EECS 545 – Machine Learning - Homework #5

Daniel LeJeune & Benjamin Bray

Due: 11:00pm 04/04/2016

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/).

Student contributors: Dominic Calabrese, Qi Luo, Amlan Nayak

1) **Forwards vs. Reverse KL Divergence, (20 pts).**

Consider a factored approximation $q(x, y) = q_1(x)q_2(y)$ to a joint distribution $p(x, y)$.

(a) Show that to minimize the forwards divergence $D_{KL}(p||q)$, we should set $q_1(x) = p(x)$ and $q_2(y) = p(y)$, that is, the optimal approximation is a product of marginals.

First, notice that

$$\begin{aligned}
D_{KL}(p\|q) &= \min -\int\int p(x,y)\log(\frac{q_1(x)q_2(y)}{p(x,y)})dxdy \\
&= -\int\int p(x,y)\log(\frac{q_1(x)q_2(y)}{p(x,y)})dxdy \\
&= -\int \log q_1(x)\int p(x,y)dydx - \int \log q_2(y)\int p(x,y)dxdy + \int\int p(x,y)\log p(x,y)dxdy \\
&= -\int \log q_1(x)p(x)dx - \int \log q_2(y)p(y)dy + \int\int p(x,y)\log p(x,y)dxdy \\
&= H(p(x), q_1(x)) + H(p(y), q_2(y)) - H(p(x,y)) \\
&= H(p(x)) + D_{KL}(p(x)\|q_1(x)) + H(p(y)) + D_{KL}(p(y)\|q_2(y)) - H(p(x,y))
\end{aligned}$$

The only terms that depend on q now are $D_{KL}(p(x)\|q_1(x))$ and $D_{KL}(p(y)\|q_2(y))$, which are uniquely minimized by $q_1(x) = p(x)$ and $q_2(y) = p(y)$, respectively.

(b) Now consider the joint distribution shown in the table. Show that the reverse divergence $D_{KL}(q||p)$ has three distinct minima. Identify those minima and evaluate $D_{KL}(q||p)$ at each.

Notice that

$$D_{KL}(q\|p) = -H(q(x,y)) + \sum_{x,y} q_1(x)q_2(y)\log\frac{1}{p(x,y)}$$

In this problem, the first term is always finite, and each term in the double sum is always non-negative. However, if one of the terms has $q_1(x)q_2(y) \neq 0$ when $p(x,y) = 0$, then our divergence will be infinite. As a

	x_1	x_2	x_3	x_4
y_1	1/8	1/8		
y_2	1/8	1/8		
y_3			1/4	
y_4				1/4

Table 1: Joint probability table for $p(x, y)$.

result, any distribution q that will give finite reverse divergence must have values of 0 when $p(x, y) = 0$, so we can limit our search for minima to q that obeys this property.

We don't want to put any constraints on q that aren't necessary, so we limit our 0-forcing strategy to *minimal* 0-forcing schemes – that is, we should not be able to remove a 0 constraint on q_1 or q_2 and still have all of the necessary 0 constraints. There are three such schemes, and we find a local minimum within each scheme:

- Force $q_1(x_3) = q_1(x_4) = q_2(y_3) = q_2(y_4) = 0$. In order to minimize $D_{KL}(q||p)$ in this domain, we take the gradient for the remaining parameters and set it equal to 0. For notational convenience we make the following substitutions :

  $$a = q_1(x_1), \quad b = q_1(x_2), \quad c = q_2(y_1), \quad d = q_2(y_2)$$

  Note that $p(x, y) = 1/8$ for all pairs (x, y) in S. Further since q_1 and q_2 are probability mass functions, we must have $a + b = 1$ and $c + d = 1$. With these substitutions, we find:

  $$\begin{aligned} D_{KL}(q||p) &= \sum_{x,y} q_1(x)q_2(y) \log\left(\frac{q_1(x)q_2(y)}{p(x,y)}\right) \\ &= ac\log(8ac) + ad\log(8ad) + bc\log(8bc) + bd\log(8bd) \\ &= \log 8 + ac\log(ac) + ad\log(ad) + bc\log(bc) + bd\log(bd) \\ &= \log 8 + a\log(a)(c+d) + c\log(c)(a+b) + d\log(d)(b+a) + b\log(b)(c+d) \\ &= \log 8 + a\log(a) + c\log(c) + d\log(d) + b\log(b) \\ &= \log 8 + a\log(a) + c\log(c) + (1-c)\log(1-c) + (1-a)\log(1-a) \\ \Rightarrow \frac{\partial D_{KL}}{\partial a} &= \log(a) + 1 - \log(1-a) - 1 = 0 \Rightarrow \log(a) = \log(1-a) \Rightarrow a = \frac{1}{2} \\ \rightarrow a + b &= 1 \Rightarrow b = \frac{1}{2} \\ \frac{\partial D_{KL}}{\partial c} &= \log(c) + 1 - \log(1-c) - 1 = 0 \Rightarrow \log(c) = \log(1-c) \Rightarrow c = \frac{1}{2} \\ \rightarrow c + d &= 1 \Rightarrow d = \frac{1}{2} \end{aligned}$$

  Using the values obtained above, we get:

  $$D_{KL}(q||p) = \sum_{x,y} q_1(x)q_2(y) \log\frac{q_1(x)q_2(y)}{p(x,y)} = 4 \cdot \left[\frac{1}{4}\log\left(\frac{1/4}{1/8}\right)\right] = \log 2$$

- Force $q_1(x_1) = q_1(x_2) = q_1(x_4) = q_2(y_1) = q_2(y_2) = q_2(y_4) = 0$. In this case, we have no choice but to pick $q_1(x_3) = 1$ and $q_2(y_3) = 1$. Then we see that:

  $$D_{KL}(q||p) = \sum_{x,y} q_1(x)q_2(y) \log\frac{q_1(x)q_2(y)}{p(x,y)} = (1)\log\left(\frac{1}{1/4}\right) = \log 4$$

- Force $q_1(x_1) = q_1(x_2) = q_1(x_3) = q_2(y_1) = q_2(y_2) = q_2(y_3) = 0$. Similar to the previous case, we find that $q_1(x_4) = 1$ and $q_2(y_4) = 1$. Thus, we obtain:

$$D_{KL}(q||p) = \sum_{x,y} q_1(x)q_2(y) \log \frac{q_1(x)q_2(y)}{p(x,y)} = (1)\log\left(\frac{1}{1/4}\right) = \log 4$$

(c) What is the value of $D_{KL}(q||p)$ if we set $q(x,y) = p(x)p(y)$ using the joint distribution in Table 1?

$$\begin{aligned} D_{KL}(q\|p) &= -\sum_x \sum_y p(x)p(y) \log \frac{p(x,y)}{p(x)p(y)} \\ &= -4\frac{1}{4}\frac{1}{4}\log\frac{1/8}{1/4\times 1/4} - 2\frac{1}{4}\frac{1}{4}\log\frac{1/4}{1/4\times 1/4} - 6\log 0 \\ &= -\frac{1}{2}\log 2 - 6\log 0 \\ &= \infty \end{aligned}$$

2) **Gibbs Sampling from a 2D Gaussian, (20 pts).**

Suppose $X \sim \mathcal{N}(\mu, \Sigma)$, where

$$\mu = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 1 & 1/2 \\ 1/2 & 1 \end{bmatrix}$$

(a) Derive the full conditionals $p(x_1|x_2)$ and $p(x_2|x_1)$. You should not use already existing results on conditional probabilities for jointly Gaussian random variables.

By definition of multivariate Gaussian distributions, $x_2 \sim \mathcal{N}(1,1)$. Also note that

$$|\Sigma| = \frac{3}{4}, \quad \Sigma^{-1} = \frac{4}{3}\begin{bmatrix} 1 & -\frac{1}{2} \\ -\frac{1}{2} & 1 \end{bmatrix}$$

Then

$$\begin{aligned} p(x_1|x_2) &= \frac{p(x_1,x_2)}{p(x_2)} \\ &= \frac{(2\pi\sqrt{3/4})^{-1}\exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}))}{(\sqrt{2\pi})^{-1}\exp(-\frac{1}{2}(x_2-1)^2)} \\ &= \frac{1}{\sqrt{2\pi(\frac{3}{4})}}\exp\left(-\frac{1}{2}\left[\frac{4}{3}(x_1^2 - x_1x_2 - x_1 - x_2 + x_2^2 + 1) - x_2^2 + 2x_2 - 1\right]\right) \\ &= \frac{1}{\sqrt{2\pi(\frac{3}{4})}}\exp\left(-\frac{(x_1 - \frac{x_2+1}{2})^2}{2(\frac{3}{4})}\right) \\ &\Rightarrow p(x_1|x_2) \sim \mathcal{N}\left(\frac{x_2+1}{2}, \frac{3}{4}\right) \end{aligned}$$

By symmetry, $p(x_2|x_1) \sim \mathcal{N}\left(\frac{x_1+1}{2}, \frac{3}{4}\right)$

(b) Implement a Gibbs sampling algorithm for estimating $p(x_1, x_2)$ using the conditionals determined in part **(a)**.

Implementation details:

- Start with $x_1 = 0$, then sample x_2 conditioned on the current value of x_1. Then sample x_1 conditioned on the current value of x_2, and so on.
- Store the values of x_1 and x_2 as you go, because you will plot histograms later.
- Sample points in this manner until you have 5000 points for each x_1 and x_2.

Deliverables:

- Plots of the one-dimensional marginals $p(x_1)$ and $p(x_2)$ as histograms. Superimpose a plot of the exact (true) marginals on each.
- Please submit your code, as usual.

See Figure 1 for plots. See `hw5p2_sol.py` for code.

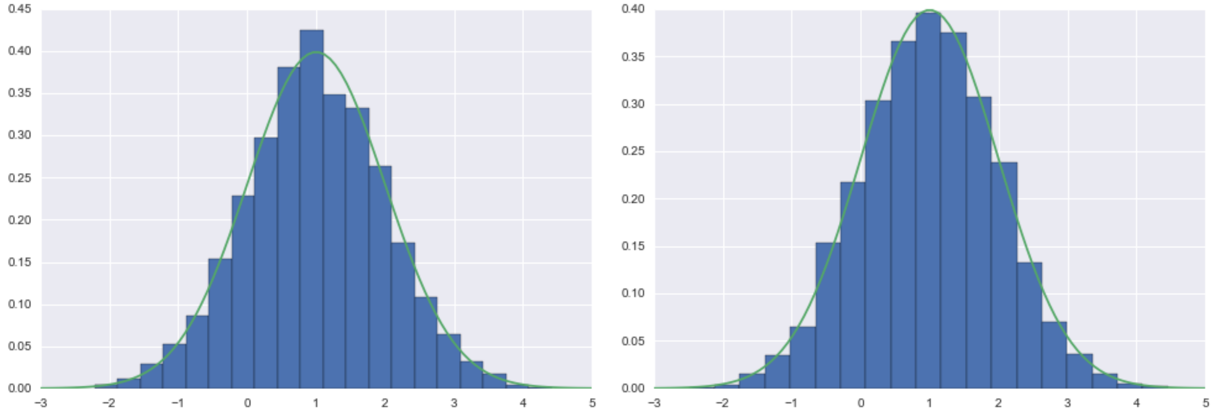

Figure 1: *Left*: histogram of x_1 with true marginal overlaid. *Right*: histogram of x_2 with true marginal overlaid.

3) **Hidden Markov Models, (20 pts).**

Consider the following HMM with 3 states (0,1,2) and binary observations (0,1):

$$A = \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.2 & 0.4 & 0.4 \\ 0.4 & 0.1 & 0.5 \end{bmatrix} \qquad \phi = \begin{bmatrix} 0.8 & 0.2 \\ 0.1 & 0.9 \\ 0.5 & 0.5 \end{bmatrix} \qquad \pi_0 = \begin{bmatrix} 0.5 \\ 0.3 \\ 0.2 \end{bmatrix}$$

A is the transition matrix, where A_{ij} (using 0-based indexing) is the probability of transition from state i to state j. ϕ is the observation matrix consisting of emission probabilities, i.e. ϕ_{ij} is the probability of seeing observation j at state i. π_0 specifies the initial distribution over states.

(a) Given the sequence of observations 0101, compute the posterior distribution over the sequence of states and report the 3 most probable sequences. Fill out the table below. In the table, the prior probability means the unconditional probability of the state sequence in the row. The likelihood probability means the conditional probability of the observation sequence given the state sequence.

The posterior probability means the conditional probability of the state sequence in the row given the sequence of observations. We suggest writing code for this, rather than evaluating all of these things by hand, and please turn in your code.

Most Probable State Sequences	Prior Probability	Likelihood	Posterior Probability
0222	0.0375	0.1	0.073522
0122	0.02	0.18	0.070581
0201	0.012	0.288	0.067758

(b) Sample 5000 observation sequences of length 4 from the HMM (or see `hw5p3.py`). Then, treat the first N sequences as training data and learn the HMM parameters by the Baum–Welch algorithm (Please refer to the exercise 13.12 in Bishop's book for the E step and M step details).

Implementation details:

- Run the experiment for $N = 500, 1000, 2000, 5000$.
- Initialize your parameter estimates by sampling uniformly from $[0, 1]$ and then scaling them so that your distributions meet the summation constraints.
- Run EM for 50 iterations and after each iteration, compute the unconditional distributions over all possible observation sequences of length 4 given by the current parameters and compare to the distribution given by the true parameters.
- Use the same initial values of parameters for different N values in the EM algorithm.

Deliverables:

- A plot of the distance defined below between the distributions as a function of the number of iterations [1]. Draw the curves for $N = 500, 1000, 2000, 5000$ on the same figure.
- Please submit your code, as usual.

See Figure 2 for an example plot of the distance. Plots can vary due to random sampling and initialization. See `hw5p3_sol.py` for code.

4) **Kaggle Challenge, (20 pts).**

You can use any algorithm and design any features to remove the noise from handwritten digits from the MNIST dataset. Please refer to `https://inclass.kaggle.com/c/handwritten-digit-denoising` for details. This problem will be graded separately based on your performance on the public leaderboard.

See `hw5p4_sol.py` for an example solution that gives around 29 RMSE.

5) **Independent Component Analysis, (20 pts).**

Consider the scenario where we observe a random vector $\mathbf{x} \in \mathbb{R}^d$ generated according to

$$\mathbf{x} = A\boldsymbol{s}$$

[1] Given two distributions μ, μ' over a finite set X, the distance is defined as

$$\delta(\mu, \mu') = \frac{1}{2} \sum_{x \in X} |\mu(x) - \mu'(x)|$$

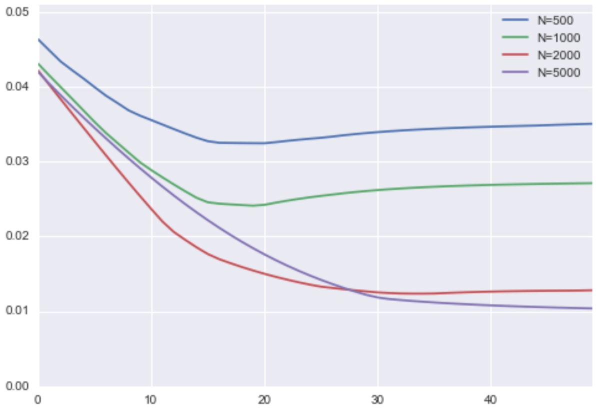

Figure 2: Distance vs iteration for different numbers of observed sequences.

where $A \in \mathbb{R}^{d \times d}$ is unknown, and $\boldsymbol{s} \in \mathbb{R}^d$ is a vector of *independent* random variables, each of unknown distribution. Given *iid* observances $(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)})$, we would like to recover $(\boldsymbol{s}^{(1)}, \dots, \boldsymbol{s}^{(N)})$. Collect the observations $\mathbf{x}^{(i)}$ into a matrix $X \in \mathbb{R}^{d \times N}$, and collect $\boldsymbol{s}^{(i)}$ into a matrix $S \in \mathbb{R}^{d \times N}$. Now our model is

$$X = AS$$

Our goal in this problem is to obtain a matrix $Y \in \mathbb{R}^{d \times N}$ satisfying

$$Y = WX$$

for some $W \in \mathbb{R}^{d \times d}$, such that, if viewed as a collection of *iid* observations $(\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(N)})$ of a random variable $\mathbf{y}$, maximizes the pairwise independence of the elements of $\mathbf{y}$. Our hope is that the recovered Y is similar to S; however, because A is unknown, it is impossible to recover the scale or permutations of the rows of S. We will consider all random variables to have zero mean in this problem. This problem is known as independent component analysis (ICA).

(a) If the elements of $\mathbf{y}$ are independent, then they are uncorrelated, so we can restrict our solution to those with diagonal covariance. Further, since we can't recover the scale of the rows of S, we can restrict each row of Y to have unit variance. Combining these two, we restrict our set of possible solutions to the set of matrices with identity covariance; i.e.,

$$\frac{1}{N} Y Y^T = I$$

Data with this property is said to be *white*. If X is also white, then any solution to $Y = WX$ must have W as an orthogonal matrix, which reduces the search space of our problem significanly. To that end, we define $\tilde{X} = DX$ for some $D \in \mathbb{R}^{d \times d}$, such that $\tilde{X}$ is white, and then solve the new version of the problem, where $Y = W\tilde{X}$.

Provide such a whitening transformation matrix D that will whiten X. You must be able to construct D using X.

We must have

$$\frac{1}{N} D X X^T D^T = I$$

or, defining the covariance matrix $C_X = \frac{1}{N} X X^T$, we need $D C_X D^T = I$. C_X is positive semi-definite, so it has eigenvalue decomposition $U \Lambda U^T = U \Sigma \Sigma^T U^T$, where $U \Sigma V^T$ is the SVD of X. Then we see that any D

of the form

$$D = Q\tilde{\Sigma}^{-1}U^T$$

is a whitening transformation, where Q is an orthogonal matrix, and $\tilde{\Sigma}$ is a square diagonal matrix with the singular values of X. To show this:

$$\begin{aligned} DC_X D^T &= Q\tilde{\Sigma}^{-1}U^T U\Sigma\Sigma^T U^T U\tilde{\Sigma}^{-1}Q^T \\ &= Q\tilde{\Sigma}^{-1}\Sigma\Sigma^T\tilde{\Sigma}^{-1}Q^T \\ &= QQ^T \\ &= I \end{aligned}$$

Two common examples of whitening matrices that fit this form are $D = C_X^{-1/2}$ and $D = \tilde{\Sigma}^{-1}U^T$.

(b) There are several ways to quantify how independent the elements of $\mathbf{y}$ are (negentropy and mutual information, for example), but it turns out that they are all equivalent to measuring the non-Gaussianity of $\mathbf{y}$.[2] So, to maximize independence between the elements of $\mathbf{y}$, we can maximize the non-Gaussianity of the elements of $\mathbf{y}$. We will use as our measure of non-Gaussianity the following function:

$$J(y) = (\mathbb{E}[G(y)] - \gamma)^2$$

where $\gamma \triangleq \mathbb{E}[G(\nu)]$ for a Gaussian random variable ν with zero mean and unit variance, and $G(y)$ is a well-chosen [non-quadratic] function. Letting $\mathbf{w}_k$ denote the k^{th} row of W, our total non-Gaussianity is

$$J(\mathbf{y}) = \sum_{k=1}^{d}(\mathbb{E}[G(y_k)] - \gamma)^2 = \sum_{k=1}^{d}(\mathbb{E}[G(\mathbf{w}_k^T\mathbf{x})] - \gamma)^2$$

Let's try this out. Generate data as follows (or see `hw5p5.py`):

- Use $d = 2$, $N = 10,000$.
- Let $s_1^{(i)} = \sin(i/200)$ (a sinusoidal wave).
- Let $s_2^{(i)} = \text{remainder}(i/200, 2) - 1$ (a sawtooth wave).
- Let $A = \begin{bmatrix} 1 & 2 \\ -2 & 1 \end{bmatrix}$
- Compute $X = AS$.

Obtain the matrix Y with maximal independence using the total measure of non-Gaussianity above. Do this by finding the orthogonal matrix W that achieves the maximum. Since W is a 2×2 orthogonal matrix, let

$$W(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Implementation details:

- Whiten X using your whitening matrix D from part **(a)** first.
- Use $G(y) = \log\cosh y$.
- Estimate the values of all expectations using empirical means.
- Estimate γ with the empirical mean of $G(\cdot)$ applied to 10^6 random standard normal values.

[2] For more details, see Hyvärinen, a., & Oja, E. (2000). Independent component analysis: Algorithms and applications. Neural Networks, 13(4-5), 411-430. http://doi.org/10.1016/S0893-6080(00)00026-5

- Use a grid search on $\theta \in [0, \pi/2]$ to select the optimal θ.

Deliverables:

- A plot of your estimate of $J(\mathbf{y})$ versus θ for $\theta \in [0, \pi/2]$.
- A plot of each row of the recovered Y, preferably in the same plot but not overlapping.
- Please submit your code, as usual.

See Figure 3 for a plot of the objective and the recovered components. These plots were generated using $D = C_X^{-1/2}$. Other whitening matrices will recover the same signals (maybe out of order or with signs flipped), but the objective will be shifted left or right. See `hw5p5_sol.py` for code.

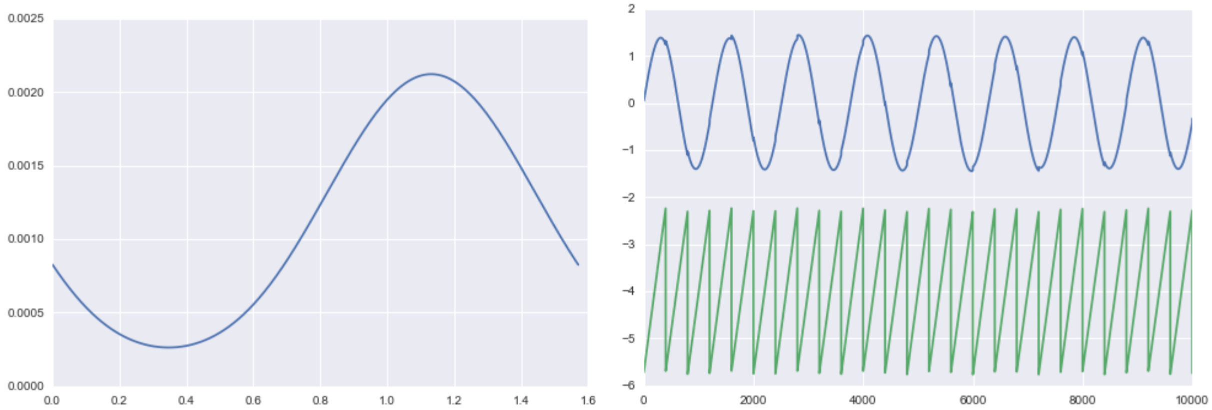

Figure 3: *Left:* ICA objective function. *Right:* recovered components.