CS229 Problem Set 3

anjay.h.friedman

August 2020

1 Introduction

This document represents my solutions to the 2019 CS229 course's third problem set. I appreciate that the materials for the course are available from Stanford's and other websites and hope that I am not infringing on any rights. It took me **11 hours** to complete.

2 Problems

2.1 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:

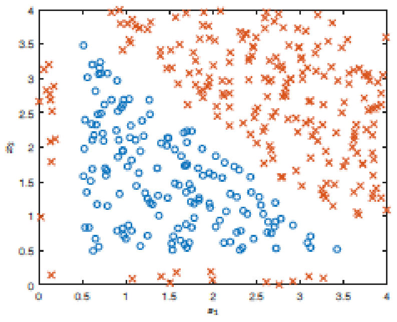


Figure 1: Plot of dataset X.

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

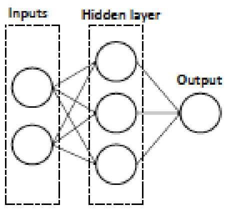


Figure 2: Architecture for our simple neural network.

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 $w^{[1]}_{i,j}$ for $i \in \{1, 2\}, j \in \{1, 2, 3\}$, and the weight from h_j to o be $w^{[2]}_j$. Finally, denote the intercept weight for h_j as $w^{[1]}_{0,j}$, and the intercept weight for o as $w^{[2]}_0$. 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 $w^{[1]}_{1,2}$, 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.

Solution
The gradient descent update rule to a weight $w_{1,2}^{[1]}$ is $w_{1,2}^{[1]} := w_{1,2}^{[1]} - \alpha \frac{\partial J}{\partial w_{1,2}^{[1]}}$ where α is the learning rate
Since $J = \frac{1}{m} \sum_{i=1}^{m} (o^{(i)} - y^{(i)})^2$, $\quad w_{1,2}^{[1]} := w_{1,2}^{[1]} - \frac{\alpha}{m} \sum_{i=1}^{m} \frac{\partial J^{(i)}}{\partial w_{1,2}^{[1]}}$

$$:= w_{1,2}^{[1]} - \frac{\alpha}{m} \sum_{i=1}^{m} \frac{\partial J^{(i)}}{\partial o^{(i)}} \frac{\partial o^{(i)}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial a_2} \frac{\partial a_2}{\partial z_2^{[1]}} \frac{\partial z_2^{[1]}}{\partial w_{1,2}^{[1]}} \quad (1)$$

$$:= w_{1,2}^{[1]} - \frac{\alpha}{m} \sum_{i=1}^{m} (2(o^{(i)} - y^{(i)}))(o^{(i)}(1 - o^{(i)}))(w_2^{[2]})((g(z_2^{[1]})(1 - g(z_2^{[1]})))(x_1^{(i)}) \quad (2)$$

$$:= w_{1,2}^{[1]} - \frac{\alpha}{m} \sum_{i=1}^{m} 2(o^{(i)} - y^{(i)})o^{(i)}(1 - o^{(i)})w_2^{[2]} h_2^{(i)}(1 - h_2^{(i)})x_1^{(i)} \quad (3)$$

$$\longrightarrow w_{1,2}^{[1]} := w_{1,2}^{[1]} - \frac{2\alpha w_2^{[2]}}{m} \sum_{i=1}^{m} (o^{(i)} - y^{(i)})(o^{(i)})(1 - o^{(i)})(h_2^{(i)})(1 - h_2^{(i)})x_1^{(i)} \quad \diamond$$

(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.

Solution
The basic approach is to set the first layer of weights such that $h_1 > 0$ if the data point is above the bottom edge of the triangle, $h_2 > 0$ if the data point is to the right of the left edge of the triangle and $h_3 > 0$ if the data point is below the diagonal edge of the triangle. We will then set the second layer of weights

such that only if h_1, h_2, h_3 are greater than 0 is the label 0 (since this indicates that the point is within the triangle and should be labelled with 0)

$$w^{[1]}_{0,1} = -0.4 \quad w^{[1]}_{1,1} = 0 \quad w^{[1]}_{2,1} = 1$$

$$w^{[1]}_{0,2} = -.4 \quad w^{[1]}_{1,2} = 1 \quad w^{[1]}_{2,2} = 0$$

$$w^{[1]}_{0,3} = 4.2 \quad w^{[1]}_{1,3} = -1 \quad w^{[1]}_{2,3} = -1$$

$$w^{[2]}_{0} = 2.5 \quad w^{[2]}_{1} = -1 \quad w^{[2]}_{2} = -1 \quad w^{[2]}_{3} = -1$$

It's easy to check that the final label is only 0 and inside the triangle if all of h_i are greater than 0 or inside the lines of the triangle.

(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.

Solution
No. Clearly the data is not linearly separable. Thus there is no set of weights that can correctly classify all the data since linear activation functions preserve linearity or $h(f(x)) = g(x)$ is linear if f and h are both linear.

2.2 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.

Solution

MAP Problem $\rightarrow \max_\theta \, (\prod_{i=1}^{m}\sum_{z^{(i)}} p(x^{(i)}, z^{(i)}|\theta))p(\theta)$

$$= \max_\theta \, log(\prod_{i=1}^{m}\sum_{z^{(i)}} p(x^{(i)}, z^{(i)}|\theta)) + log(p(\theta)) \quad (1)$$

$$= \max_\theta \, \sum_{i=1}^{m} log(\sum_{z^{(i)}} p(x^{(i)}, z^{(i)}|\theta)) + log(p(\theta)) \quad (2)$$

$$= \max_\theta \, \sum_{i=1}^{m} log(\sum_{z^{(i)}} Q(z^{(i)})\frac{p(x^{(i)}, z^{(i)}|\theta)}{Q(z^{(i)})}) + log(p(\theta)) \quad (3)$$

$$\geq \max_\theta \, \sum_{i=1}^{m}\sum_{z^{(i)}} Q(z^{(i)}) log(\frac{p(x^{(i)}, z^{(i)}|\theta)}{Q(z^{(i)})}) + log(p(\theta)) \quad (4)$$

(1) and (2) come from simplifying (3) from multiplying and dividing by Q and (4) from Jensen's Inequality since the log function is concave

For (4) to hold with equality, $\frac{p(x^{(i)}, z^{(i)}|\theta)}{Q(z^{(i)})}$ must be a "constant valued" RV or $Q(z^{(i)}) \, \alpha \, p(x^{(i)}, z^{(i)}|\theta)$

This is satisfied when $Q(z^{(i)}) = P(z^{(i)}|x^{(i)}, \theta)$

Proof of monotonically increasing θ

$$l(\theta^{(i+1)}) \sum_{i=1}^{m} \sum_{z^{(i)}} Q(z^{(i)}) log(\frac{p(x^{(i)}, z^{(i)}|\theta^{(i+1)})}{Q(z^{(i)})}) + log(p(\theta^{(i+1)})) \quad (1)$$

$$\geq \sum_{i=1}^{m} \sum_{z^{(i)}} Q(z^{(i)}) log(\frac{p(x^{(i)}, z^{(i)}|\theta^{(i)})}{Q(z^{(i)})}) + log(p(\theta^{(i)})) \quad (2)$$

$$= l(\theta^{(i)}) \diamond \quad (3)$$

(1) comes from Jensen's Inequality (2) from how we choose $\theta^{(i+1)}$ and (3) from Jensen's Inequality again

2.3 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

Solution

I will just put the results for this problem since the calculations are a bit dry and long.

$$\begin{bmatrix} y \\ z \\ x \end{bmatrix} \sim \mathcal{N}(\mu_{yzx}, \Sigma_{yzx})$$

$$\mu_{yzx} = \begin{bmatrix} \mu_p \\ v_r \\ u_p + v_r \end{bmatrix} \quad \Sigma_{yzx} = \begin{bmatrix} \sigma_p^2 & 0 & \sigma_p^2 \\ 0 & \tau_r^2 & \tau_r^2 \\ \sigma_p^2 & \tau_r^2 & \sigma_p^2 + \tau_r^2 + \sigma^2 \end{bmatrix}$$

$$p(y, z|x) \sim \mathcal{N}(\mu_{y,z|x}, \Sigma_{y,z|x})$$

$$\mu_{y,z|x} = \begin{bmatrix} \mu_p \\ v_r \end{bmatrix} + \frac{x - \mu_p - v_r}{\sigma_p^2 + \tau_r^2 + \sigma^2} \begin{bmatrix} \sigma_p^2 \\ \tau_r^2 \end{bmatrix} \quad \Sigma_{yzx} = \begin{bmatrix} \sigma_p^2 & 0 \\ 0 & \tau_r^2 \end{bmatrix} - \frac{1}{\sigma_p^2 + \tau_r^2 + \sigma^2} \begin{bmatrix} \sigma_p^4 & \sigma_p^2\tau_r^2 \\ \sigma_p^2\tau_r^2 & \tau_r^4 \end{bmatrix}$$

(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 $Q_{pr}(y^{(pr)}, z^{(pr)})$.]

Solution

For the M-step, we are trying to maximize w.r.t $[\mu_p, v_r, \sigma_p^2, \tau_r^2]$,
$\sum_{p,r=1}^{P,R} \int_z^{pr} \int_y^{pr} Q_{pr}(y^{pr}, z^{pr}) log \frac{P(x^{pr}, y^{pr}, z^{pr}; \mu_p, v_p, \sigma_p^2, \tau_r^2)}{Q_{pr}(y^{pr}, z^{pr})} dydz$

$$= \sum_{p,r=1}^{P,R} \int_z^{pr} \int_y^{pr} Q_{pr}(y^{pr}, z^{pr}) log P(x^{pr}, y^{pr}, z^{pr}; \mu_p, v_p, \sigma_p^2, \tau_r^2) dydz \quad (1)$$

$$= \sum_{p,r=1}^{P,R} \int_z^{pr} \int_y^{pr} Q_{pr}(y^{pr}, z^{pr}) log P(x^{pr}|y^{pr}, z^{pr}) P(y^{pr}|\mu_p, \sigma_p^2) P(z^{pr}|v_p, \tau_r^2) dydz \quad (2)$$

$$= \sum_{p,r=1}^{P,R} \int_z^{pr} \int_y^{pr} Q_{pr}(y^{pr}, z^{pr}) log P(y^{pr}|\mu_p, \sigma_p^2) + log P(z^{pr}|v_p, \tau_r^2) dydz \quad (3)$$

$$= \sum_{p,r=1}^{P,R} E_{y^{pr},z^{pr}|x^{pr}}[logP(y^{pr}|\mu_p,\sigma_p^2) + logP(z^{pr}|\upsilon_p,\tau_r^2)] \quad (4)$$

(1) is due to the denominator Q being constant and therefore not contributing to the expression to be maximized (2) splitting up P(x,y,z) (3)

From here, the solution proceeds by substituting in the probability distributions of y and z and taking the partial derivative with respect to the parameters. and setting it to zero.

2.4 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.]

Solution

Claim: $\forall P, QKL(P||Q) \geq 0$

Proof:

$$KL(P||Q) = \sum_x P(x)log\frac{P(x)}{Q(x)} = E_{P(x)}[-log\frac{Q(x)}{P(x)}]$$

$$\geq -logE_{P(x)}[\frac{Q(x)}{P(x)}] = -log\sum_x P(x)\frac{Q(x)}{P(x)} = -log\sum_x Q(x) = -log1 = 0 \quad \diamond$$

From Jensen's Inequality, the previous result holds with equality instead of inequality if and only if $E[\frac{Q(x)}{P(x)}] = c$

$\sum_x P(x)\frac{Q(x)}{P(x)} = c$

$\sum_x Q(x) = 1 = c \to \frac{Q(x)}{P(x)} = 1 \longrightarrow P = Q \quad \diamond$

(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:

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

Solution

$$KL(P(X,Y)||Q(X,Y)) = KL(P(Y|X)P(X)||Q(Y|X)Q(X))$$

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

$$= \sum_x P(x)\sum_y P(y|x)log\frac{P(y|x)}{Q(y|x)} + log\frac{P(x)}{Q(x)}$$

$$= \sum_x P(x)\sum_y P(y|x)log\frac{P(y|x)}{Q(y|x)} + \sum_x P(x)log\frac{P(x)}{Q(x)}\sum_y P(y|x)$$

$$= \sum_x P(x)\sum_y P(y|x)log\frac{P(y|x)}{Q(y|x)} + \sum_x P(x)log\frac{P(x)}{Q(x)}$$

$$\longrightarrow = KL(P(Y|X)||Q(Y|X)) + KL(P(X)||Q(X)) \quad \diamond$$

(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 $\hat{P}(x) = \frac{1}{m}\sum_{i=1}^{m} 1\{x^{(i)} = x\}$.

($\hat{P}$ 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:

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

Solution

$$argmin_\theta\ KL(P||P_\theta) = argmin_\theta\ \sum_x P(x) log \frac{P(x)}{P_\theta(x)}$$

$$= argmin_\theta\ \sum_x -P(x) log P_\theta(x) = argmax_\theta\ \sum_x P(x) log P_\theta(x)$$

$$= argmax_\theta\ \sum_x \frac{1}{m}\sum_{i=1}^{m} 1[x^{(i)} = x] log P_\theta(x) = argmax_\theta\ \frac{1}{m}\sum_{i=1}^{m}\sum_x 1[x^{(i)} = x] log P_\theta(x)$$

$$= argmax_\theta\ \sum_{i=1}^{m} log P_\theta(x^{(i)}) \quad \diamond$$

2.5 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`

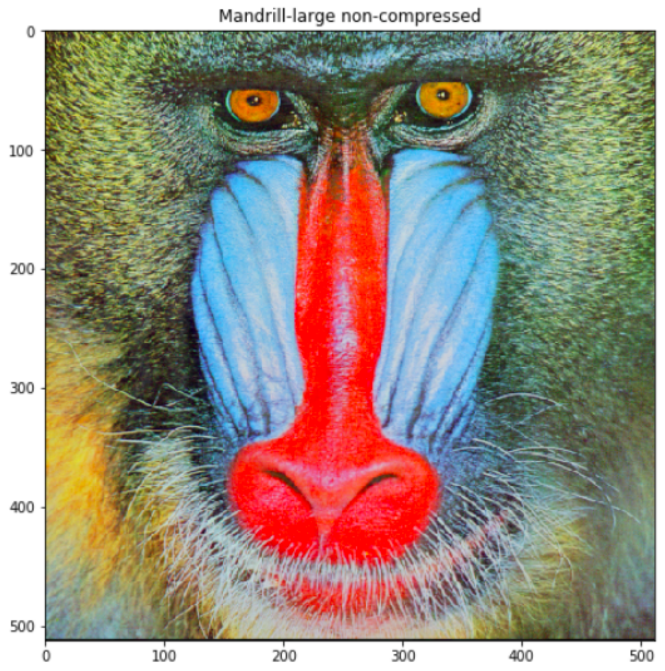
The `mandrill-large.tiff` file contains a 512x512 image of a mandrill represented in 24-bit color. This means that, for each of the 262144 pixels in the image, there are three 8-bit numbers (each ranging from 0 to 255) that represent the red, green, and blue intensity values for that pixel. The straightforward 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.

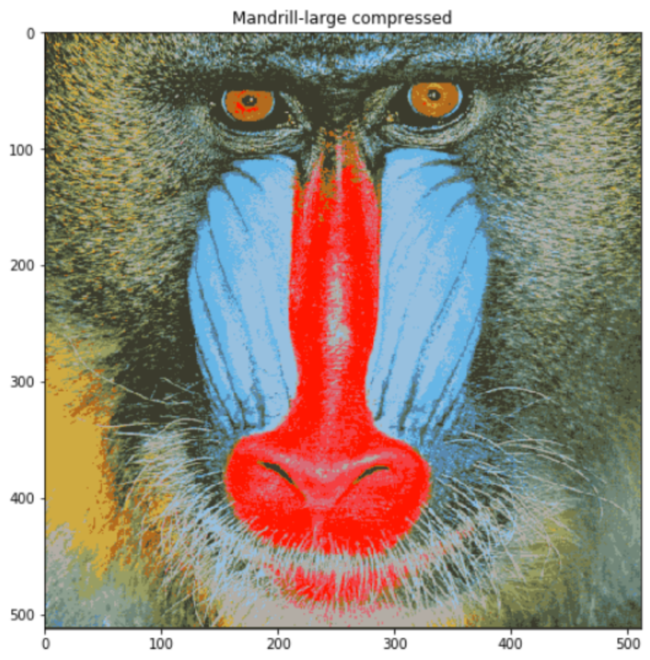
Python: Start up Python, and type
`from matplotlib.image import imread; import matplotlib.pyplot as plt;`
and run `A = imread('mandrill-large.tiff')` . Now, `A` is a "three dimensional matrix," and `A[:,:,0]`, `A[:,:,1]` and `A[:,:,2]` are 512x512 arrays that respectively contain the red, green, and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

Code for this problem can be found here



(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?

Solution
Before, the image was made of 512 x 512 = 262,144 pixels with each pixel being made up of three 8 bit numbers. After compressing, our image has 262,144 pixels each with only 4 bits. Thus, we have compressed our image by a factor of $\frac{3*8}{1*4} = 6$.