EECS 545 – Machine Learning - Homework #4

Daniel LeJeune & Benjamin Bray Due: 11:00pm 03/21/2016

Homework Policy: Working in groups is fine, but each member must submit their own writeup. Please write the members of your group on your solutions. There is no strict limit to the size of the group but we may find it a bit suspicious if there are more than 4 to a team. **For coding problems, please include your code and report your results (values, plots, etc.)** in your PDF submission. You will lose points if your experimental results are only accessible through rerunning your code. Homework will be submitted via Gradescope (https://gradescope.com/).

Solution contributors: Luyao Yuan

1) **Information Theory, (20 pts).**

Many algorithms for learning probabilistic models are best understood in terms of *information theory*. Consequently, it is useful to understand and manipulate these quantities in different contexts.

(a) Show that

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

where $I(X,Y)$ is the mutual information of X and Y, $H(X)$ is the entropy of X, and $H(X|Y)$ is the conditional entropy of X given Y.

$$\begin{aligned}
I(x,y) &= \int\int p(x,y)\log(\frac{p(x,y)}{p(x)p(y)})dxdy \\
&= \int\int p(x,y)\log(p(x|y))dxdy - \int\int p(x,y)\log(p(x))dxdy \\
&= \int\int p(x,y)\log(p(x|y))dxdy - \int \log(p(x))\int p(x,y)dydx \\
&= \int\int p(x,y)\log(p(x|y))dxdy - \int \log(p(x))p(x)dx \\
&= \int\int p(x,y)\log(p(x|y))dxdy + H(X) \\
&= \int\int p(y)p(x|y)\log(p(x|y))dxdy + H(X) \\
&= \int p(y)\int p(x|y)\log(p(x|y))dxdy + H(X) \\
&= \int p(y)H(X|y)dy + H(X) \\
&= H(X) - H(X|Y)
\end{aligned}$$

The other equality holds by symmetry, so $I(X,Y) = H(Y) - H(X|Y) = H(X) - H(X|Y)$.

(b) Prove that if X and Y are related by a bijection f (i.e., $X = f(Y)$ and $Y = f^{-1}(X)$), then $I(X,Y) = H(X) = H(Y)$.

Using the answer from **(a)**, we just need to show that $H(X|Y) = 0$.

$$\begin{aligned} H(X|Y) &= \mathbb{E}_Y[\mathbb{E}_X[\log p_X(X|Y)|Y]] \\ &= \mathbb{E}_Y[\log p_X(f^{-1}(Y)|Y)] \\ &= \mathbb{E}_Y[\log 1] \\ &= 0 \end{aligned}$$

(c) Suppose we observe N samples $\mathcal{D} = (x_1, \ldots, x_N)$ from some unknown distribution. Define $\hat{p}(x)$ to be the empirical probability density estimate,

$$\hat{p}(x) \triangleq \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}[x = x_i]$$

Let $q(x|\theta)$ be the probability density corresponding to some known probabilistic model with parameter θ. Show that the minimum Kullback–Leibler divergence

$$\min_{\theta} D_{KL}(\hat{p}||q)$$

is obtained by the maximum likelihood estimate θ_{ML} given the data $\mathcal{D}$.

$$\begin{aligned} D_{KL}(\hat{p}||q) &= \int \hat{p}(x) \log\left(\frac{\hat{p}(x)}{q(x|\theta)}\right) dx \\ &= \int \hat{p}(x) \log(\hat{p}(x)) - \hat{p}(x) \log(q(x|\theta)) dx \\ &= H(\hat{p}(x)) - \int \hat{p}(x) \log(q(x|\theta)) dx \\ &= H(\hat{p}(x)) - E_{\hat{p}(x)}[\log(q(x|\theta))] \end{aligned}$$

$H(\hat{p}(x))$ is a fixed value, so minimizing $D_{KL}(\hat{p}||q)$ is equivalent to maximizing $E_{\hat{p}(x)}[\log(q(x|\theta))]$. $\hat{p}(x)$ is a discrete distribution, so $E_{\hat{p}(x)}[\log(q(x|\theta))] = \sum_x \hat{p}(x) \log(q(x|\theta))$

$$\begin{aligned} \sum_x \hat{p}(x) \log(q(x|\theta)) &= \sum_x (\frac{1}{N} \sum_i^N \mathbf{1}(x = x_i)) \log(q(x|\theta)) \\ &= \frac{1}{N} \sum_i^N \log(q(x_i|\theta)) \\ &= \frac{1}{N} \log(\prod_i^N q(x_i|\theta)) \\ &= \frac{1}{N} \log(q(\mathbf{X}|\theta)) \end{aligned}$$

If multiplied by N, this is exactly the log-likelihood.

(d) Let $p = \mathcal{N}(\mu, \sigma^2)$ be a Gaussian distribution and q be any probability density with mean μ and variance σ^2. Prove that $H(q) \leq H(p)$, that is, the Gaussian distribution has maximum entropy among all distributions of the same variance. *Hint: Refer to the textbook (PRML by Bishop §1.6) for a proof outline.*

We want to maximize $-\int p(x)\log(p(x))dx$, given p(x) is a probability distribution with mean μ and variance σ^2 Transfer that to a Lagrange optimization problem. We only need to minimize

$$\begin{aligned} L(\lambda, p(x)) = &\int p(x)\log(p(x))dx + \lambda_1(\int p(x)dx - 1) \\ &+ \lambda_2(\int xp(x)dx - \mu) + \lambda_3(\int (x-\mu)^2 p(x)dx - \sigma^2) \end{aligned}$$

with respect to p(x).
Define functional $I[p(x)] = \int p(x)f(x)dx$ and let $\eta(x)$ be an arbitrary function of x. Hence,

$$\begin{aligned} I[p(x) + \varepsilon\eta(x)] &= \int (p(x) + \varepsilon\eta(x))f(x)dx \\ &= \int p(x)f(x)dx + \varepsilon\int \eta(x)f(x)dx \\ \therefore \frac{\delta I}{\delta p(x)} &= f(x) \end{aligned}$$

Define functional $J[p(x)] = \int p(x)\log(p(x))dx$ and let $\eta(x)$ be an arbitrary function of x. Hence,

$$\begin{aligned} J[p(x) + \varepsilon\eta(x)] &= \int (p(x) + \varepsilon\eta(x))\log(p(x) + \varepsilon\eta(x))dx \\ &= \int p(x)\log(p(x))dx + \varepsilon(\int \eta(x)\log(p(x))dx + \int \eta(x)p(x)\frac{1}{p(x)}dx) \\ \therefore \frac{\delta J}{\delta p(x)} &= \log(p(x)) + 1 \end{aligned}$$

$\therefore$

$$\frac{\delta L(\lambda, p(x))}{\delta p(x)} = \log(p(x)) + 1 + \lambda_1 + \lambda_2 x + \lambda_3(x-\mu)^2$$

Let's set that gradient to 0, and we get

$$\begin{aligned} p(x) &= \exp(-1 - \lambda_1 - \lambda_2 x + \lambda_3(x-\mu)^2) \\ &= \exp(-a(x-b)^2 + c) \\ &= \exp(-a(x-b)^2)\exp(c) \end{aligned}$$

We can put it into this last form because we know the term inside the exponential must be a concave quadratic – otherwise, the distribution would not have finite moments. We see that the distribution is symmetric about b, so we must have $b = \mu$. We must also have

$$\begin{aligned} c &= -\log\int \exp(-a(x-b)^2)dx \\ &= -\log\sqrt{\frac{\pi}{a}} \end{aligned}$$

Making the substitution $a = 1/2s^2$, we get

$$p(x) = \frac{1}{\sqrt{2\pi}s} \exp\left(-\frac{1}{2s^2}(x-b)^2\right)$$

And we see that this is the Gaussian distribution.

2) **Dirichlet Maximum Likelihood (20 pts).**

In this problem, you will derive and implement a Newton-Raphson algorithm for maximizing the Dirichlet log-likelihood function. Unlike for the simple distributions we have encountered in the past (Multinomial, Poisson, etc.), no closed-form solution exists for the Maximum Likelihood estimate of Dirichlet parameters.

Recall a Dirichlet-distributed random vector $\boldsymbol{p} = (p_1, \ldots, p_m) \in \Delta^{m-1}$ governed by nonnegative concentration parameters $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)$ has following distribution,

$$\text{Dirichlet}(\boldsymbol{p}|\boldsymbol{\alpha}) = \frac{\Gamma(\sum_{k=1}^m \alpha_k)}{\prod_{k=1}^m \Gamma(\alpha_k)} \prod_{k=1}^m p_k^{\alpha_k - 1}$$

where $\Gamma(t)$ is the *Gamma function* and Δ^{m-1} is the unit simplex in $\mathbb{R}^m$.

(a) Show that the Dirichlet distribution belongs to the *exponential family*, that is, find a natural parameter $\eta \triangleq \eta(\boldsymbol{\alpha})$, a sufficient statistics function $T(\boldsymbol{p})$, and a log-partition function $A(\boldsymbol{\alpha})$ such that

$$\text{Dirichlet}(\boldsymbol{p}|\boldsymbol{\alpha}) = \exp\left[\eta(\boldsymbol{\alpha})^T T(\boldsymbol{p}) - A(\boldsymbol{\alpha})\right]$$

This guarantees that the log-likelihood is convex and Newton's method will converge to a global optimum.

(1) Observe that, by the usual trick, $\text{Dirichlet}(\boldsymbol{p}|\boldsymbol{\alpha}) = \exp[\log \text{Dirichlet}(\boldsymbol{p}|\boldsymbol{\alpha})]$, so

$$\text{Dirichlet}(\boldsymbol{p}|\boldsymbol{\alpha}) = \exp\left[\sum_{k=1}^m (\alpha_k - 1)\log p_k + \log\Gamma\left(\sum_{k=1}^m \alpha_k\right) - \sum_{k=1}^m \log\Gamma(\alpha_k)\right]$$

(2) It is easy to see now that the Dirichlet takes exponential family form, with

$$\begin{aligned}
\eta(\boldsymbol{\alpha}) &= \boldsymbol{\alpha} - 1 = (\alpha_1 - 1, \ldots, \alpha_m - 1) \\
T(\boldsymbol{p}) &= \log \boldsymbol{p} = (\log p_1, \ldots, \log p_m) \\
A(\boldsymbol{\alpha}) &= \sum_{k=1}^m \log\Gamma(\alpha_k) - \log\Gamma\left(\sum_{k=1}^m \alpha_k\right)
\end{aligned}$$

(b) Given observations $\mathcal{D} = (\boldsymbol{p}^{(1)}, \ldots, \boldsymbol{p}^{(N)})$, derive an expression for the Dirichlet log-likelihood function $F(\boldsymbol{\alpha}) = \log P(\mathcal{D}|\alpha)$ in terms of the *observed sufficient statistics*,

$$\hat{t}_k = \frac{1}{N}\sum_{j=1}^N \log p_k^{(j)}$$

(1) Observe that

$$F(\alpha) = \log P(\mathcal{D}|\alpha) = \log \prod_{j=1}^{N} P(\boldsymbol{p}_j|\alpha) = \sum_{j=1}^{N} \log P(\boldsymbol{p}_j|\alpha)$$

(2) Here, the exponential family form comes in handy. We have

$$\begin{aligned} F(\alpha) &= \sum_{j=1}^{N} \left[\sum_{k=1}^{m} (\alpha_k - 1) \log p_{jk} + \log \Gamma \left(\sum_{k=1}^{m} \alpha_k \right) - \sum_{k=1}^{m} \log \Gamma(\alpha_k) \right] \\ &= \sum_{k=1}^{m} (\alpha_k - 1) \sum_{j=1}^{N} \log p_{jk} + N \left[\log \Gamma \left(\sum_{k=1}^{m} \alpha_k \right) - \sum_{k=1}^{m} \log \Gamma(\alpha_k) \right] \\ &= N \left[\sum_{k=1}^{m} (\alpha_k - 1) \hat{t}_k + \log \Gamma \left(\sum_{k=1}^{m} \alpha_k \right) - \sum_{k=1}^{m} \log \Gamma(\alpha_k) \right] \end{aligned}$$

(c) Derive an expression for the gradient of the log-likelihood in terms of the observed sufficient statistics and the *digamma function*,

$$\Psi(t) = \frac{\mathrm{d} \log \Gamma(t)}{\mathrm{d}t} = \frac{\Gamma'(t)}{\Gamma(t)}$$

(*Hint:* Specify each component $\frac{\partial F}{\partial \alpha_k}$ separately, rather than trying to use matrix operations.)

(1) Using the expression we found in Part B for the log-likelihood,

$$\begin{aligned} (\nabla F)_k = \frac{\partial F}{\partial \alpha_k} &= N \left[\sum_{k=1}^{m} \frac{\partial}{\partial \alpha_k} \left[(\alpha_k - 1) \hat{t}_k \right] + \frac{\partial}{\partial \alpha_k} \left[\log \Gamma \left(\sum_{k=1}^{m} \alpha_k \right) \right] - \sum_{k=1}^{m} \frac{\partial}{\partial \alpha_k} \left[\log \Gamma(\alpha_k) \right] \right] \\ &= N \left[\hat{t}_k + \frac{\partial}{\partial \alpha_k} \left[\log \Gamma \left(\sum_{k=1}^{m} \alpha_k \right) \right] - \Psi(\alpha_k) \right] \end{aligned}$$

(2) We are left with one problematic derivative. Applying the chain rule twice,

$$\begin{aligned} \frac{\partial}{\partial \alpha_k} \left[\log \Gamma \left(\sum_{k=1}^{m} \alpha_k \right) \right] &= \frac{1}{\Gamma \left(\sum_{k=1}^{m} \alpha_k \right)} \frac{\partial}{\partial \alpha_k} \left[\Gamma \left(\sum_{k=1}^{m} \alpha_k \right) \right] \\ &= \frac{\Gamma' \left(\sum_{k=1}^{m} \alpha_k \right)}{\Gamma \left(\sum_{k=1}^{m} \alpha_k \right)} = \Psi \left(\sum_{k=1}^{m} \alpha_k \right) \end{aligned}$$

(3) Finally,

$$(\nabla F)_k = \frac{\partial F}{\partial \alpha_k} = N \left[\hat{t}_k + \Psi \left(\sum_{k=1}^{m} \alpha_k \right) - \Psi(\alpha_k) \right]$$

(d) Show that the Hessian matrix $H \triangleq \nabla^2_{\boldsymbol{\alpha}} F(\boldsymbol{\alpha})$ of the log-likelihood can be written as the sum of a diagonal matrix and a matrix whose elements are all the same,

$$H = \nabla^2_{\boldsymbol{\alpha}} F(\boldsymbol{\alpha}) = Q + c \mathbf{1}\mathbf{1}^T$$

where $c \in \mathbb{R}$ is a constant and $Q \in \mathbb{R}^{m \times m}$ is diagonal with entries $q_{11}, \ldots, q_{mm}$. (*Note:* It is okay to simply write Ψ' for the derivative of the digamma function.)

(1) First, note that by the chain rule,

$$\frac{\partial}{\partial \alpha_j}\left[\Psi\left(\sum_{k=1}^{m}\alpha_k\right)\right] = \Psi'\left(\sum_{k=1}^{m}\alpha_k\right)$$

(2) Taking second derivatives, the observed sufficient statistics disappear since they do not depend on the parameters $\boldsymbol{\alpha}$, therefore

$$\frac{\partial^2 F}{\partial \alpha_k^2} = N\left[\Psi'\left(\sum_{k=1}^{m}\alpha_k\right) - \Psi'(\alpha_k)\right]$$
$$\frac{\partial^2 F}{\partial \alpha_k \alpha_j} = N\Psi'\left(\sum_{k=1}^{m}\alpha_k\right) \qquad (k \neq j)$$

(3) We can rewrite this as

$$H = Q + c11^T$$
$$q_{jk} = -N\Psi'(\alpha_k)\mathbb{I}_{j=k}$$
$$c = N\Psi'\left(\sum_{k=1}^{m}\alpha_k\right)$$

(e) The Newton-Raphson method provides a quadratically converging method for parameter estimation. The general update rule can be written in terms of the Hessian matrix as follows:

$$\boldsymbol{\alpha}^{new} = \boldsymbol{\alpha}^{old} - [H_F(\boldsymbol{\alpha}^{old})]^{-1} \cdot \nabla F(\boldsymbol{\alpha}^{old})$$

Use the Sherman-Morrison matrix inversion lemma to derive a closed-form update for $\boldsymbol{\alpha}$ (any remaining matrix inversions should be diagonal).

(1) From the matrix inversion lemma, we know that

$$H^{-1} = Q^{-1} - \frac{Q^{-1}11^TQ^{-1}}{1/c + 1^TQ^{-1}1}$$

(f) Implement this Newton-Raphson update in your language of choice:

- Generate $N = 1000$ samples from the Dirichlet distribution with parameter $\boldsymbol{\alpha} = (10, 5, 15, 20, 50)$ (see `hw4p2.py` on Canvas). Use an initial estimate of $\hat{\boldsymbol{\alpha}} = (1, 1, 1, 1, 1)$.
- The following Python functions may be useful: `from scipy.special import gammaln, polygamma`.

Deliverables:

- A plot of the log-likelihood as a function of iteration number. Also plot the log-likelihood given the true parameters as a constant (horizontal) line. Terminate your algorithm when the log-likelihood increases by less than 10^{-4}.
- The estimated model parameters.
- Please submit you code, as usual.

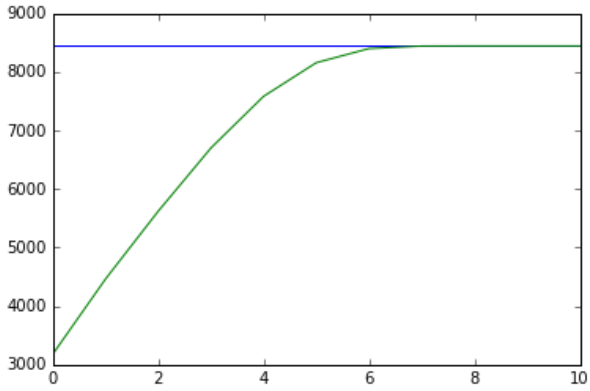


Figure 1: Log-likelihood vs. iteration. The blue line is the log-likelihood with the true parameters.

See Figure 1 for a plot of the log-likelihood. Usually the likelihood exceeds the likelihood of the true parameters slightly.

The estimated model parameters should be close to the true (see above), but error of a few percent is not uncommon, and is more noticeable for the larger elements of alpha.

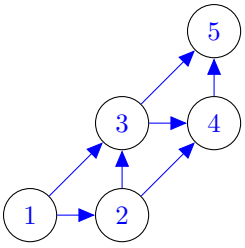
See `hw4p2_sol.py` for code solution.

3) **Graphical Models (15 pts).**

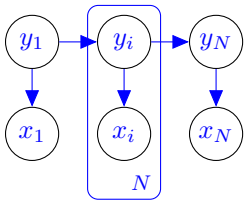
In this problem, you will explore the independence properties of directed graphical models and practice translating them to factored probability distributions and back.

(a) Draw a directed graphical model for each of the following factored distributions. Take advantage of plate notation when convenient, and represent as many independencies with your graph as possible (i.e., don't draw a fully connected graph!).

(i) $P(y_1, y_2, y_3, y_4, y_5) = P(y_1)P(y_2|y_1)\prod_{k=3}^{5} P(y_k|y_{k-1}, y_{k-2})$

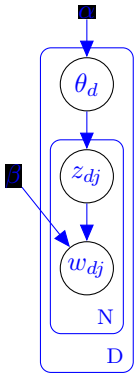


(ii) $P(x_1, \ldots, x_N, y_1, \ldots, y_N) = P(y_1)\prod_{k=2}^{N} P(y_k|y_{k-1})\prod_{k=1}^{N} P(x_k|y_k)$



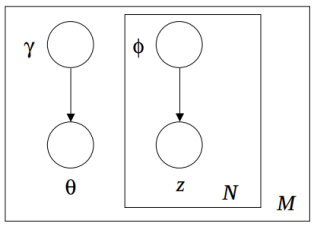
(b) Draw a directed graphical model for the following model specification, where $\alpha \in \mathbb{R}^K$ and $\beta \in \mathbb{R}^{K \times W}$ are fixed hyperparameters, and $\boldsymbol{\beta}_k$ denotes the k^{th} row of β:

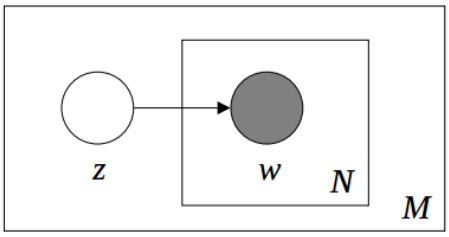
$$\begin{aligned} \boldsymbol{\theta}_1 \ldots, \boldsymbol{\theta}_d &\overset{iid}{\sim} \text{Dirichlet}(\boldsymbol{\alpha}) && \\ z_{d1}, \ldots, z_{dN} | \boldsymbol{\theta}_d &\overset{iid}{\sim} \text{Categorical}(\boldsymbol{\theta}_d) && \forall\, d \in \{1, \ldots, D\} \\ w_{dj} | z_{dj} &\sim \text{Categorical}(\boldsymbol{\beta}_{z_{dj}}) && \forall\, d \in \{1, \ldots, D\}, j \in \{1, \ldots, N\} \end{aligned}$$



α is the solid node above and β is the solid node on the left. They are not latent variables.

(c) For each directed model below, write down the factorized joint distribution over all variables.





$P(\gamma_1, \ldots, \gamma_M, \theta_1, \ldots, \theta_M, \phi_{11}, \ldots, \phi_{1N}, \ldots, \phi_{M1}, \ldots, \phi_{MN}, z_{11}, \ldots, z_{1N}, \ldots, z_{M1}, \ldots, z_{MN})$
$= \prod_{i=1}^{M} P(\gamma_i) P(\theta_i | \gamma_i) \prod_{j=1}^{M} \prod_{k=1}^{N} P(\phi_{jk}) P(z_{jk} | \phi_{jk})$

$P(z_1, \ldots, z_M, w_{11}, \ldots, w_{1N}, \ldots, w_{M1}, \ldots, w_{MN}) = \prod_{i=1}^{M} P(z_i) \prod_{j=1}^{M} \prod_{k=1}^{N} P(w_{jk} | z_j)$

4) **Clustering (20 points).**

Download the image `mandrill.png` from Canvas. In this problem you will apply the k-means algorithm to image compression. In this context it is also known as the Lloyd-Max algorithm.

(a) First, partition the image into blocks of size $M \times M$ and reshape each block into a vector of length $3M^2$ (see `hw4p4.py` on Canvas). The 3 comes from the fact that this is a color image, and so there are three intensities for each pixel. Assume that M, like the image dimensions, is a power of 2.

Next, write a program that will cluster the vectors from **(a)** using the k-means algorithm. **You should implement the k-means algorithm yourself.** Please initialize the cluster means to be randomly selected data points, sampled without replacement.

Finally, reconstruct a quantized version of the original image by replacing each block in the original image by the nearest centroid. Test your code using $M = 2$ and $k = 64$.

Deliverables:

- A plot of the k-means objective function value versus iteration number.
- A description of how the compressed image looks compared to the original. What regions are best preserved, and which are not?
- A picture of the difference of the two images. You should add a neutral gray $(128, 128, 128)$ to the difference before generating the image.
- The compression ratio (use your forumla from **(b)**).
- The relative mean absolute error of the compressed image, defined as

$$\frac{\frac{1}{3N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{r=1}^{3}|\tilde{I}(i,j,r) - I(i,j,r)|}{255}$$

  where $\tilde{I}$ and I are the compressed and original images, respectively, viewed as 3-D arrays. This quantity can be viewed as the average error in pixel intensity relative to the range of pixel intensities.

- Please submit you code, as usual.

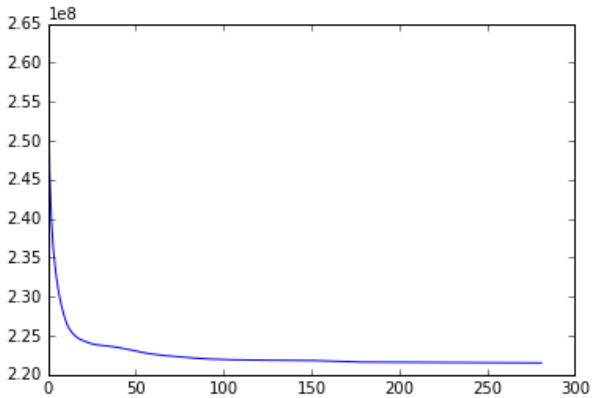


Figure 3: Objective function vs. iteration for k-means

See Figure 3 for a plot of objective function value.

In general, the smoother the area, the better preserved, but even though the hairs are very non-smooth, the degradation is not very noticeable. The eyes in the compressed image have the most noticeable compression artifacts.

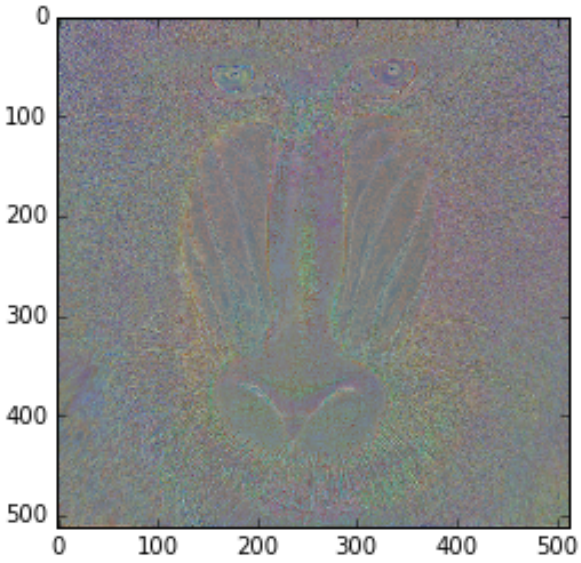


Figure 4: Difference of compressed and original images

See Figure 4 for the difference of the two images.

The compression ratio is bpp/24 $\approx 6.348\%$.

The relative mean absolute error should be around 0.05.

See `hw4p4_sol.py` for a code solution.

(b) The original uncompressed image uses 24 bits per pixel (bpp), 8 bits for each color. Assuming an image of size $N \times N$, where N is a power of 2, what is the number of bits per pixel, as a function of M, N, and k, needed to store the compressed image? What is the compression ratio, which is defined as the ratio of bpp in the compressed image relative to the original uncompressed image? *Hint: To calculate the bpp of the compressed image, imagine you need to transmit the compressed image to a friend who knows the compression strategy as well as the values of M, N, and k.*

We need to store each of the cluster centers' pixel values, and we need to store the cluster assignments, so

$$\text{bpp} = \frac{\#\text{ bits}}{\#\text{ pixels}} = \frac{24M^2k + \frac{N^2}{M^2}\log_2 k}{N^2}$$

(c) (Optional, ungraded) Play around with M and k.

5) **EM Algorithm for Mixed Linear Regression (25 points).**

Consider regression training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, iid realizations of $(\boldsymbol{X}, Y) \in \mathbb{R}^d \times \mathbb{R}$, where the conditional distribution of Y given $\boldsymbol{X}$ is modeled by the pdf

$$f(y|\mathbf{x};\boldsymbol{\theta}) \sim \sum_{k=1}^{K} \pi_k \phi(y; \mathbf{w}_k^T \mathbf{x} + b_k, \sigma_k^2),$$

where $\boldsymbol{\theta} = (\pi_1, \ldots, \pi_K, \mathbf{w}_1, \ldots, \mathbf{w}_K, b_1, \ldots, b_K, \sigma_1^2, \ldots, \sigma_K^2)$ is a list of the model parameters such that $\pi_k \geq 0$ and $\sum_{k=1}^K \pi_k = 1$, and $\phi(y; \mu, \sigma^2)$ is the pdf of a Gaussian random variable with mean μ and variance σ^2 evaluated at y. Viewing the $\mathbf{x}_i$ as deterministic, derive an EM algorithm for maximizing the likelihood by following these steps.

(a) Denote $\underline{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ and $\underline{y} = (y_1, \ldots, y_N)$. Write down the formula for the log-likelihood, $\log f(\underline{y}|\underline{\mathbf{x}};\boldsymbol{\theta})$, where $f(\underline{y}|\underline{\mathbf{x}};\boldsymbol{\theta})$ is the model (with parameters $\boldsymbol{\theta}$) for $\underline{y}$ given $\underline{\mathbf{x}}$.

$$\begin{aligned}
\log f(\underline{y}|\underline{\mathbf{x}};\boldsymbol{\theta}) &= \log \prod_{i=1}^{N} f(y_i|\mathbf{x}_i;\boldsymbol{\theta}) \\
&= \sum_{i=1}^{N} \log f(y_i|\mathbf{x}_i;\boldsymbol{\theta}) \\
&= \sum_{i=1}^{N} \log \sum_{k=1}^{K} \pi_k \phi(y; \mathbf{w}_k^T \mathbf{x}_i + b_k, \sigma_k^2)
\end{aligned}$$

(b) Introduce hidden variable $\underline{z} = (z_1, \ldots, z_N)$ which determines the mixture component that y is drawn from, i.e., $f(y|\mathbf{x}, z = k;\boldsymbol{\theta}) = \phi(y; \mathbf{w}_k^T \mathbf{x} + b_k, \sigma_k^2)$. Write down the complete-data log-likelihood, $\log f(\underline{y}, \underline{z}|\underline{\mathbf{x}};\boldsymbol{\theta})$. *Hint: Define a random variable* $\Delta_{ik} \triangleq \mathbb{I}[z_i = k]$ *so that you can write the log-likelihood as a double sum and so that* $\underline{z}$ *only appears in the expression through* Δ_{ik}.

$$\begin{aligned}
\log f(\underline{y}, \underline{z}|\underline{\mathbf{x}};\boldsymbol{\theta}) &= \log \prod_{i=1}^{N} f(y_i, z_i|\mathbf{x}_i;\boldsymbol{\theta}) \\
&= \sum_{i=1}^{N} \log f(y_i, z_i|x_i;\theta) \\
&= \sum_{i=1}^{N} \log \prod_{k=1}^{K} (\pi_k \phi(y_i; \mathbf{w}_k^T \mathbf{x}_i + b_k, \sigma_k^2))^{\Delta_{ik}} \\
&= \sum_{i=1}^{N} \sum_{k=1}^{K} \Delta_{ik} \log(\pi_k \phi(y_i; \mathbf{w}_k^T \mathbf{x}_i + b_k, \sigma_k^2))
\end{aligned}$$

(c) Determine the E-step. Give an explicit formula for

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \mathbb{E}_{\underline{z}} \left[\log f(\underline{y}, \underline{z}|\underline{\mathbf{x}};\boldsymbol{\theta}) | \underline{y}, \underline{\mathbf{x}}; \boldsymbol{\theta}^{\text{old}} \right]$$

in terms of $\boldsymbol{\theta}$, $\boldsymbol{\theta}^{\text{old}}$, and the data. Remember that in this expectation, you should treat only $\underline{z}$ as a random variable, and the distribution of $\underline{z}$ is conditioned on $\underline{y}$ and $\underline{\mathbf{x}}$ and governed by the parameters $\boldsymbol{\theta}^{\text{old}}$. The log-likelihood inside the expectation, on the other hand, is parameterized by $\boldsymbol{\theta}$ and is non-random for a fixed $\underline{z}$.

$$
\begin{aligned}
Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) &= \mathbb{E}_{\underline{z}}[\log(f(\underline{y}, \underline{z}|\underline{\mathbf{x}}, \boldsymbol{\theta}))] \\
&= \sum_{i=1}^{N}\sum_{k=1}^{K} \mathbb{E}_{\underline{z}}[\Delta_{ik} log(\pi_k \phi(y_i; \mathbf{w}_k^T \mathbf{x}_i + b_k, \sigma_k^2))] \\
&= \sum_{i=1}^{N}\sum_{k=1}^{K} \mathbb{E}_{\underline{z}}[\Delta_{ik}] \log(\pi_k \phi(y_i; \mathbf{w}_k^T \mathbf{x}_i + b_k, \sigma_k^2)) \\
&= \sum_{i=1}^{N}\sum_{k=1}^{K} p(z_i|y_i, \mathbf{x}_i; \theta^{old}) log(\pi_k \phi(y_i; \mathbf{w}_k^T \mathbf{x}_i + b_k, \sigma_k^2)) \\
&= \sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{ik} \log(\pi_k \phi(y_i; \mathbf{w}_k^T \mathbf{x}_i + b_k, \sigma_k^2)) \\
&= \sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{ik} \log(\pi_k) + \sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{ik} \phi(y_i; \mathbf{w}_k^T \mathbf{x}_i + b_k, \sigma_k^2))
\end{aligned}
$$

We define $\gamma_{ik} = p(z_i = k|y_i, \mathbf{x}_i; \boldsymbol{\theta}^{old})$:

$$
\begin{aligned}
p(z = k|y, \mathbf{x}; \boldsymbol{\theta}^{old}) &= \frac{p(z = k, y|\mathbf{x}; \boldsymbol{\theta}^{old})}{p(y|x; \boldsymbol{\theta}^{old})} \\
&= \frac{p(z = k, y|\mathbf{x}; \boldsymbol{\theta}^{old})}{\sum_{k=1}^{K} p(y, z = k|x; \boldsymbol{\theta}^{old})} \\
&= \frac{p(y|z = k, \mathbf{x}; \theta^{old}) p(z = k|x; \boldsymbol{\theta}^{old})}{\sum_{k=1}^{K} p(y, z = k|x; \theta^{old})} \\
&= \frac{\pi_k \phi(y; \mathbf{w}_k^T \mathbf{x} + b_k, \sigma_k^2)}{\sum_{k'=1}^{K} \pi_{k'} \phi(y; \mathbf{w}_{\mathbf{k}'}^{\mathbf{T}} \mathbf{x} + b_{k'}, \sigma_{k'}^2)}
\end{aligned}
$$

(d) Determine the M-step. That is, determine the $\boldsymbol{\theta}$ that maximizes $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$. *Suggestions:* Use Lagrange multiplier theory to optimize the weights π_k. To optimize $(\mathbf{w}_k, b_k, \sigma_k^2)$, first hold σ_k^2 fixed and find the optimal $(\mathbf{w}_k, b_k)$, then plug that in and find the optimal σ_k^2. Just treat σ_k^2 as a variable (not the square of a variable).

First we calculate π_k:

$$
\begin{aligned}
L(\theta, \lambda) &= Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) + \lambda(\sum_{k=1}^{K} \pi_k - 1) \\
\because Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) &= \sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{ik} \log(\pi_k) + \sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{ik} \phi(y_i; \mathbf{w}_k^T \mathbf{x}_i + b_k, \sigma_k^2)) \\
&= \sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{ik} \log(\pi_k) + \sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{ik} \log \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(y_i - \mathbf{w}_k^T \mathbf{x}_i - b_k)^2}{2\sigma_k^2}} \\
\therefore \frac{\delta L}{\delta \pi_k} &= \sum_{i=1}^{N} \gamma_{ik} \frac{1}{\pi_k} + \lambda = 0 \\
\therefore \pi_k &= \frac{\sum_{i=1}^{N} \gamma_{ik}}{-\lambda} \\
\because \sum_{k=1}^{K} \pi_k &= 1 \\
\therefore \sum_{k=1}^{K} \frac{\sum_{i=1}^{N} \gamma_{ik}}{-\lambda} &= 1 \\
\therefore \lambda \sum_{k=1}^{K}\sum_{i=1}^{N} \gamma_{ik} &= 1 \\
\therefore \lambda &= -N \\
\pi_k &= \frac{\sum_{i=1}^{N} \gamma_{ik}}{N}
\end{aligned}
$$

Next, we calculate b_k and $\mathbf{w}_k$ by combining them into a vector $\tilde{\mathbf{w}}_k = (\mathbf{w}_k, b_k)$ and defining $\tilde{\mathbf{x}}_i = (\mathbf{x}_i, 1)$. Then:

$$
\begin{aligned}
\frac{\delta L}{\delta \tilde{\mathbf{w}}_k} &= \sum_{i=1}^{N} \gamma_{ik} \frac{y_i - \tilde{\mathbf{w}}_k^T \tilde{\mathbf{x}}_i}{\sigma^2} \tilde{\mathbf{x}}_i = 0 \\
\therefore \sum_{i=1}^{N} \gamma_{ik} y_i \tilde{\mathbf{x}}_i &= \sum_{i=1}^{N} \gamma_{ik} \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T \tilde{\mathbf{w}}_k \\
\therefore \tilde{\mathbf{w}}_k &= (\sum_{i=1}^{N} \gamma_{ik} \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T)^{-1} \sum_{i=1}^{N} \gamma_{ik} y_i \tilde{\mathbf{x}}_i
\end{aligned}
$$

Finally, we calculate σ_k:

$$
\begin{aligned}
\frac{\delta L}{\delta \sigma_k^2} &= \sum_{i=1}^{N} \gamma_{ik} (\frac{(y_i - \mathbf{w}_k^T \mathbf{x}_i - b_k)^2}{2\sigma^4} - \frac{1}{2\sigma_k^2}) \\
&= \frac{1}{2\sigma_k^4} \sum_{i=1}^{N} \gamma_{ik} ((y_i - \mathbf{w}_k^T \mathbf{x}_i - b_k)^2 - \sigma_k^2) = 0 \\
\therefore \sigma_k^2 &= \frac{\sum_{i=1}^{N} \gamma_{ik} (y_i - \mathbf{w}_k^T \mathbf{x}_i - b_k)^2}{\sum_{i=1}^{N} \gamma_{ik}}
\end{aligned}
$$

(e) Now let's put these ideas into practice. Generate the data as follows (or see `hw4p5.py`):

- Use $d = 1$ and $N = 500$. Let $\underline{x}$ be sampled independently and uniformly from the interval $[0, 1]$.
- Use $\pi_1 = 0.7, \pi_2 = 0.3$.
- Use $w_1 = -2, w_2 = 1$.
- Use $b_1 = 0.5, b_2 = -0.5$.
- Use $\sigma_1 = 0.4, \sigma_2 = 0.3$. *Note: This is σ, not σ^2.*
- Draw $\underline{y}$ from the distribution using the above parameters and your already sampled $\underline{\mathbf{x}}$.

Implement the EM algorithm using the updates you derived in **(d)**, and estimate the model parameters, initializing your estimates with the following values:

- Use $\hat{\pi}_1 = \hat{\pi}_2 = 0.5$.
- Use $\hat{w}_1 = 1, \hat{w}_2 = -1$.
- Use $\hat{b}_1 = \hat{b}_2 = 0$.
- Use $\hat{\sigma}_1 = \hat{\sigma}_2 = \text{std}(\underline{y})$. This is the standard deviation of your generated $\underline{y}$.

Deliverables:

- A plot of the log-likelihood as a function of iteration number. Terminate your algorithm when the log-likelihood increases by less than 10^{-4}.
- The estimated model parameters.
- A plot showing the data and estimated lines together.
- Please submit you code, as usual.

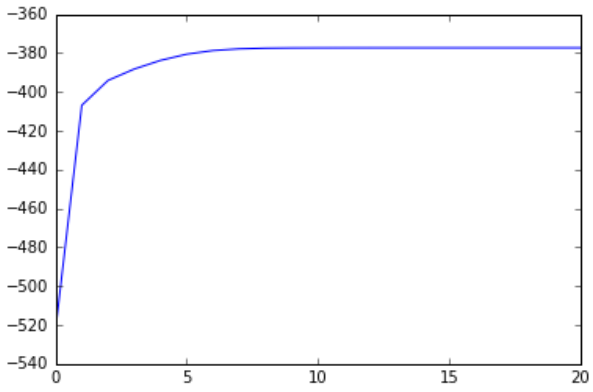


Figure 5: Log-likelihood vs. iteration

See Figure 5 for the plot of the log-likelihood.

Estimated parameters should be pretty close to the true parameters (see above), but it's not uncommon for them to be off by as much as 0.1. It is ok if the parameter order is reversed from the true parameters.

See Figure 6 for a plot of the data with lines.

See `hw4p5_sol.py` for a code solution.

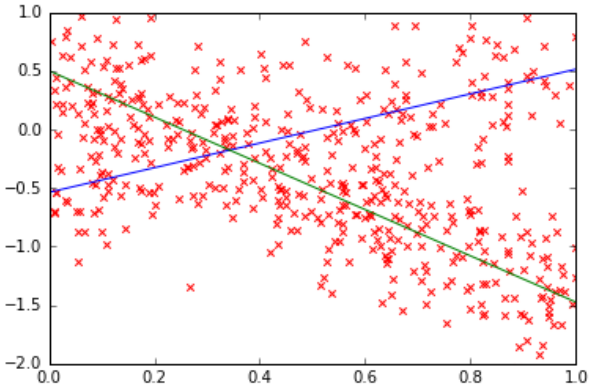


Figure 6: Data with learned lines