## Summative Assignments

### Probabilistic and Unsupervised Learning

Peter Orbanz

Some questions are marked as BONUS QUESTIONS. Attempt the non-bonus questions before answering these.

1. **[25 marks] Models for binary vectors**. Consider a data set of binary (black and white) images. Each image is arranged into a vector of pixels by concatenating the columns of pixels in the image. The data set has $N$ images $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$ and each image has $D$ pixels, where $D$ is (number of rows $\times$ number of columns) in the image. For example, image $\mathbf{x}^{(n)}$ is a vector $(x_1^{(n)}, \dots, x_D^{(n)})$ where $x_d^{(n)} \in \{0, 1\}$ for all $n \in \{1, \dots, N\}$ and $d \in \{1, \dots, D\}$.

   (a) Explain why a multivariate Gaussian would not be an appropriate model for this data set of images. [5 marks]

   Assume that the images were modelled as independently and identically distributed samples from a D-dimensional **multivariate Bernoulli distribution** with parameter vector $\mathbf{p} = (p_1, \dots, p_D)$, which has the form

$$P(\mathbf{x}|\mathbf{p}) = \prod_{d=1}^{D} p_d^{x_d}(1 - p_d)^{(1-x_d)}$$

   where both $\mathbf{x}$ and $\mathbf{p}$ are $D$-dimensional vectors

   (b) What is the equation for the maximum likelihood (ML) estimate of $\mathbf{p}$? Note that you can solve for $\mathbf{p}$ directly. [5 marks]

   (c) Assuming independent Beta priors on the parameters $p_d$

$$P(p_d) = \frac{1}{B(\alpha, \beta)} p_d^{\alpha-1}(1 - p_d)^{\beta-1}$$

   and $P(\mathbf{p}) = \prod_d P(p_d)$. Fine the maximum a posteriori (MAP) estimator for $\mathbf{p}$. [5 marks]

   Download the data set `binarydigits.txt` from the course website, which contains $N = 100$ images with $D = 64$ pixels each, in an $N \times D$ matrix. These pixels can be displayed as $8 \times 8$ images by rearranging them. View them in Matlab by running `bindigit.m` or in Python by running `bindigit.py`.

   (d) Write code to learn the ML parameters of a multivariate Bernoulli from this data set and display these parameters as an $8 \times 8$ image. Include your code with your submission, and a visualisation of the learned parameter vector as an image. (You may use Matlab, Octave or Python) [5 marks]

   (e) Modify your code to learn MAP parameters with $\alpha = \beta = 3$. Show the new learned parameter vector for this data set as an image. Explain why this might be better or worse than the ML estimate. [5 marks]

2. **[15 marks] Model selection**. In the binary data model above, find the expressions needed to calculate the (relative) probability of the following three different models:

   (a) all $D$ components are generated from a Bernoulli distribution with $p_d = 0.5$

   (b) all $D$ components are generated from Bernoulli distributions with unknown, but identical, $p_d$

   (c) each component is Bernoulli distributed with separate, unknown $p_d$

Assume that all three models are equally likely *a priori*, and take the prior distributions for any unknown probablities to be uniform. Calculate the posterior probabilities of each of the three models having generated the data in `binarydigits.txt`.

3. **[65 marks + 5 BONUS] EM for Binary Data**.

Consider the data set of binary (black and white) images used in the previous question.

(a) Write down the likelihood for a model consisting of a mixture of $K$ multivariate Bernoulli distributions. Use the parameters $\pi_1, \ldots, \pi_K$ to denote the mixing proportions ($0 \leq \pi_k \leq 1; \sum_k \pi_k = 1$) and arrange the $K$ Bernoulli parameter vectors into a matrix $\mathsf{P}$ with elements $p_{kd}$ denoting the probability that pixel $d$ takes value 1 under mixture component $k$. Assume the images are iid under the model, and that the pixels are independent of each other within each component distribution. [5 marks]

Just as we can for a mixture of Gaussians, we can formulate this mixture as a latent variable model, introducing a discrete hidden variable $s^{(n)} \in \{1, \ldots, K\}$ where $P(s^{(n)} = k|\boldsymbol{\pi}) = \pi_k$.

(b) Write down the expression for the responsibility of mixture component $k$ for data vector $\mathbf{x}^{(n)}$, i.e. $r_{nk} \equiv P(s^{(n)} = k|\mathbf{x}^{(n)}, \boldsymbol{\pi}, \mathsf{P})$. This computation provides the E-step for an EM algorithm. [5 marks]

(c) Find the maximizing parameters for the expected log-joint

$$\underset{\boldsymbol{\pi},\mathsf{P}}{\operatorname{argmax}} \left\langle \sum_n \log P(\mathbf{x}^{(n)}, s^{(n)}|\boldsymbol{\pi}, \mathsf{P}) \right\rangle_{q(\{s^{(n)}\})}$$

thus obtaining an iterative update for the parameters $\boldsymbol{\pi}$ and $\mathsf{P}$ in the M-step of EM. [10 marks]

(d) Implement the EM algorithm for a mixture of $K$ multivariate Bernoullis.

Your code should take as input the number $K$, a matrix $X$ containing the data set, and a maximum number of iterations to run. The algorithm should terminate after that number of iterations, or earlier if the log likelihood converges (does not increase by more than a very small amount).

Hand in clearly commented code.

Run your algorithm on the data set for values of $K$ in $\{2, 3, 4, 7, 10\}$. Plot the log likelihood as a function of the iteration number, and display the parameters found.

[30 marks]

[Hints: Although the implementation may seem simple enough given the equations, there are many numerical pitfalls (that are common in much of probabilistic learning). A few suggestions:

- Likelihoods can be very small; it is often better to work with log-likelihoods.
- You may still encounter numerical issues computing responsibilities. Consider scaling the numerator and denominator of the equation by a suitable constant while still in the log domain.
- It may also help to introduce (weak) priors on $\mathsf{P}$ and $\boldsymbol{\pi}$ and use EM to find the MAP estimates, rather than ML. State clearly whether you use this approach; if you do, specify the prior chosen, and report the log posterior instead of the log likelihood.

(e) Run the algorithm a few times starting from randomly chosen initial conditions. Do you obtain the same solutions (up to permutation)? Does this depend on K? Show the learned probability vectors as images.

Comment on how well the algorithm works, whether it finds good clusters (look at the cluster means and responsibilities and try to interpret them), and how you might improve the model. [10 marks]

(f) [BONUS] Express the log-likelihoods obtained in bits and relate these numbers to the length of the naive encoding of these binary data. How does your number compare to gzip (or another compression algorithm)? Why the difference? [5 marks]

(g) [BONUS] Consider the *total* cost of encoding both the model parameters and the data given the model. How does this total cost compare to gzip (or similar)? How does it depend on $K$? What might this tell you? [5 marks]

4. **[BONUS: 35 marks] LGSSMs, EM and SSID**.

Download the datafiles `ssm_spins.txt` and `ssm_spins_test.txt`. Both have been generated by an LGSSM:

$$\mathbf{y}_t \sim \mathcal{N}(A\mathbf{y}_{t-1}, Q) \quad [t = 2 \dots T] \qquad \mathbf{y}_1 \sim \mathcal{N}(0, I)$$
$$\mathbf{x}_t \sim \mathcal{N}(C\mathbf{y}_t, R) \quad [t = 1 \dots T]$$

using the parameters:

$$A = 0.99 \begin{pmatrix} \cos(\frac{2\pi}{180}) & -\sin(\frac{2\pi}{180}) & 0 & 0 \\ \sin(\frac{2\pi}{180}) & \cos(\frac{2\pi}{180}) & 0 & 0 \\ 0 & 0 & \cos(\frac{2\pi}{90}) & -\sin(\frac{2\pi}{90}) \\ 0 & 0 & \sin(\frac{2\pi}{90}) & \cos(\frac{2\pi}{90}) \end{pmatrix} \qquad Q = I - AA^{\mathsf{T}}$$

$$C = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0.5 & 0.5 & 0.5 & 0.5 \end{pmatrix} \qquad R = I$$

but different random seeds. We shall use the first as a training data set and the second as a test set.

(a) Run the function `ssm_kalman.m` or `ssm_kalman.py` that we have provided (or a re-implementation in your favourite language if you prefer) on the training data. Warning (for MATLAB version): the function expects data vectors in columns; you will need to transpose the loaded matrices!

Make the following plots (or equivelent):

```
logdet = @(A)(2*sum(log(diag(chol(A)))));
[Y,V,~,L] = ssm_kalman(X',Y0,Q0,A,Q,C,R, 'filt');
plot(Y'):
plot(cellfun(logdet,V));

[Y,V,Vj,L] = ssm_kalman(X',Y0,Q0,A,Q,C,R, 'smooth');
plot(Y'):
plot(cellfun(logdet,V));
```

Explain the behaviour of `Y` and `V` in both cases (and the differences between them). [5 marks]

(b) Write a function to learn the parameters $A$, $Q$, $C$ and $R$ using EM (we will assume that the distribution on the first state is known *a priori*). The M-step update equations for $A$ and $C$ were derived in lecture. You should show that the updates for $R$ and $Q$ can be written:

$$R_{\text{new}} = \frac{1}{T}\Big[\sum_{t=1}^{T} \mathbf{x}_t\mathbf{x}_t^{\mathsf{T}} - \Big(\sum_{t=1}^{T} \mathbf{x}_t\langle\mathbf{y}_t^{\mathsf{T}}\rangle\Big)C_{\text{new}}^{\mathsf{T}}\Big]$$

$$Q_{\text{new}} = \frac{1}{T-1}\Big[\sum_{t=2}^{T} \langle\mathbf{y}_t\mathbf{y}_t^{\mathsf{T}}\rangle - \Big(\sum_{t=2}^{T} \langle\mathbf{y}_t\mathbf{y}_{t-1}^{\mathsf{T}}\rangle\Big)A_{\text{new}}^{\mathsf{T}}\Big]$$

where $C_{\text{new}}$ and $A_{\text{new}}$ are as in the lecture notes. Store the log-likelihood at every step (easy to compute from the fourth value returned by `ssm_kalman`) and check that it increases.

[Hint: the matlab code

```
cellsum=@(C)(sum(cat(3,C{:}),3))
```

defines an inline function `cellsum()` that sums the entries of a cell array of matrices.]

Run at least 50 iterations of EM starting from a number of different initial conditions: (1) the generating parameters above (why does EM not terminate immediately?) and (2) 10 random choices.

Show how the likelihood increases over the EM iterations (hand in a plot showing likelihood vs iteration for each run, plotted in the same set of axes). Explain the features of this plot that you think are salient.

[If your code and/or computer is fast enough, try running more EM iterations. What happens?]

[25 marks]

(c) Evaluate the likelihood of the test data under the true parameters, and all of the parameters found above (EM initialised at the true parameters, random parameters and the SSID parameters, as well as the SSID parameters without EM). Show these numbers on or next to the training data likelihoods plotted above. Comment on the results.

[5 marks]

5. **[70 points] Decrypting Messages with MCMC**. You are given a passage of English text that has been encrypted by remapping each symbol to a (usually) different one. For example,

$$\begin{aligned} a &\to s \\ b &\to \text{!} \\ \langle\text{space}\rangle &\to v \\ &\vdots \end{aligned}$$

Thus a text like 'a boy...' might be encrypted by 'sv!op...'. Assume that the mapping between symbols is one-to-one. The file **symbols.txt** gives the list of symbols, one per line (the second line is ⟨space⟩). The file **message.txt** gives the encrypted message.

Decoding the message by brute force is impossible, since there are 53 symbols and thus 53! possible permutations to try. Instead we will set up a Markov chain Monte Carlo sampler to find modes in the space of permutations.

We model English text, say $s_1 s_2 \cdots s_n$ where $s_i$ are symbols, as a Markov chain, so that each symbol is independent of the preceding text given only the symbol before:

$$p(s_1 s_2 \cdots s_n) = p(s_1) \prod_{i=2}^{n} p(s_i | s_{i-1})$$

(a) Learn the transition statistics of letters and punctuation in English: Download a large text [say the English translation of *War and Peace*, which you can find here[1]] from the web and estimate the transition probabilities $p(s_i = \alpha | s_{i-1} = \beta) \equiv \psi(\alpha, \beta)$, as well as the stationary distribution $\lim_{i\to\infty} p(s_i = \gamma) \equiv \phi(\gamma)$. Assume that the first letter of your text (and also that of the encrypted text provided) is itself sampled from the stationary distribution.

Give formulae for the ML estimates of these probabilities as functions of the counts of numbers of occurrences of symbols and pairs of symbols.

Compute the estimated probabilities. Report the values as a table. [6 marks]

(b) The state variable for our MCMC sampler will be the symbol permutation. Let $\sigma(s)$ be the symbol that stands for symbol $s$ in the encrypted text, e.g., $\sigma(a) = s$ and $\sigma(b) =$! above. Assume a uniform prior distribution over permutations.

Are the latent variables $\sigma(s)$ for different symbols $s$ independent?

Let $e_1 e_2 \cdots e_n$ be an encrypted English text. Write down the joint probability of $e_1 e_2 \cdots e_n$ given $\sigma$. [6 marks]

(c) We use a Metropolis-Hastings (MH) chain, with the proposal given by choosing two symbols $s$ and $s'$ at random and swapping the corresponding encrypted symbols $\sigma(s)$ and $\sigma(s')$.

How does the proposal probability $S(\sigma \to \sigma')$ depend on the permutations $\sigma$ and $\sigma'$? What is the MH acceptance probability for a given proposal? [10 marks]

(d) Implement the MH sampler, and run it on the provided encrypted text. Report the current decryption of the first 60 symbols after every 100 iterations. Your Markov chain should converge to give you a fairly sensible message. (Hint: it may help to initialize your chain intelligently and to try multiple times; in any case, please describe what you did). [30 marks]

(e) Note that some $\psi(\alpha, \beta)$ values may be zero. Does this affect the ergodicity of the chain? If the chain remains ergodic, give a proof; if not, explain and describe how you can restore ergodicity. [5 marks]

(f) Analyse this approach to decoding. For instance, would symbol probabilities alone (rather than transitions) be sufficient? If we used a second order Markov chain for English text, what problems might we encounter? Will it work if the encryption scheme allows two symbols to be mapped to the same encrypted value? Would it work for Chinese with $> 10000$ symbols? [13 marks]

[1] https://www.gutenberg.org/files/2600/2600-0.txt

6. **[BONUS 60 points] Implementing Gibbs sampling for LDA**. Take a look at the accompanying code, which sets up a framework in which you will implement both the standard and collapsed Gibbs sampling inference for LDA. Read the README which lays out the MATLAB variables used.

   (a) Implement both standard and collapsed Gibbs sampline updates, and the log joint probabilities in question 1(a), 1(c) above. The files you need to edit are stdgibbs_logjoint, stdgibbs_update, colgibbs_logjoint,colgibbs_update. Debug your code by running toyexample. Show sample plots produced by toyexample, and attach and document the MATLAB code that you wrote. [20 points]

   (b) Based upon the plots of log predictive and joint probabilities produced by toyexample, how many iterations do you think are required for burn-in? Discarding the burn-in iterations, compute and plot the autocorrelations of the log predictive and joint probabilities for both Gibbs samplers. You will need to run toyexample for a larger number of iterations to reduce the noise in the autocorrelation. Based upon the autocorrelations how many samples do you think will be need to have a representative set of samples from the posterior? Describe what you did and justify your answers with one or two sentences. [10 points]

   (c) Based on the computed autocorrelations, which of the two Gibbs samplers do you think converge faster, or do they converge at about the same rate? If they differ, why do you think this might be the case? Justify your answers. [5 points]

   (d) Try varying $\alpha$, $\beta$ and $K$. What effects do these have on the posterior and predictive performance of the model? Justify your answers. [5 points]

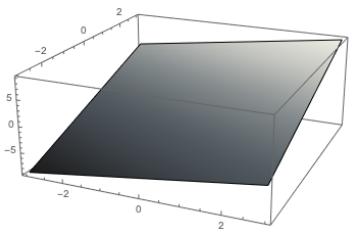
   **Topic modelling of NeurIPS papers**. Now that we have code for LDA, we can try our hands on finding the topics at a major machine learning conference (NeurIPS). In the provided code there is a file nips.data which contains preprocessed data. The vocabulary is given in nips.vocab.

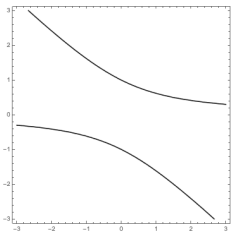
   (e) The data in nips.data is probably too big so that our MATLAB implementation will be too slow. We will try to reduce the data set to a more tractable size, by removing words from the vocabulary. Come up with a metric for how informative/relevant/topical a vocabulary word is. You may want to experiment and try multiple metrics, and make sure that keywords like "Bayesian", "graphical", "Gaussian", "support", "vector", "kernel", "representation", "regression", "classification" etc have high metric. Report on your experiences, and use your metric to prune the data set to just the top few hundred words (say 500, or lower if the implementation is still too slow). You may find it useful to read up on `tf-idf` on wikipedia. [10 points]

   (f) Now run LDA on the reduced NeurIPS data, using one of the Gibbs samplers you have just written. You will need to experiment with various settings of $\alpha$, $\beta$ and $K$ until the topics discovered looks "reasonable". Describe the topics you found. How do the topics change (qualitatively) as $\alpha$, $\beta$ and $K$ are varied? [10 points]

7. **[15 marks] Optimization**.

   (a) Find the local (!) extrema of the function $f(x, y) := x + 2y$ subject to the constraint $y^2 + xy = 1$. For illustration, here are plots of the function $f$ (left) and the set of points satisfying the constraints (right) on the square $[-3, 3]^2$:



Please derive your solution using a Lagrange multiplier, and denote this multiplier by $\lambda$. We are asking for the points at which the local extrema occur, not for the function values at these points. [9 marks]

(b) Suppose we have a numerical routine to evaluate the exponential function $\exp(x)$. How can we compute the function $\ln(a)$, for a given $a \in \mathbb{R}_+$, using Newton's method?

i. Derive a function $f(x, a)$ to which Newton's method can be applied to find $x$ such that $x = \ln(a)$.

ii. Specify the update equation $x_{n+1} = \ldots$ in Newton's algorithm for this problem.

[6 marks]

8. **[BONUS: 20 marks] Eigenvalues as solutions of an optimization problem**. Let $A$ be a symmetric $n \times n$-matrix, and define

$$q_A(x) := x^\top Ax \quad \text{and} \quad R_A(x) := \frac{x^\top Ax}{x^\top x} = \frac{q_A(x)}{\|x\|^2} \quad \text{for } x \in \mathbb{R}^n \, .$$

We have already encountered the quadratic form $q_A$ in class. The purpose of this problem is to verify the following fact:
*If $A$ is a symmetric $n \times n$-matrix, the optimization problem*

$$x^* := \operatorname*{argmax}_{x \in \mathbb{R}^n} R_A(x)$$

*has a solution, $R_A(x^*)$ is the largest eigenvalue of $A$, and $x^*$ is a corresponding eigenvector.*

This result is very useful in machine learning, where we are often interested in the largest eigenvalue specifically—it allows us to compute the largest eigenvalue without computing the entire spectrum, and it replaces an algebraic characterization (the eigenvalue equation) by an optimization problem. We will assume as known that the function $q_A$ is continuous.

(a) Use the extreme value theorem of calculus (recall: a continuous function on a compact domain attains its maximum and minimum) to show that $\sup_{x \in \mathbb{R}^n} R_A(x)$ is attained.
**Hint**: Since $\mathbb{R}^n$ is not compact, transform the supremum over $\mathbb{R}^n$ into an equivalent supremum over the unit sphere $S = \{x \in \mathbb{R}^n | \, \|x\| = 1\}$. The set $S$ is compact (which you can assume as known). [6 marks]

(b) Let $\lambda_1 \geq \ldots \geq \lambda_n$ be the eigenvalues of $A$ enumerated by decreasing size, and $\xi_1, \ldots, \xi_n$ corresponding eigenvectors that form an ONB. Recall from class that we can represent any vector $x \in \mathbb{R}^n$ as

$$x = \sum_{i=1}^{n} (\xi_i^\top x) \xi_i \, .$$

Show that $R_A(x) \leq \lambda_1$. [9 marks]

Since clearly $R_A(\xi_1) = \lambda_1$, we have in fact shown the existence of the maximum twice, using two different arguments! In summary, we now know the maximum exists, and that $\xi_1$ attains it. What we still have to show is that any vector in $S$ that is *not* an eigenvector for $\lambda_1$ does not maximize $R_A$.

(c) Recall that there may be several linearly independent eigenvectors that all have eigenvalue $\lambda_1$. Let these be $\xi_1, \ldots, \xi_k$, for some $k \leq n$. Show that, if $x \in \mathbb{R}^n$ is not contained in $\operatorname{span}\{\xi_1, \ldots, \xi_k\}$, then $R_A(x) < \lambda_1$. [5 marks]