CS 229 Autumn 2017
Problem Set #3: Deep Learning & Unsupervised learning

Due Wednesday, Nov 15 at 11:59 pm on Gradescope.

Notes: (1) These questions require thought, but do not require long answers. Please be concise where possible. (2) If you have a question about this homework, we encourage you to post your question on our Piazza forum, at `https://piazza.com/stanford/fall2017/cs229`. (3) If you missed the first lecture or are unfamiliar with the collaboration or honor code policy, please read the policy on Handout #1 (available from the course website) before starting work. (4) For problems that require programming, please include in your submission a printout of your code (with comments) and any figures that you are asked to plot.

Remember to tag all question parts in Gradescope to avoid docked points. If you are skipping a question, please include it on your PDF/photo, but leave the question blank and tag it appropriately on Gradescope. This includes extra credit problems. If you are scanning your document by cellphone, please see `https://gradescope.com/help#help-center-item-student-scanning` for suggested practices.

1. **[20 points] A Simple Neural Network**

   Let $X = \{x^{(1)}, \cdots, x^{(m)}\}$ be a dataset of m samples with 2 features, i.e $x^{(i)} \in \mathbb{R}^2$. The samples are classified into 2 categories with labels $y^{(i)} \in \{0, 1\}$. A scatter plot of the dataset is shown in Figure 1:

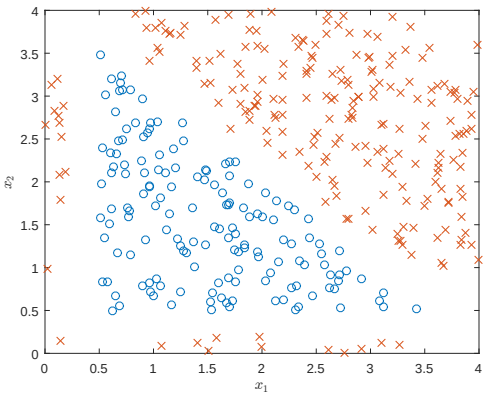

Figure 1: Plot of dataset X.

The examples in class 1 are marked as as "×" and examples in class 0 are marked as "○". We want to perform binary classification using a simple neural network with the architecture shown in Figure 1:

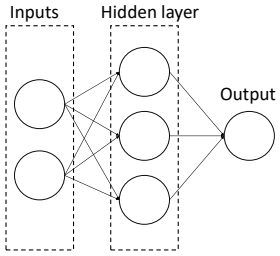

Figure 2: Architecture for our simple neural network.

Denote the two features x_1 and x_2, the three neurons in the hidden layer h_1, h_2, and h_3, and the output neuron as o. Let the weight from x_i to h_j be $w_{i,j}^{[1]}$ for $i \in \{1, 2\}, j \in \{1, 2, 3\}$, and the weight from h_j to o be $w_j^{[2]}$. Finally, denote the intercept weight for h_j as $w_{0,j}^{[1]}$, and the intercept weight for o as $w_0^{[2]}$. For the loss function, we'll use average squared loss instead of the usual negative log-likelihood:

$$l = \frac{1}{m} \sum_{i=1}^{m} (o^{(i)} - y^{(i)})^2,$$

where $o^{(i)}$ is the result of the output neuron for example i.

(a) [5 points] Suppose we use the sigmoid function as the activation function for h_1, h_2, h_3 and o. What is the gradient descent update to $w_{1,2}^{[1]}$, assuming we use a learning rate of α? Your answer should be written in terms of $x^{(i)}$, $o^{(i)}$, $y^{(i)}$, and the weights.

(b) [10 points] Now, suppose instead of using the sigmoid function for the activation function for h_1, h_2, h_3 and o, we instead used the step function $f(x)$, defined as

$$f(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$$

What is one set of weights that would allow the neural network to classify this dataset with 100% accuracy? Please specify a value for the weights in the order given below and explain your reasoning.

$$w_{0,1}^{[1]} =?, w_{1,1}^{[1]} =?, w_{2,1}^{[1]} =?$$
$$w_{0,2}^{[1]} =?, w_{1,2}^{[1]} =?, w_{2,2}^{[1]} =?$$
$$w_{0,3}^{[1]} =?, w_{1,3}^{[1]} =?, w_{2,3}^{[1]} =?$$
$$w_0^{[2]} =?, w_1^{[2]} =?, w_2^{[2]} =?, w_3^{[2]} =?$$

Hint: There are three sides to a triangle, and there are three neurons in the hidden layer.

(c) [5 points] Let the activation functions for h_1, h_2, h_3 be the linear function $f(x) = x$ and the activation function for o be the same step function as before. Is there a specific set of weights that will make the loss 0? If yes, please explicitly state a value for every weight. If not, please explain your reasoning.

2. **[15 points] EM for MAP estimation**

   The EM algorithm that we talked about in class was for solving a maximum likelihood estimation problem in which we wished to maximize

   $$\prod_{i=1}^{m} p(x^{(i)}; \theta) = \prod_{i=1}^{m} \sum_{z^{(i)}} p(x^{(i)}, z^{(i)}; \theta),$$

   where the $z^{(i)}$'s were latent random variables. Suppose we are working in a Bayesian framework, and wanted to find the MAP estimate of the parameters θ by maximizing

   $$\left(\prod_{i=1}^{m} p(x^{(i)}|\theta) \right) p(\theta) = \left(\prod_{i=1}^{m} \sum_{z^{(i)}} p(x^{(i)}, z^{(i)}|\theta) \right) p(\theta).$$

   Here, $p(\theta)$ is our prior on the parameters. Generalize the EM algorithm to work for MAP estimation. You may assume that $\log p(x, z|\theta)$ and $\log p(\theta)$ are both concave in θ, so that the M-step is tractable if it requires only maximizing a linear combination of these quantities. (This roughly corresponds to assuming that MAP estimation is tractable when x, z is fully observed, just like in the frequentist case where we considered examples in which maximum likelihood estimation was easy if x, z was fully observed.)

   Make sure your M-step is tractable, and also prove that $\prod_{i=1}^{m} p(x^{(i)}|\theta)p(\theta)$ (viewed as a function of θ) monotonically increases with each iteration of your algorithm.

3. **[25 points] EM application**

Consider the following problem. There are P papers submitted to a machine learning conference. Each of R reviewers reads each paper, and gives it a score indicating how good he/she thought that paper was. We let $x^{(pr)}$ denote the score that reviewer r gave to paper p. A high score means the reviewer liked the paper, and represents a recommendation from that reviewer that it be accepted for the conference. A low score means the reviewer did not like the paper.

We imagine that each paper has some "intrinsic," true value that we denote by μ_p, where a large value means it's a good paper. Each reviewer is trying to estimate, based on reading the paper, what μ_p is; the score reported $x^{(pr)}$ is then reviewer r's guess of μ_p.

However, some reviewers are just generally inclined to think all papers are good and tend to give all papers high scores; other reviewers may be particularly nasty and tend to give low scores to everything. (Similarly, different reviewers may have different amounts of variance in the way they review papers, making some reviewers more consistent/reliable than others.) We let ν_r denote the "bias" of reviewer r. A reviewer with bias ν_r is one whose scores generally tend to be ν_r higher than they should be.

All sorts of different random factors influence the reviewing process, and hence we will use a model that incorporates several sources of noise. Specifically, we assume that reviewers' scores are generated by a random process given as follows:

$$\begin{aligned} y^{(pr)} &\sim \mathcal{N}(\mu_p, \sigma_p^2), \\ z^{(pr)} &\sim \mathcal{N}(\nu_r, \tau_r^2), \\ x^{(pr)}|y^{(pr)}, z^{(pr)} &\sim \mathcal{N}(y^{(pr)} + z^{(pr)}, \sigma^2). \end{aligned}$$

The variables $y^{(pr)}$ and $z^{(pr)}$ are independent; the variables (x, y, z) for different paper-reviewer pairs are also jointly independent. Also, we only ever observe the $x^{(pr)}$'s; thus, the $y^{(pr)}$'s and $z^{(pr)}$'s are all latent random variables.

We would like to estimate the parameters $\mu_p, \sigma_p^2, \nu_r, \tau_r^2$. If we obtain good estimates of the papers' "intrinsic values" μ_p, these can then be used to make acceptance/rejection decisions for the conference.

We will estimate the parameters by maximizing the marginal likelihood of the data $\{x^{(pr)}; p = 1, \ldots, P, r = 1, \ldots, R\}$. This problem has latent variables $y^{(pr)}$ and $z^{(pr)}$, and the maximum likelihood problem cannot be solved in closed form. So, we will use EM. Your task is to derive the EM update equations. Your final E and M step updates should consist only of addition/subtraction/multiplication/division/log/exp/sqrt of scalars; and addition/subtraction/multiplication/inverse/determinant of matrices. For simplicity, you need to treat only $\{\mu_p, \sigma_p^2; p = 1 \ldots P\}$ and $\{\nu_r, \tau_r^2; r = 1 \ldots R\}$ as parameters. I.e. treat σ^2 (the conditional variance of $x^{(pr)}$ given $y^{(pr)}$ and $z^{(pr)}$) as a fixed, known constant.

(a) In this part, we will derive the E-step:

(i) The joint distribution $p(y^{(pr)}, z^{(pr)}, x^{(pr)})$ has the form of a multivariate Gaussian density. Find its associated mean vector and covariance matrix in terms of the parameters $\mu_p, \sigma_p^2, \nu_r, \tau_r^2$, and σ^2.

[Hint: Recognize that $x^{(pr)}$ can be written as $x^{(pr)} = y^{(pr)} + z^{(pr)} + \epsilon^{(pr)}$, where $\epsilon^{(pr)} \sim \mathcal{N}(0, \sigma^2)$ is independent Gaussian noise.]

(ii) Derive an expression for $Q_{pr}(y^{(pr)}, z^{(pr)}) = p(y^{(pr)}, z^{(pr)}|x^{(pr)})$ (E-step), using the rules for conditioning on subsets of jointly Gaussian random variables (see the notes

on Factor Analysis).

(b) Derive the M-step updates to the parameters $\{\mu_p, \nu_r, \sigma_p^2, \tau_r^2\}$. [Hint: It may help to express the lower bound on the likelihood in terms of an expectation with respect to $(y^{(pr)}, z^{(pr)})$ drawn from a distribution with density $Q_{pr}(y^{(pr)}, z^{(pr)})$.]

Remark. In a recent machine learning conference, John Platt (whose SMO algorithm you've seen) implemented a method quite similar to this one to estimate the papers' true scores μ_p. (There, the problem was a bit more complicated because not all reviewers reviewed every paper, but the essential ideas are the same.) Because the model tried to estimate and correct for reviewers' biases ν_r, its estimates of μ_p were significantly more useful for making accept/reject decisions than the reviewers' raw scores for a paper.

4. **[15 points] KL divergence and Maximum Likelihood**

The Kullback-Leibler (KL) divergence between two discrete-valued distributions $P(X), Q(X)$ is defined as follows:[1]

$$KL(P\|Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

For notational convenience, we assume $P(x) > 0, \forall x$. (Otherwise, one standard thing to do is to adopt the convention that "$0 \log 0 = 0$.") Sometimes, we also write the KL divergence as $KL(P||Q) = KL(P(X)||Q(X))$.

The KL divergence is an assymmetric measure of the distance between 2 probability distributions. In this problem we will prove some basic properties of KL divergence, and work out a relationship between minimizing KL divergence and the maximum likelihood estimation that we're familiar with.

(a) [5 points] Nonnegativity. Prove the following:

$$\forall P, Q \quad KL(P\|Q) \geq 0$$

and

$$KL(P\|Q) = 0 \quad \text{if and only if } P = Q.$$

[Hint: You may use the following result, called **Jensen's inequality**. If f is a convex function, and X is a random variable, then $E[f(X)] \geq f(E[X])$. Moreover, if f is strictly convex (f is convex if its Hessian satisfies $H \geq 0$; it is *strictly* convex if $H > 0$; for instance $f(x) = -\log x$ is strictly convex), then $E[f(X)] = f(E[X])$ implies that $X = E[X]$ with probability 1; i.e., X is actually a constant.]

(b) [5 points] **Chain rule for KL divergence.** The KL divergence between 2 conditional distributions $P(X|Y), Q(X|Y)$ is defined as follows:

$$KL(P(X|Y)\|Q(X|Y)) = \sum_y P(y) \left(\sum_x P(x|y) \log \frac{P(x|y)}{Q(x|y)} \right)$$

This can be thought of as the expected KL divergence between the corresponding conditional distributions on x (that is, between $P(X|Y = y)$ and $Q(X|Y = y)$), where the expectation is taken over the random y.

Prove the following chain rule for KL divergence:

$$KL(P(X,Y)\|Q(X,Y)) = KL(P(X)\|Q(X)) + KL(P(Y|X)\|Q(Y|X)).$$

(c) [5 points] **KL and maximum likelihood.**

Consider a density estimation problem, and suppose we are given a training set $\{x^{(i)}; i = 1, \ldots, m\}$. Let the empirical distribution be $\hat{P}(x) = \frac{1}{m} \sum_{i=1}^{m} 1\{x^{(i)} = x\}$.

[1] If P and Q are densities for continuous-valued random variables, then the sum is replaced by an integral, and everything stated in this problem works fine as well. But for the sake of simplicity, in this problem we'll just work with this form of KL divergence for probability mass functions/discrete-valued distributions.

($\hat{P}$ is just the uniform distribution over the training set; i.e., sampling from the empirical distribution is the same as picking a random example from the training set.)

Suppose we have some family of distributions P_θ parameterized by θ. (If you like, think of $P_\theta(x)$ as an alternative notation for $P(x;\theta)$.) Prove that finding the maximum likelihood estimate for the parameter θ is equivalent to finding P_θ with minimal KL divergence from $\hat{P}$. I.e. prove:

$$\arg\min_\theta KL(\hat{P}\|P_\theta) = \arg\max_\theta \sum_{i=1}^{m} \log P_\theta(x^{(i)})$$

Remark. Consider the relationship between parts (b-c) and multi-variate Bernoulli Naive Bayes parameter estimation. In the Naive Bayes model we assumed P_θ is of the following form: $P_\theta(x,y) = p(y)\prod_{i=1}^{n} p(x_i|y)$. By the chain rule for KL divergence, we therefore have:

$$KL(\hat{P}\|P_\theta) = KL(\hat{P}(y)\|p(y)) + \sum_{i=1}^{n} KL(\hat{P}(x_i|y)\|p(x_i|y)).$$

This shows that finding the maximum likelihood/minimum KL-divergence estimate of the parameters decomposes into $2n+1$ independent optimization problems: One for the class priors $p(y)$, and one for each of the conditional distributions $p(x_i|y)$ for each feature x_i given each of the two possible labels for y. Specifically, finding the maximum likelihood estimates for each of these problems individually results in also maximizing the likelihood of the joint distribution. (If you know what Bayesian networks are, a similar remark applies to parameter estimation for them.)

5. **[20 points] K-means for compression**

   In this problem, we will apply the K-means algorithm to lossy image compression, by reducing the number of colors used in an image.

   We will be using the following files:

   - `http://cs229.stanford.edu/ps/ps3/mandrill-small.tiff`
   - `http://cs229.stanford.edu/ps/ps3/mandrill-large.tiff`

   The `mandrill-large.tiff` file contains a 512x512 image of a mandrill represented in 24-bit color. This means that, for each of the 262144 pixels in the image, there are three 8-bit numbers (each ranging from 0 to 255) that represent the red, green, and blue intensity values for that pixel. The straightforward representation of this image therefore takes about $262144 \times 3 = 786432$ bytes (a byte being 8 bits). To compress the image, we will use K-means to reduce the image to $k = 16$ colors. More specifically, each pixel in the image is considered a point in the three-dimensional (r, g, b)-space. To compress the image, we will cluster these points in color-space into 16 clusters, and replace each pixel with the closest cluster centroid.

   Follow the instructions below. Be warned that some of these operations can take a while (several minutes even on a fast computer)![2]

   (a) MATLAB/Octave: Start up MATLAB/Octave, and type `A = double(imread('mandrill-large.tiff'));` to read in the image. Now, `A` is a "three dimensional matrix," and `A(:,:,1)`, `A(:,:,2)` and `A(:,:,3)` are 512x512 arrays that respectively contain the red, green, and blue values for each pixel. Enter `imshow(uint8(round(A)));` to display the image.

   Python: Start up Python, and type

   ```
   from matplotlib.image import imread; import matplotlib.pyplot as plt;
   ```

   and run `A = imread('mandrill-large.tiff')` . Now, `A` is a "three dimensional matrix," and `A[:,:,0]`, `A[:,:,1]` and `A[:,:,2]` are 512x512 arrays that respectively contain the red, green, and blue values for each pixel. Enter `plt.imshow(A); plt.show()` to display the image.

   (b) Since the large image has 262144 pixels and would take a while to cluster, we will instead run vector quantization on a smaller image. Repeat (a) with `mandrill-small.tiff`. Treating each pixel's (r, g, b) values as an element of $\mathbb{R}^3$, run K-means[3] with 16 clusters on the pixel data from this smaller image, iterating (preferably) to convergence, but in no case for less than 30 iterations. For initialization, set each cluster centroid to the (r, g, b)-values of a randomly chosen pixel in the image.

   (c) Take the matrix `A` from `mandrill-large.tiff`, and replace each pixel's (r, g, b) values with the value of the closest cluster centroid. Display the new image, and compare it visually to the original image. Hand in all your code and a copy of your compressed image.

   (d) If we represent the image with these reduced (16) colors, by (approximately) what factor have we compressed the image?

[2] In order to use the `imread` and `imshow` commands in octave, you have to install the Image package from octave-forge. This package and installation instructions are available at: `http://octave.sourceforge.net`

[3] Please implement K-means yourself, rather than using built-in functions.