Homework 3, Problems 1 and 2

Name: Shashank Singh[1] [2]

10-715 Advanced Introduction to Machine Learning

Due: Wednesday October 15, 2014

1 Dimensionality Reduction (Samy)

1.1 Principal Component Analysis

1. We want to maximize $\frac{1}{n}\sum_i = 1^n (a_1^T X_i)^2$ subject to $\|a_1\|_2^2 = 1$. The Lagrangian of this is

$$L(a_1, \lambda) = \frac{1}{n}\sum_{i=1}^{n}(a_1^T X_i)^2 - \lambda(\|a_1\|_2^2 - 1).$$

Thus, stationarity gives

$$0 = \nabla_{a_1} L(a_1, \lambda) = \frac{1}{n}\sum_{i=1}^{n} 2(a_1^T X_i)X_i - 2\lambda a_1,$$

which can be re-written as $n\lambda a_1 = \sum_{i=1}^n X_i X_i^T a_1 = X^T X a_1$. Maximizing λ tells us that a_1 is the first right singular vector of X. ■

2. Now, we are maximizing $\frac{1}{n}\sum_{i=1}^n (a_{k+1}^T \tilde{x}_i)^2$ subject to $\|a_{k+1}\|_2^2 = 1$. For $\tilde{x} = [\tilde{x}_1^T; \cdots; \tilde{x}_n^T]$, it follows from part 1. that a_{k+1} is the first right singular vector of $\tilde{x}$.

Since singular vectors are orthogonal, the singular vectors of $\tilde{x}$ are the same as those of X, and the singular values of $\tilde{x}$ are the singular values of X with the first k replaced by zero. Hence, a_{k+1} is the $(k+1)^{th}$ right singular vector of X. ■

1.2 Affine Subspace Identification (ASI)

1. By construction of A_2, b_2, and Z_2,

$$\begin{aligned}\|x_i - A_2 Z_{2,i} - b_2\| &= \left\|x_i - A_1 C^{-1}(Z_1 C^T + 1d^T)_i - b_1 + A_1 C^{-1} d\right\| \\ &= \left\|x_i - A_1 C^{-1}\left((Z_1 C^T + 1d^T)_i - d\right) - b_1\right\| \\ &= \left\|x_i - A_1 C^{-1}\left(C Z_{1,i} + d_i - d\right) - b_1\right\| = \|x_i - A_1 Z_{1,i} - b_1\|,\end{aligned}$$

and it follows that $J(A_2, b_2, Z_2) = J(A_1, b_1, Z_1)$. ■

1. Stationarity gives

$$0 = \nabla_b \sum_{i=1}^{n} \|Az_i + b - x_i\|^2 = \sum_{i=1}^{n} Az_i + b - x_i,$$

which implies $b = \frac{1}{n}\sum_{i=1}^n x_i - Az_i$,

2.

[1] sss1@andrew.cmu.edu

[2] Machine Learning Department & Department of Statistics

1.3 Factor Analysis (FA)

1.

2.

3.

4.

5.

1.4 Experiment

Questions

1.

2.

3. This isn't surprising because, by definition, ASI and PCA choose a representations that minimize reconstruction error of the data, whereas FA instead maximizes a likelihood and normalized PCA minimizes the reconstruction error of the normalized data (which, e.g. weighs directions of large variance less than unnormalized PCA).

Results

Code

2 Some Random Topics (Samy)

2.1 K-means Clustering

1. For $K \in \mathbb{N}$, suppose $\mu_1, \dots, \mu_K$ and $f : \mathcal{X} \to \{1, \dots, K\}$ minimizing $J(\mu_1^K, f; X_1^n)$. Let $\mu_{K+1} \in \mathcal{X}$, and let $g : \mathcal{X} \to \{1, \dots, K, K+1\}$ such that $g(x) = f(x)$ for all $x \in \mathcal{X}$. Then, $J_K(X_1^n) = J(\mu_1^{K+1}, g; X_1^n) = J(\mu_1^K, f; X_1^n)$, and hence $J_K(X_1^n) \leq J_{K+1}(X_1^n)$. ■

2. For $i \in \mathbb{N}, j \in \{1, \dots, K\}$, let $\mu_j(i)$ denote the j^{th} centroid and let f_i denote the cluster assignments computed in the i^{th} step of the algorithm. Setting the gradient of the objective in μ to 0 shows that

$$\mu_j(i) = \frac{1}{n_j} \sum_{k=1}^{n} \mathbb{1}(f_i(X_k) = j) X_k = \arg\min_{\mu} \sum_{k=1}^{n} \mathbb{1}(f_i(X_\ell) = j) \|X_k - \mu_j(i)\|^2,$$

so that $\mu_1^K(i) = \arg\min_{\mu_1^K} J(\mu_1^K, f_i; X_1^n)$. By construction of the K-means algorithm,

$$f_i = \arg\min_{f} J(\mu_1^K(i-1), f; X_1^n).$$

Hence, J is non-increasing with each iteration. Since there are only finitely many possible values of μ_1^K and f (one per partition of $\{1, \ldots, n\}$ into K subsets), $J(\mu_1^K(i), f_i; X_1^n)$ is eventually constant and the algorithm terminates. ■

2.2 Independent Components Analysis

1. See Figure 1 and code below.

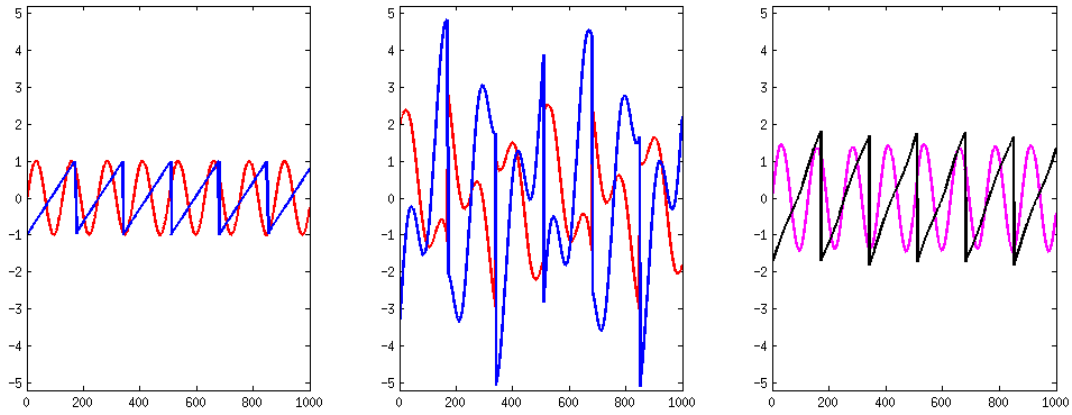

Figure 1: Original, mixed, and unmixed signal for question 1 in Section 2.2.

Code:

```
N = 1000;
signal1 = sin(linspace(0,50,N));
signal2 = sawtooth(linspace(0,37,N));
subplot(1,3,1); % plot original signals
plot(1:N, signal1, 'r', 1:N, signal2, 'b', 'linewidth', 2); ylim([-5.2 5.2]);

mix1 = signal1 - 2*signal2;
mix2 = 1.73*signal1 + 3.41*signal2;

subplot(1,3,2); % plot mixed signals
plot(1:N, mix1, 'r', 1:N, mix2, 'b', 'linewidth', 2); ylim([-5.2 5.2]);

rec = fastica([mix1; mix2]);
rec1 = rec(1,:); rec2 = rec(2,:);

subplot(1,3,3); % plot unmixed signals
plot(1:N, rec1, 'm', 1:N, rec2, 'k', 'linewidth', 2); ylim([-5.2 5.2]);
```

2. We are trying determine a matrix factorization $Y = MX$ (where Y is the input to the algorithm, and M and X are the unknown mixing matrix and original components, respectively). For any (nonsingular) scaling (i.e., diagonal) matrix D, $Y = MD^{-1}DX$, and the rows of DX are still independent, i.e., the input to the algorithm is invariant to scaling the original components and inversely scaling the mixing matrix. ■

Homework 3, Problems 3 and 4

Name: Shashank Singh[3] [4]

10-715 Advanced Introduction to Machine Learning

Due: Wednesday October 15, 2014

3 Graphical Models (Veeru)

1. (a) The distribution factors as $P(I, W, G, L) = P(I)P(W)P(G|I, W)P(L|G)$.

   (b) We need to know 9 parameters for $P(I)$, 9 parameters for $P(W)$, $9 \cdot 10^2 = 900$ parameters for $P(G|I, W)$, and $9 \cdot 10 = 90$ parameters for $P(L|G)$, for a total of 9+9+900+90 = $\boxed{1008}$ parameters.

2. (a) No; I and W are related through G, which depends on L, e.g., a student who did little work and received an A was likely very intelligent.

   (b) No; I and G are directly connected.

   (c) Yes; L depends only on G.

   (d) No; G depends on L not through W; e.g., if L is a function of G.

3. (a) We first compute the probabilities of $G = 1$ and $G = 2$:

$$\begin{aligned} P(G = 1) &= P(G = 1|I = 0, W = 0)P(I = 0)P(W = 0) \\ &\quad + P(G = 1|I = 0, W = 1)P(I = 0)P(W = 1) \\ &\quad + P(G = 1|I = 1, W = 0)P(I = 1)P(W = 0) \\ &\quad + P(G = 1|I = 1, W = 1)P(I = 1)P(W = 1) \\ &= (0.1)(0.3)(0.2) + (0.6)(0.3)(0.8) + (0.7)(0.7)(0.2) + (0.1)(0.7)(0.8) = 0.304. \end{aligned}$$

   Since $P(G = 2|I = 0, W = 0) = 0$,

$$\begin{aligned} P(G = 1) &+ P(G = 2|I = 0, W = 1)P(I = 0)P(W = 1) \\ &\quad + P(G = 2|I = 1, W = 0)P(I = 1)P(W = 0) \\ &\quad + P(G = 2|I = 1, W = 1)P(I = 1)P(W = 1) \\ &= (0.3)(0.3)(0.8) + (0.2)(0.7)(0.2) + (0.9)(0.7)(0.8) = 0.604. \end{aligned}$$

   Since $P(L = 1|G = 0) = 0$, the probability of $L = 1$ is

$$\begin{aligned} P(L = 1) &= P(L = 1|G = 1)P(G = 1) + P(L = 1|G = 2)P(G = 2) \\ &= (0.3)(0.304) + (0.8)(0.604) = \boxed{0.5744.} \end{aligned}$$

   (b) Since $P(L = 1|G = 0) = 0$, the probability of $L = 1$ given $I = 1$ and $W = 0$ is

$$\begin{aligned} P(L = 1|I = 1, W = 0) &= P(L = 1|G = 1)P(G = 1|I = 1, W = 0) \\ &\quad + P(L = 1|G = 2)P(G = 2|I = 1, W = 0) \\ &= (0.3)(0.7) + (0.8)(0.2) = \boxed{0.37.} \end{aligned}$$

[3] sss1@andrew.cmu.edu

[4] Machine Learning Department & Department of Statistics

4. (a) The probability of this sequence is approximately 6.1939×10^{-5}.
   (b) The most likely sequence is $2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1$.

Code for 4(a) and 4(b) is included below.

Code

```
T = [0.7 0.2 0.1; 0.2 0.7 0.1; 0.1 0.3 0.6];
P = [0.9 0.1 0 0; 0.8 0.1 0.1 0; 0.5 0.3 0.1 0.1];
data = min(3,[1 5 0 1 1 0 0 0 0 0 0 0]);

[seq p_tot] = MLE_seq(T, P, data)

function [seq p_tot] = MLE_seq(T, P, data)

  n = length(data);
  k = size(T,1);
  seq = zeros(1,n); % most likely sequence
  l = -Inf; % log-likelihood of seq
  p_tot = 0; % probability of data

  for i=0:(k^n - 1) % for each possible sequence
    seq_new = (dec2base(i, k, ceil(log(k^n)/log(k)) - 1) - '0') + 1;
    l_new = 0; % log-likelihood of new_seq
    for j=1:(n - 1) % for each transition
      l_new = l_new + log(T(seq_new(j), seq_new(j + 1)));
    end
    for j=1:n % for each observation
      l_new = l_new + log(P(seq_new(j), data(j) + 1));
    end

    p_tot = p_tot + exp(l_new);

    if l_new > l
      seq = seq_new - 1;
      l = l_new;
    end
  end
end
```

4 Markov Chain Monte Carlo (Veeru)

4.1 Markov Chain properties

1. Recall first that T and T^T have the same eigenvalues, since

$$\det(A^T - \lambda I) = \det((A^T - \lambda I)^T) = \det(A - \lambda I),$$

and λ is an eigenvalue if and only if the above determinant is 0. For $i \in [n]$ since $j \mapsto T_{ij}$ is the probability density function of transitions from state i, $\sum_{j=1} T_{ij} = 1$, and so, if $\mathbb{1}_n \in \mathbb{R}^n$ is the vector of all ones, then $\mathbb{1}_n = T\mathbb{1}_n$. Thus, 1 is an eigenvalue of T, and hence of T^T. ■

2. For

$$v_1 = \begin{bmatrix} 0 & 7/6 & 0 & 1 \end{bmatrix} \quad \text{and} \quad v_2 = \begin{bmatrix} 4/7 & 0 & 1 & 0 \end{bmatrix},$$

$v_1 T = v_1$ and $v_2 T = v_2$ (i.e., $v_1/\|v_1\|_1$ and $v_2/\|v_2\|_1$ both encode stationary distributions). Intuitively, this reflects the fact that the Markov chain is reducible. ■

3. Consider a 2-state Markov chain with transition matrix and initial distribution

$$\boxed{T = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \text{and} \quad p_0 = \begin{bmatrix} 1 & 0 \end{bmatrix}.}$$

Then, $p_0 T^n = p_0$ when n is even, and $p_0 T^n = [0 \quad 1]$ when n is odd, so that $p_0 T^n$ does not converge. This occurs because the Markov chain is periodic.

4.2 Detailed balance property

1.

2.

4.3 Experiments

1. (a) The means are: $4.9712, -5.0576, 4.9554, -5.1235, 5.0784$, and -5.1124, which differ substantially from the true population mean of 0. From Figure 2, it is clear that the sampler is getting "stuck" in the two modes (-5 and 5) of the target distribution.

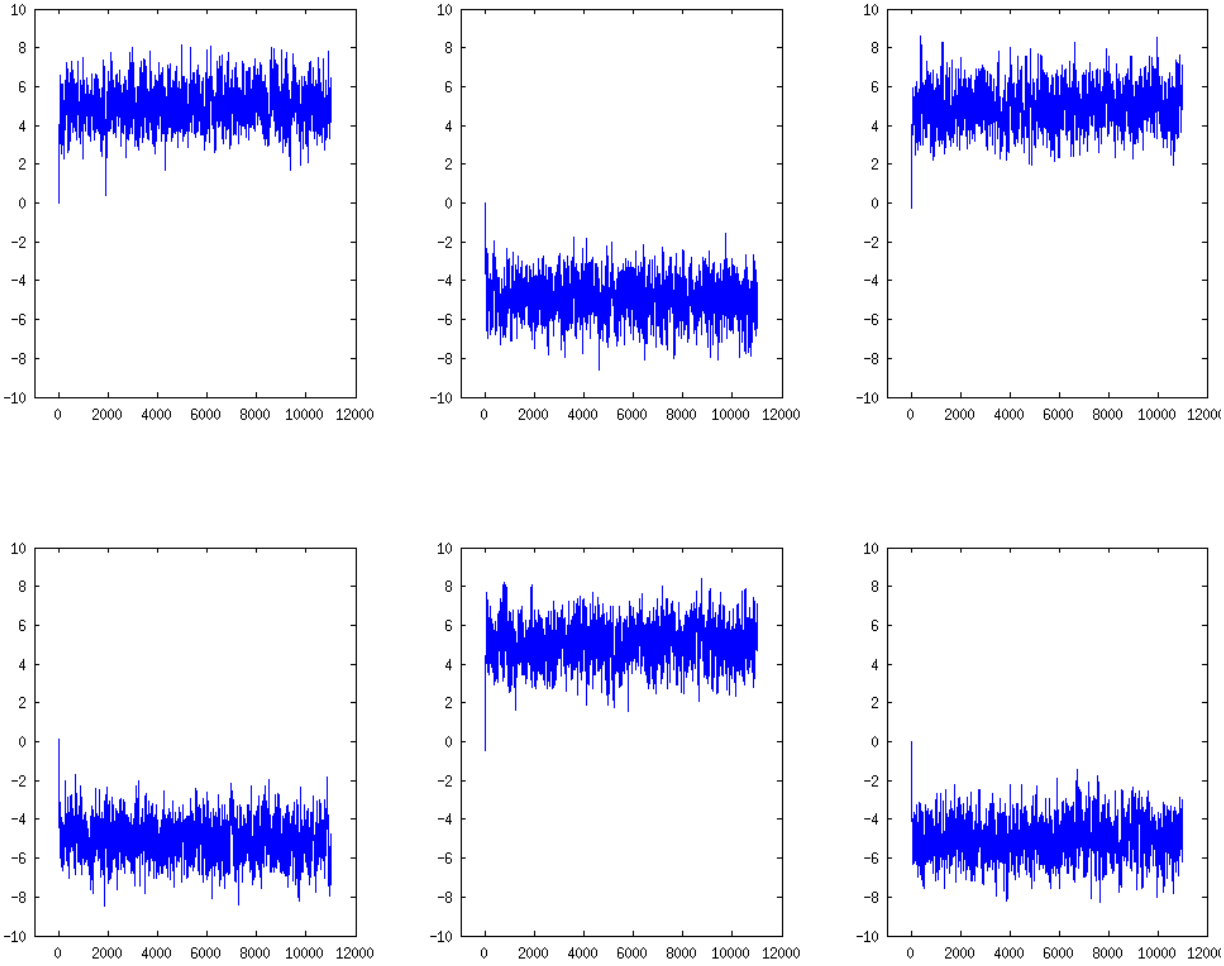

Figure 2: Plots of Metropolis-Hastings samples (including burnin samples) for each of 6 trials.

(b) The means are now $-0.1101, 0.1731, -0.6260, 0.8826, 0.0223$, and 0.7333, much closer to the true population mean of 0. This larger σ is better because it allows the sampler to jump between the two modes of the target distribution.

(c) The average of 50 sample means is -0.7939.

Code for Problems 1(a)-(c) of Section 4.3 is included below:

Code

```
b = 10^4; % number of burnin samples
n = 10^3; % number of mixed samples
sigma = 0.5; % standard deviation of proposal distribution; also tried 5
Q = @(x_star,x) normpdf(x_star, x, sigma); % proposal density
x0 = 0; % initial x
m = 6; % number of repetitions; also tried 50
mu = 5; % mean of target distribution
pi = @(x) exp(-(x - mu)^2/2) + exp(-(x + mu)^2/2); % target density

means = zeros(m,1);

for rep = 1:m

  samples = zeros(n + b, 1);
  samples(1) = x0;

  for sample = 2:(n + b)
    u = unifrnd(0,1);
    x = samples(sample - 1); % previous sample
    x_star = normrnd(samples(sample - 1),sigma); % sample from proposal

    if u < min(1, pi(x_star)*Q(x, x_star)/(pi(x)*Q(x_star, x)))
      samples(sample) = x_star;
    else
      samples(sample) = x;
    end
  end
  means(rep) = mean(samples(( b + 1):end)); % mean of mixed samples
end
```

2. (a) By Bayes' Rule,

$$\begin{aligned} p(z_i = k|x, z_{-i}, \mu) &\propto p(x_i|z_i = k, x_{-i}, z_{-i}, \mu)p(z_i = k|x_{-i}, z_{-i}, \mu) \\ &= p(x_i|z_i = k, \mu_k)p(z_i = k), \end{aligned}$$

since z_i is independent of x_{-i}, z_{-i}, and μ, and x_i is conditionally independent of x_{-i}, z_{-i}, and μ_{-i}. Similarly, applying Bayes' Rule again,

$$\begin{aligned} p(\mu_k = u|x, z, \mu_{-k}) &\propto p(x|\mu_k = u, z, \mu_{-k})p(\mu_k = u|z, \mu_{-k}) \\ &= p(\mu_k = u|z, \mu_{-k}) \prod_{i=1}^{n} p(x_i|\mu_k = u, z, \mu_{-k}) \\ &= p(\mu_k = u) \prod_{\{i:z_i=k\}} p(x_i|\mu_k = u, z_i = k), \end{aligned}$$

since (for $i \neq j$) x_i and x_j are conditionally independent given μ and z, and since μ_k is independent of z and μ_{-k}, and x_i is conditionally independent of μ_k given $z_i \neq k$, and x_i is conditionally independent of z_{-k} and μ_{-k} given $z_i = k$. ■

Note that we can easily resample our estimates of $\mu_1, \ldots, \mu_k$ because our normal prior is conjugate to the normal distribution of the data.

(b) The final clustering is shown in Figure 3, and code is included below.

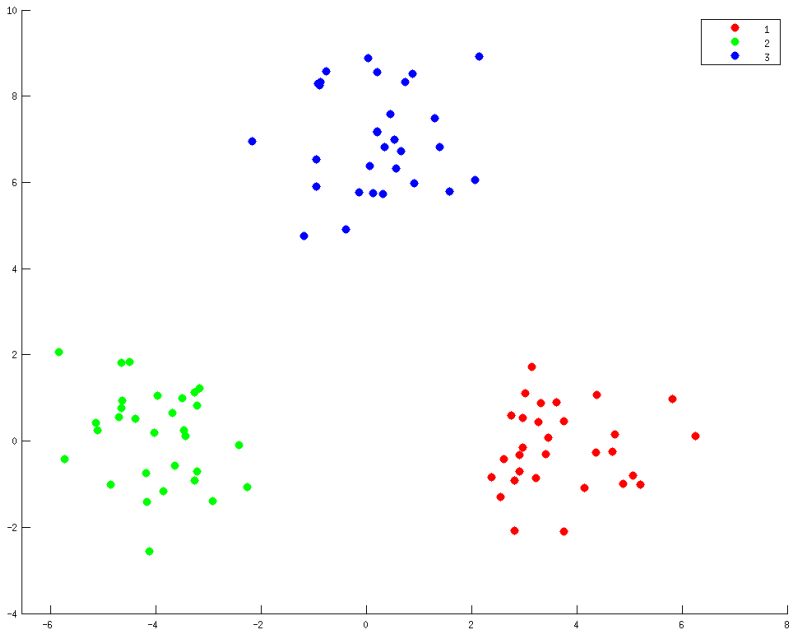

Figure 3: Cluster assignments of 90 data points (30 from each cluster).

Code

```
n_iters = 5001; % number of Gibbs sampling iterations
n = 30; % number of samples per cluster
true_mus = [-4 0; 4 0; 0 7]; % true cluster means
sigma = 1; % cluster standard deviations
[k d] = size(true_mus); % number of clusters and dimension of data
N = n*k; % total number of samples

Xs = zeros(0, d);
for cluster = 1:k % generate data from each cluster
  Xs = [Xs; normrnd(repmat(true_mus(cluster, :), n, 1), sigma, n, d)];
end

Zs = randi(k, N, 1); % initial (uniformly random) cluster assignments
Mus = mvnrnd(zeros(d, 1), eye(d), k); % initial random cluster means

for iter = 1:n_iters

  for sample = 1:N % resample cluster labels for each sample
    for cluster = 1:k % compute conditional probability given each cluster
      w(cluster) = mvnpdf(Xs(sample, :), Mus(cluster, :));
    end
    % sample cluster label by weight w
    Zs(sample) = sum(rand >= cumsum(w./sum(w))) + 1;
  end

  for cluster = 1:k % resample cluster means for each cluster
    clust_Xs_mean = mean(Xs(Zs &=&  cluster, :), 1);
    n_clust = sum(Zs &=&  cluster);
    sd = 1/(n_clust + 1);
    Mus(cluster, :) = mvnrnd(clust_Xs_mean*n_clust*sd, eye(d)*sd);
  end
end
```