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/).

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.

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

(c) Suppose we observe N samples $\mathcal{D} = (x_1, \dots, x_N)$ 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 $\mathcal{D}$.

(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.*

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 $\boldsymbol{p} = (p_1, \dots, p_m) \in \Delta^{m-1}$ governed by nonnegative concentration parameters $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_m)$ 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.

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

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

(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.)

(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,

$$H = \nabla^2_{\boldsymbol{\alpha}} F(\boldsymbol{\alpha}) = Q + c11^T$$

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

(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).

(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.

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!).

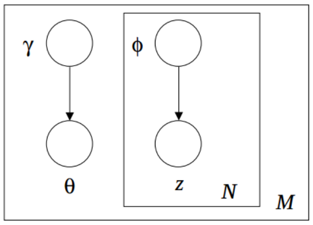
(i) $P(y_1, y_2, y_3, y_4, y_5) = P(y_1)P(y_2|y_1)\prod_{k=3}^{5} P(y_k|y_{k-1}, y_{k-2})$

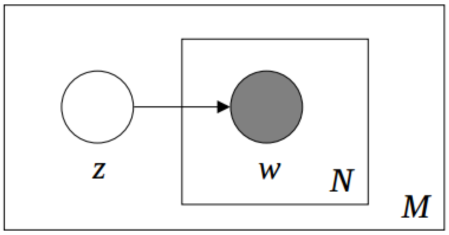
(ii) $P(x_1, \ldots, x_N, y_1, \ldots, y_N) = P(y_1)\prod_{k=2}^{N} P(y_k|y_{k-1})\prod_{k=1}^{N} P(x_k|y_k)$

(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:

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

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





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.

(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.*

(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 $\underline{\mathbf{x}}$.

(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}.*

(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 $\underline{z}$.

(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).

(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.