Homework 4
Deep learning, Factor Analysis, Mixture Models, EM

CMU 10-701: Machine Learning (Fall 2016)
https://piazza.com/class/is95mzbrvpn63d
OUT: October 25th
DUE: November 14th, 11:59 PM

START HERE: Instructions

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 3.4"). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only.
- **Late Submission Policy:** Late submissions will not receive full credit. Half credit will be awarded to correct solutions submitted within 48 hours of the original deadline. Otherwise, no credit will be given.
- **Submitting your work:** Non-programming parts of the assignment should be submitted as PDFs using Gradescope unless explicitly stated otherwise. Each derivation/proof should be completed on a separate page. Submissions can be handwritten, but should be labeled and clearly legible. If your writing is not legible, you will not be awarded marks. Alternatively, submissions can be written in LaTeX. Upon submission, label each question using the template provided.

Problem 1: Implementing Multi-layer perceptron [40] (Siddharth)

As part of this question, you will implement neural network (with number of hidden layers < 2) for the task of both classification and regression.

For the task of classification, the dataset can be found *here*[1]. (with self-annotated file names). Each input feature vector corresponds to an image of a digit and is a 784-dimensional vector (this dataset has been sampled from the MNIST dataset). The goal is to classify digits as either 2 or 7. For regression, we use the Airfoil Self-Noise dataset and the files can be found *here*[2]. The goal is to predict sound pressure level given 6 dimensional feature vectors.

For both tasks, you will implement two different 1-layer neural networks (NN) such that the activation function in the hidden layer is different (To be clear, 1-layer NN corresponds to 1 hidden layer, in addition to the usual input and output layers). The 2 activation functions are sigmoid and Rectified linear unit (ReLU). Let's call the NN with sigmoid activation as NN-sig-x, and the NN with ReLU activation as NN-relu-x (for x $\in$ { cl, re }, denoting classification and regression respectively). For both input and hidden layers, a bias unit should also be added.

Initialization of weights

The weights for each layer should be initialized by sampling from $\mathcal{U}(-\frac{\sqrt{6}}{\sqrt{V_{l-1}+V_l}}, \frac{\sqrt{6}}{\sqrt{V_{l-1}+V_l}})$, where V_i is the number of units in i^{th} layer. For $i = 0$ (which corresponds to the input layer), $V_i = d$ (dimensionality of the input space). ($\mathcal{U}$ is uniform distribution)

The bias weights in each layer should be initialized to 0.

Normalizing input

As a preprocessing step, normalize the inputs by z-score normalization. For each input dimension, compute the mean μ_d and standard deviation σ_d for all training points. Then transform each $x_{i,d}$ to $\frac{x_{i,d}-\mu_d}{\sigma_d}$. Please use the appropriate means and standard deviations obtained from the training points to do z-normalization for testing and validation points.

1.1 NN for classification [20]

For multi-class classification, a softmax output layer is commonly used. The softmax output layer has k output units (corresponding to k classes). The output from this layer for a particular input sample is basically a $k \times 1$ vector, which corresponds to $[P(label = 1), P(label = 2)...P(label = k)]^T$. The actual output for that particular input corresponds to a vector with all zeros expect value one for the correct class. (For example, if the correct class for a given sample was 2, then the actual output vector would correspond to $[0, 1, 0, 0...0]^T$). The softmax output is computed as follows:

For k inputs (let's call it $[I_1, I_2, ..I_k]^T$) to the output layer, the softmax function transforms this input into: $[\frac{\exp(I_1)}{S}, \frac{\exp(I_2)}{S}, ...\frac{\exp(I_k)}{S}]^T$ where $S = \sum_{m=1}^{k} \exp(I_m)$.

The cross-entropy loss associated with softmax is given as (for the i^{th} training sample, whose label is y_i):

$$L_i = -\log \frac{\exp(I_{y_i})}{\sum_{m=1}^{k} \exp(I_m)} \tag{1}$$

[1] https://www.dropbox.com/s/bx0teegwhdsacvm/hw4_dataset_classf.tar.gz?dl=0
[2] https://www.dropbox.com/s/9becc2vtqkqupls/hw4_dataset_regr.tar.gz?dl=0

So an appropriate loss function for such a scenario, is the mean cross entropy loss which is given as:

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i \tag{2}$$

Here we will also include a $L2$ regularization term, which modifies the above equation to:

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + \frac{\lambda}{2}\sum_{l=0}^{nh}\sum_{d}\sum_{f}(w_{df}{}^{(l)})^2 \tag{3}$$

where the first sum (for l) is for the different layers, and the 2 internal summations are for weights between $layer_l$ and $layer_{l+1}$, nh is the number of hidden layers (basically $nh = 1$).

Consider the loss function as the sum of mean cross-entropy loss and the regularization loss. Let the number of hidden units in the hidden layer l for the be h_l (note $l = 0$ corresponds to input layer). You should implement Batch gradient descent (where the all training samples are considered for gradient update in any iteration), with appropriate modules for forward and backward propagation.

Since, we are employing the cross entropy loss function and dealing with 2 classes, the number of output units = 2. The number of hidden layers is 1. The regularization parameter (λ) should be fixed at 0.4. Let the step size (learning rate) for gradient descent be α. For each of the following parts, fix the number of iterations at 500.

1. Fix $h_1 = 200$. Plot the loss function for the training and validation points as a function of number of iterations for NN-sig-cl with $\alpha = 0.01$.
2. Fix $h_1 = 200$. Plot the loss function for the training and validation points as a function of number of iterations for NN-relu-cl with $\alpha = 0.01$.
3. Use the validation set and the loss function as the appropriate metric for finding the best setting of h_1 and learning rate. Consider only the combinations corresponding to $h_1 \in \{100, 200, 400\}$ and $\alpha \in \{0.01, 0.05\}$ for NN-sig-cl. Consider only the combinations corresponding to $h_1 \in \{100, 200, 400\}$ and $\alpha \in \{0.01, 0.05\}$ for NN-relu-cl.
4. Report the final train, validation and test accuracy values for both NN-sig-cl and NN-relu-cl for the best setting of the hyperparameters.

1.2 NN for regression [20]

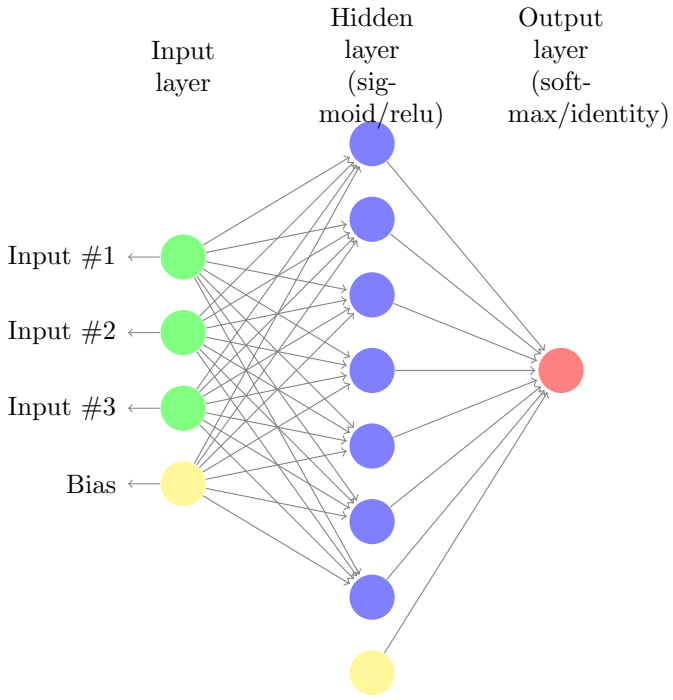
For regression, we will consider the loss function as the sum of mean squared error loss (MSE: $\frac{1}{n}\sum_i (y_i - o_i)^2$ (where y_i is the actual output, and o_i is the predicted output)). and the L2 regularization loss (as above). Let the number of hidden units in the hidden layer l be r_l (note that $l = 0$ corresponds to input layer). You should implement Batch gradient descent (where the all training samples are considered for gradient update in any iteration), with appropriate modules for forward and backward propagation.

Here, the number of output units = 1. The number of hidden layers = 1. The regularization parameter should be fixed at 0.4. Let the step size for gradient descent be α. For each of the following parts, fix the number of iterations at 300.

1. Fix $r_1 = 3$. Plot the loss function for the training and validation points as a function of number of iterations for NN-sig-re with $\alpha = 0.001$.
2. Fix $r_1 = 3$. Plot the loss function for the training and validation points as a function of number of iterations for NN-relu-re with $\alpha = 0.005$.
3. Use the validation set and the loss function as the appropriate metric for finding the best setting of r_1 and learning rate. Consider only the combinations corresponding to $r_1 \in \{3, 4, 5\}$ and $\alpha \in \{0.01, 0.001\}$

for NN-sig-re. Consider only the combinations corresponding to $r_1 \in \{3, 4, 5\}$ and $\alpha \in \{0.001, 0.005\}$ for NN-relu-cl.

4. Report the final train, validation and test MSE values for both NN-sig-re and NN-relu-re for the best setting of the hyperparameters.



Problem 2: Probabilistic Principal Component Analysis (Petar)[20]

Probabilistic Principal Component Analysis (PPCA) is a generative latent variable view on PCA. It is a simpler version of Factor Analysis. In this setting, we assume that a data point was generated by applying linear transformation on a lower-dimensional latent variable and then adding Gaussian noise. Thus, PPCA is a constrained form of a multivariate Gaussian, where there is a limited number of free parameters allowed to capture the main correlations in the dataset. Namely, we can represent generating a data vector $\mathbf{x} \in \mathbb{R}^{d\times 1}$ as:

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon} \tag{4}$$

where $\mathbf{z} \in \mathbb{R}^{m\times 1}, m < d$ is the lower-dimensonal latent vector, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ is d-dimensional noise and:

$$\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}) \tag{5}$$

$$\mathbf{x}|\mathbf{z} \sim \mathcal{N}(\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I}) \tag{6}$$

In this exercise we will derive this models marginal and conditional distributions, as well as the MLE and EM updates.

2.1 Marginal and Conditional distributions [20]

a) Show that $p(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I})$, and that $f(\mathbf{W}) = \mathbf{W}\mathbf{W}^T$ is rotation invariant. What does this tell us about this distribution? A function of matrix $\mathbf{A}$: $f(\mathbf{A})$ is rotation invariant if for some rotation matrix $\mathbf{R}$ and another matrix $\hat{\mathbf{A}} = \mathbf{A}\mathbf{R}$, $f(\mathbf{A}) = f(\mathbf{A}\mathbf{R}) = f(\hat{\mathbf{A}})$. (Hint: Consider the expression of $\mathbf{x}$ in terms of $\mathbf{W}, \mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\epsilon}$ and try to find the mean and covariance matrix directly, without integrating).

b) Show that $p(\mathbf{z}|\mathbf{x}) = \mathcal{N}((\mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I})^{-1}\mathbf{W}^T(\mathbf{x} - \boldsymbol{\mu}), \sigma^2(\mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I})^{-1})$. You may use equation 2.116, 2.117 from the Bishop textbook.

2.2 MLE and EM[10] (Bonus Part)

a) Given a matrix $\mathbf{X}$ with N data vectors $\mathbf{x_i}^T$ in each row, write down the likelihood function $L(\mathbf{X}|\mathbf{W}, \mu, \sigma)$.

b) Find the MLE for μ and write down the maximized likelihood for μ.

c) Let there be a latent lower-dimensional row vector $\mathbf{z_i}^T$ for each corresponding $\mathbf{x_i}^T$. Let $\mathbf{Z}$ be the latent matrix of those rows. Write down the complete log likelihood $L(\mathbf{X}, \mathbf{Z}|\mathbf{W}, \mu, \sigma)$ d) Derive the expectation step for $\mathbf{Z}$ and the maximization update steps for W^{new} and σ^2_{new}

Problem 3: Power Series Mixture Models (Hemank)[25]

EM Algorithm for Mixtures

Suppose the data is given by $Y_i = (X_i, Z_i)$ where X_i is observed, and Z_i is unobserved. When maximizing the data log-likelihood $\log P(X|\varphi)$ w.r.t parameter φ, the EM algorithm augments the observed data to a set of complete data. The EM algorithm maximizes $\log P(X|\varphi)$ by iteratively maximizing $E[\log P(Y|\varphi)]$. At the E-Step of $(k+1)$th iteration, the expected loglikelihood of the complete data model is computed as $Q(\varphi|\varphi^{(k)})$, where $Q(\varphi|\varphi^{(k)}) = E(\log P(Y|\varphi)|X, \varphi^{(k)})$ and expectation is taken with respect to conditional distribution $f(Y|X, \phi^{(k)})$. In M-Step, $Q(\varphi|\varphi^{(k)})$ is maximized over φ. The algorithm can be summarized as follows:
E-Step: Using the current estimates $\varphi^{(k)}$, taken from k-th iteration, calculate $t_{ij} = E(h_j(\theta)|X_i, \varphi^{(k)})$, where $h_j(.)$ are certain functions, and θ are the unobserved quantities. θ_i are the realizations of the unobserved mixing parameter for each data point X_i.
M-Step: Use t_{ij} from the E-Step to maximize the likelihood of the mixing distribution and obtain the updated estimates $\varphi^{(k+1)}$

Power Series Distributions: A discrete distribution is said to belong to the power series family of distributions if its probability function is given by $P(x|\theta) = \alpha_x \theta^x((A(\theta))^{-1})$, with $x = 0, 1 \ldots$, and $\alpha_x > 0$, and $A(\theta)$ is a function of θ not depending on x. Many well-known distributions belong to this family - Poisson, Binomial, Negative Binomial and others.

Now suppose the parameter θ is the mixing random variable, and comes from a distribution $g(\theta)$, then the resultant is a power series mixture and is given by:

$$P_g(x|\varphi) = \int_\theta \frac{\alpha_x \theta^x}{A(\theta)} g(\theta|\varphi) d\theta$$

The posterior expectation $E(\theta^r|x)$ where x conditional on θ follows a power series discrete distribution and θ follows pdf $g(\theta)$ is given by:

$$E(\theta^r|x) = \frac{P_g(x+r|\varphi)\alpha_x}{P_g(x|\varphi)\alpha_{x+r}}$$

where $P_g(x|\varphi)$ is the power series mixture model defined above [1].

3.1 Poisson Exponential Mixture[15]

Consider the following mixture model: $X_i \sim$ Poission(θ_i), $\theta_i \sim$ Exponential(λ)

1. Compute $p(x|\lambda)$.
2. E-Step: Find the t_i update rule.
3. M-Step: Find the updated estimate λ_{new}
4. What distribution does the $p(x|\lambda)$ look-like? Do the EM update rules guarantee convergence towards similar distribution? Is EM necessary for such mixtures?

[1] Sapatinas, 1995: Identifiability of mixtures of power-series distributions and related-characterizations. *Annals of the Institute of Statistical Mathematics*, 47, 447-459

3.2 Case of Negative Binomial[10]

Let X have a Poisson distribution with parameter θ, where $\theta \sim$Gamma(α, β). Show that the unconditional density $P(X)$ is a Negative Binomial distribution.

Problem 4: Expectation Maximization (EM) [25] (Hyun-Ah)

EM algorithm is an algorithm that tries to find the maximum likelihood estimate (MLE) of a parameter θ of a probability distribution in iterative manner.

Let $\mathbf{y} \in \mathbb{R}^d$ be an observed data, $\mathbf{x} \in \mathbb{R}^{d_1}$ a complete data that you don't have, $\theta \in \Omega$ parameters you want to estimate, $\theta^{(m)} \in \Omega$ mth estimate of parameter θ, $p(\mathbf{y}|\theta)$ probability distribution of $\mathbf{y}$ given θ. $\mathcal{X}$ is the support of X, i.e. closure of the set of x where $p(\mathbf{x}|\theta) > 0$, and $\mathcal{X}(\mathbf{y})$ is the support of X conditioned on $\mathbf{y}$, i.e. closure of the set of $\mathbf{x}$ where $p(\mathbf{x}|\mathbf{y}, \theta) > 0$. We assume that complete data follows a probability distribution $p(\mathbf{x}|\theta)$. We do not observe the complete data, but only $\mathbf{y}$, which is the realization of the random vector Y.

From the observation $\mathbf{y}$, we would like to find the MLE: $\hat{\theta}_{MLE} = \arg\max_{\theta \in \Omega} p(\mathbf{y}|\theta)$. In some cases where it is not so trivial to compute $\hat{\theta}_{MLE}$, we can run EM algorithm.

The EM algorithm consists of two steps:

E-step: Given $\theta^{(m)}$ (estimate of parameter θ from previous iteration), compute the conditional expectation $Q(\theta|\theta^{(m)})$, where

$$\begin{aligned} Q(\theta|\theta^{(m)}) &= \int_{\mathcal{X}(\mathbf{y})} \log p(\mathbf{x}|\theta) p(\mathbf{x}|\mathbf{y}, \theta^{(m)}) dx \\ &= E_{X|\mathbf{y},\theta^{(m)}}[\log p(X|\theta)] \end{aligned} \tag{7}$$

M-step: Then the estimate $\theta^{(m+1)}$ is computed as

$$\theta^{(m+1)} = \arg\max_{\theta \in \Omega} Q(\theta|\theta^{(m)}) \tag{8}$$

In short, the EM first makes a guess on the complete data X, and computes MLE of θ assuming that our guess on X is correct. Then we make a guess on the complete data X based on our new estimate θ, and iterate this process.

4.1 Monotonicity of EM [10]

The EM algorithm is not guaranteed to find the global maximum of the likelihood function, but it is guaranteed that every iteration, the likelihood will not become worse. This property is called *monotonicity of the EM*. The theorem on the monotonicity of EM is explained in Theorem 1 below.

Theorem 1 (Monotonicity of the EM)**.** *Let random variables X and Y have parametric densities with parameter $\theta \in \Omega$. Suppose the support of X does not depend on θ, and the Markov relationship $\theta \to X \to Y$, that is,*

$$p(\mathbf{y}|\mathbf{x}, \theta) = p(\mathbf{y}|\mathbf{x}) \tag{9}$$

holds for all $\theta \in \Omega$, $\mathbf{x} \in \mathcal{X}$, and $\mathbf{y} \in \mathcal{Y}$. Then for $\theta \in \Omega$ and any $\mathbf{y} \in \mathcal{Y}$ with $\mathcal{X}(\mathbf{y}) \neq \emptyset$,

$$l(\theta) \geq l(\theta^{(m)}) \tag{10}$$

if $Q(\theta|\theta^{(m)}) \geq Q(\theta^{(m)}|\theta^{(m)})$. (here, $l(\theta) = \log p(y|\theta)$)

1. **[10 pt]** Prove the monotonicity of the EM (Theorem 1).
   (hint: use Jensen's inequality: "if X is a random variable and ϕ is a convex function, then $\phi(E[X]) \leq E[\phi(X)]$"[3])

[3] https://en.wikipedia.org/wiki/Jensen%27s_inequality

4.2 EM for i.i.d. samples [15]

Most of the cases that you will be dealing with assume that the complete data X is a set of n independently and identically distributed (i.i.d.) random vectors ($X = [X_1, X_2, \cdots X_n]^T$), and ith observed sample $\mathbf{y}_i$ is a function of $\mathbf{x}_i$ only. Then using below proposition, you can simply decompose the Q function into a sum, which makes the computation easier.

Proposition 1 (EM for i.i.d. samples)**.** *Suppose* $p(\mathbf{x}|\theta) = \prod_{i=1}^{n} p(\mathbf{x}_i|\theta)$, $\forall \mathbf{x} \in \mathcal{X}^n$ *and* $\forall \theta \in \Omega$, *and the Markov relationship* $\theta \to X_i \to Y_i$ *holds for* $\forall i = 1, \cdots, n$, *i.e.*

$$p(\mathbf{y}_i|\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_{i-1}, \mathbf{y}_{i+1}, \cdots, \mathbf{y}_n, \theta) = p(\mathbf{y}_i|\mathbf{x}_i) \tag{11}$$

then

$$Q(\theta|\theta^{(m)}) = \sum_{i=1}^{n} Q_i(\theta|\theta^{(m)}) \tag{12}$$

where

$$Q_i(\theta|\theta^{(m)}) = E_{X_i|\mathbf{y}_i,\theta^{(m)}}\Big[\log p(X_i|\theta)\Big], i = 1, \cdots, n \tag{13}$$

We will prove above Proposition 1.

1. **[5 pt]** First, show that $p(\mathbf{x}, \mathbf{y}|\theta) = \prod_{i=1}^{n} p(\mathbf{x}_i, \mathbf{y}_i|\theta)$, which means that the elements of the set $\{(X_i, Y_i)\}$ are mutually independent given θ.

2. **[5 pt]** Now show that $p(\mathbf{x}_i|\mathbf{y}, \theta) = p(\mathbf{x}_i|\mathbf{y}_i, \theta)$

3. **[5 pt]** Now show that $Q(\theta|\theta^{(m)}) = \sum_{i=1}^{n} E_{X_i|\mathbf{y}_i,\theta^{(m)}}\Big[\log p(X_i|\theta)\Big] = \sum_{i=1}^{n} Q_i(\theta|\theta^{(m)})$