = Var(z_{l-1})$ and prove your result.

c) Implement your Neural Network

Use the preprocessed MNIST data provided in the attachment. MNIST is a dataset of images of handwritten digits that also contain labels for the number they correspond to. We want to train a neural network to learn to recognize these handwritten digits. Use gradient descent with the gradients you have derived in part a) to train a neural network on the data. Report the accuracy for the network on the train and test data. Since the sample size is large, we can approximate the true gradient with stochastic gradient for computational convenience:

$$\nabla \mathcal{L}(\theta) \approx \frac{1}{B} \sum_{j=1}^{B} \nabla \mathcal{L}(\theta)_j$$

which means at each step, instead of calculating the gradients with all samples in the dataset, we randomly pick a mini-batch of B samples and calculate the gradient using these samples. In implementing the neural network, it would be beneficial to build and train the model in a general way. That is, build a NN with d hidden layers such that we can easily modify d.

d) Vanishing Gradient

(i) In our current model, we have 2 hidden layers. What is the magnitude of $\nabla_{\mathbf{W}_1} \mathcal{L}(\theta)$ (you can use Frobenius norm or induced matrix norm)? Now try to modify the the number of hidden layers d, from 1 to 10. What happens to the magnitude of $\nabla_{\mathbf{W}_1} \mathcal{L}(\theta)$ when we increase d? Plot the magnitude against d on log scale. In this question, you don't need to train the whole neural network, just initialize it with random weights, e.g. $\mathcal{N}(0, 1)$, and calculate the gradient.

(ii) Suppose that all weights are finite and the weight matrices $\mathbf{W}_2, \mathbf{W}_3, ..., \mathbf{W}_d$ are diagonalizable matrices in which the absolute value of the eigenvalues are upper-bounded by $4 - \epsilon$. Prove that

$$\lim_{d \to \infty} \nabla_{\mathbf{W}_1} \mathcal{L}(\theta) = \mathbf{0}$$

Hints: Try to use the the relationship between $\nabla_{\mathbf{z}_l} \mathcal{L}(\theta)$ and $\nabla_{\mathbf{z}_{l-1}} \mathcal{L}(\theta)$ and try to apply a common inequality.

(iii) The result above is known as the vanishing gradient problem in the feedforward neural network. Will there still be such a problem if we use a tanh activation? What if we use a ReLU activation?

Give an intuitive explanation of why vanishing gradient happens in light of the shapes of activation functions and provide 2 ways of dealing with this problem.

e) Sigmoid Activation

One potential problem of using the sigmoid activation function for hidden layers is that it maps from $\mathbb{R}$ to $[0, 1]$, which means the output of the sigmoid function is always positive. What happens to the signs of the gradients of hidden layer weights in this situation if we only have one example? What if we are adding gradients up across a batch of data?