6.867 Machine Learning

Solutions for Problem Set 5

Monday, November 24

Problem 1: Description Length and Bayesian score

Set-up. Under model $\mathcal{M}_k$, with random parameters $\theta_k \sim p(\theta_k)$, a sample $x_i \in [0,1]$ is generated with uniform probability density over the unit interval, then a Bernoulli random variable $y_i \in \{0,1\}$ is generated with probability of $y=1$ given by $\theta(x_i) = \theta_{j;k}$ when $x_i \in S_{j;k} = [(j-1)/2^k, j/2^k]$. We are given a set of independent samples $\{(x_i, y_i), i = 1, ..., n\}$ (all generated according to the same unknown θ_k) and wish to estimate which model $\mathcal{M}_k$ generated this data set (this is related to minimizing the description length of the data set as discussed in the problem set).

(1-1) (2pts) The uninformative prior $p(\theta_k)$, for model $\mathcal{M}_k$, is a constant $p(\theta_k) = 1/V$ for all $\theta_k = (\theta_{1;k}, \ldots, \theta_{2^k;k})' \in \Theta_k$ where $\Theta_k = \{\theta_k : \theta_{j;k} \in [0,1], j = 1, \ldots, 2^k\}$. The normalization constraint $\int_{\Theta_k} d\theta_k = 1$ requires that $V = \int_{\Theta_k} d\theta_k$, the (hyper-)volume of Θ_k, which is just $V = 1$:

$$\begin{aligned} V &= \int_{\theta_{1;k}} \cdots \int_{\theta_{2^k;k}} d\theta_{1;k} \cdots d\theta_{2^k;k} && (1) \\ &= \prod_{j=1}^{2^k} \underbrace{\int_{\theta_{j;k}=0}^{1} d\theta_{j;k}}_{1} && (2) \\ &= 1 && (3) \end{aligned}$$

Hence, $p(\theta_k) = 1$ for all $\theta_k \in \Theta_k$ and is zero otherwise.

(1-2) (5pts) The conditional probability of outputs $\mathbf{y}_{j;k} = \{y_i : x_i \in S_{j;k}\}$ given the corresponding inputs $\mathbf{x}_{j;k} = \{x_i : x_i \in S_{j;k}\}$ is given by

$$\begin{aligned} P(\mathbf{y}_{j;k}|\mathbf{x}_{j;k}, \theta_{j;k}) &= \prod_{i:x_i \in S_{j;k}} P(y_i|x_i, \theta_{j;k}) && (4) \\ &= \theta_{j;k}^{n_{j;k}(1)} (1-\theta_{j;k})^{n_{j;k}(0)} && (5) \end{aligned}$$

due to the conditional independence of the outputs given the inputs. Hence, since $\mathbf{y} = \cup_j \mathbf{y}_{j;k}$ and $\mathbf{x} = \cup_j \mathbf{x}_{j;k}$, the conditional probability over all samples is just:

$$P(\mathbf{y}|\mathbf{x}, \theta_k) = \prod_{j=1}^{2^k} P(\mathbf{y}_{j;k}|\mathbf{x}_{j;k}, \theta_{j;k}) \qquad (6)$$

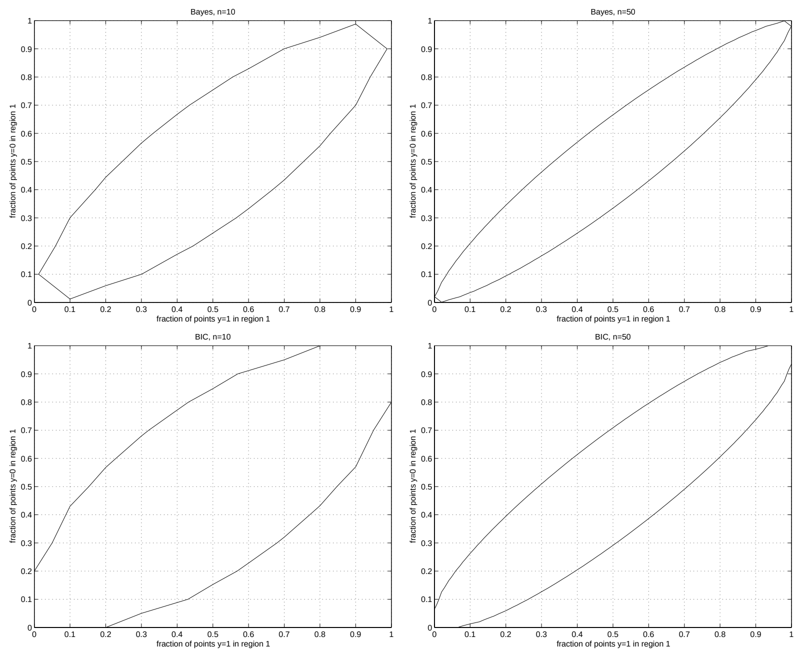

Figure 1: Plots for Problem 1. Bayesian score (top) and BIC (bottom) for $n = 10$ (left) and $n = 50$ (right).

$$= \prod_{j=1}^{2^k} \theta_{j;k}^{n_{j;k}(1)} (1-\theta_{j;k})^{n_{j;k}(0)} \tag{7}$$

(1-3) (10pts) The Bayesian score under model $\mathcal{M}_k$, where $\theta_k \sim p(\theta_k)$ is the uninformative prior, is given by simply integrating over Θ_k. Since $P(\mathbf{x}|\mathbf{y}, \theta_k)$ factors, the multiple integral seperates into a product of simple integrals:

$$\begin{aligned}
\log P(\mathbf{y}|\mathbf{x}, \mathcal{M}_k) &= \log \int_{\theta_k} P(\mathbf{y}|\mathbf{x}, \theta_k) d\theta_k & (8)\\
&= \log \int_{\theta_k} \prod_{j=1}^{2^k} \theta_{j;k}^{n_{j;k}(1)} (1-\theta_{j;k})^{n_{j;k}(0)} d\theta_k & (9)\\
&= \log \prod_{j=1}^{2^k} \int_{\theta_{j;k}} \theta_{j;k}^{n_{j;k}(1)} (1-\theta_{j;k})^{n_{j;k}(0)} d\theta_{j;k} & (10)\\
&= \log \prod_{j=1}^{2^k} \frac{n_{j;k}(0)! n_{j;k}(1)!}{(n_{j;k}+1)!} & (11)\\
&= \sum_j \log n_{j;k}(0)! + \log n_{j;k}(1)! - \log(n_{j;k}+1)! & (12)
\end{aligned}$$

(1-4) (10pts) In this problem we consider model selection between $\mathcal{M}_0$ and $\mathcal{M}_1$. We added the following code to `BayesCompare.m` which computes the Bayesian score under these two models:

```
Bscore0 = 2*lnfac(n) - lnfac(2*n+1);
Bscore1 = lnfac(n11) + lnfac(n10) - lnfac(n11+n10+1) + ...
  lnfac(n21) + lnfac(n20) - lnfac(n21+n20+1);
```

See `hw5prob1.m` (bottom of page) and Figure 1.

Interpretation. The outlined region in each plot corresponds the the set of data sets (expressed in terms of the empirical distribution) which would have a higher Bayesian score under the simpler model $\mathcal{M}_0$ then under the refined model $\mathcal{M}_1$. As n increases, we are more apt to select the higher-order model (since we have more data to support estimation of a more complex probability distribution). Hence, the region where $\mathcal{M}_0$ is selected shrinks when we increase n.

(1-5) (10pts) Here, we approximate the Bayesian score by the BIC and consider how this effects model selection between $\mathcal{M}_0$ and $\mathcal{M}_1$. See `hw5prob1.m` and Figure 1. It appears that the BIC provides a good approximation to the Bayesian score for $n = 50$, but only roughly for $n = 10$. This makes sense as the BIC is supposed to by an asymptotic approximation for the Bayesian score (really only valid for large n). In both cases, the BIC will tend to select simpler models then would be selected by the actual Bayesian score. It is still the case that, as we increase n, the BIC favors more complex models.

```
% hw5prob1.m

% (1-4)
figure(1);
compareBayes(10);
title('Bayes, n=10');
print -deps hw5prob1_4_fig1.eps;

figure(2);
compareBayes(50);
title('Bayes, n=50');
print -deps hw5prob1_4_fig2.eps;

% (1-5)
figure(3);
compareBIC(10);
title('BIC, n=10');
print -deps hw5prob1_4_fig3.eps;

figure(4);
compareBIC(50);
title('BIC, n=50');
print -deps hw5prob1_4_fig3.eps;
```

Problem 2: EM and mixture models

The Kullback-Leibler (KL) divergence $J(Q, P) = E_Q\{\log Q(x, y)/P(x, y)\}$ is best understood as an information theoretic measure of "distance" between probability distributions. By the information inequality[1], KL is non-negative and is zero if and only if $P(x, y) = Q(x, y)$ for all x, y. Hence, we may view the EM algorithm as performing a sequence of KL "projections". Let $\mathcal{M} = \{P : P(x, y) = P_\theta(x, y), \theta \in \Theta\}$ be our model and let $\mathcal{D} = \{Q : Q(x) = \sum_y Q(x, y) = \frac{1}{n}\sum_i \delta(\mathbf{x}|\mathbf{x}_i)\}$ where $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are the available data. Pick $P^{(0)} \in \mathcal{M}$ and then, for $k = 0, 1, 2, \ldots$, do:

$$Q^{(k)} = \arg\min_{Q \in \mathcal{D}} J(Q, P^{(k)}) \tag{13}$$

$$P^{(k+1)} = \arg\min_{P \in \mathcal{M}} J(Q^{(k)}, P) \tag{14}$$

The E-step projects $P^{(k)}$ to the set of probability distributions $\mathcal{D}$ consistent with the observed data. The M-step then projects $Q^{(k)}$ back to the model $\mathcal{M}$. Essentially, we are trying to find the distribution P which comes "nearest" (in KL-divergence) to the set $\mathcal{D}$.

Discrete EM

In the first part of the problem we consider a discrete version of EM where both the visible variable $\mathbf{x}$ and the hidden variable y are discrete random variables.

(2-1) (5pts) At first, we do not impose any restriction on our model $P(x, y)$ and explore what the EM algorithm would reduce to in this case.

E-step. In the problem set, you are told that the solution to the E-step, minimizing the KL-divergence $J(Q, P)$ w.r.t $Q \in \mathcal{D}$, is given by setting $Q(y|x) = P(y|x)$. To see why[2], decompose the (joint) KL-divergence $J(Q, P) = J_{x,y}(Q, P)$ as

$$J_{x,y}(Q, P) = E_Q\{\log \frac{Q(x, y)}{P(x, y)}\} \tag{15}$$

$$= E_Q\{\log \frac{Q(y|x)}{P(y|x)}\} + E_Q\{\log \frac{Q(x)}{P(x)}\} \tag{16}$$

$$= J_{\mathrm{y|x}}(Q, P) + J_{\mathrm{x}}(Q, P) \tag{17}$$

where $J_{\mathrm{y|x}}(Q, P)$ is the conditional KL-divergence between $Q(y|x)$ and $P(y|x)$. By the information inequality[3], this is nonnegative and is zero if and only $Q(y|x) = P(y|x)$ for all y and all x where $Q(x) > 0$. Hence, we must set $Q(y|x_i) = P(y|x_i)$ for all samples x_i. Given $P(x, y)$ in the form $P(x, y) = P(x|y)P(y)$ we need to calculate the "reverse" conditional model employing Bayes rule:

$$Q(y|x) = \frac{P(x|y)P(y)}{\sum_{y'} P(x|y')P(y')} \tag{18}$$

[1]As shown in recitation, the information inequality follows from Jensen's inequality.

[2]The student is *not* required to prove this in their solutions.

[3]Derived in recitation using Jensen's inequality.

We use this formula to calculate each $Q(y|x_i)$ for all y and $i = 1, ..., n$.

M-step. Next, we consider the M-step. We choose $P \in \mathcal{M}$ so as to minimize $J(Q, P)$. We consider two separate arguments which give the same result (you could provide either argument in your solution):

KL Projection. Since the Kullback-Leibler divergence $J(Q, P)$ is non-negative and is zero when $P(x, y) = Q(x, y)$ for all x, y, it is minimized by setting $P(x, y) = Q(x, y)$ for all x, y. Equivalently, in terms of a generative model of the form $P(x, y) = P(x|y)P(y)$ we set

$$\begin{aligned} \hat{P}(y) &= Q(y) \quad \forall y & (19) \\ \hat{P}(x|y) &= Q(x|y) \quad \forall x, y & (20) \end{aligned}$$

Parametric Approach. We arrive at the same result by differential analysis. Parameterize $P(x, y) = \theta_{x,y}$ subject to the constraint $\sum_{x,y} \theta_{x,y} = 1$. We then maximize the Lagrangian objective

$$J(\theta) = E_{(\mathrm{x},\mathrm{y}) \sim Q(x,y)}\{\log P_\theta(\mathrm{x}, \mathrm{y})\} - \lambda \sum_{x,y} \theta_{x,y} \tag{21}$$

where we have introduced a Lagrange multiplier λ to enforce the constraint. Differentiating w.r.t. parameter $\theta_{x,y}$ we obtain:

$$\frac{\partial J(\theta; \lambda)}{\partial \theta_{x,y}} = \frac{Q(x, y)}{\theta_{x,y}} - \lambda \tag{22}$$

Setting this to zero gives $\hat{\theta}_{x,y} = Q(x, y)/\lambda$. Setting λ to satisfy $\sum_{x,y} \hat{\theta}_{x,y}(\lambda) = 1$ we must have $\lambda = 1$ s.t.

$$\hat{\theta}_{x,y} = Q(x, y) \tag{23}$$

which is the same conclusion as in the previous argument.

Why isn't this interesting? In this version of the EM algorithm, we placed no restriction on the structure of $P(x, y)$. Consequently, the EM algorithm actually "converges" after just one iteration. Moreover, what EM converges to in this case is not very informative. If $P^{(0)}$ is our initial guess, then for all $k \geq 1$ we have:

$$P^{(k)}(x, y) = P^{(0)}(y|x)Q(x) \tag{24}$$

Essentially, all we have done is reset the marginal distribution in x to the empirical distribution of the data $Q(x) = \sum_x \delta(x|x_i) = n(x)/n$ keeping the conditional distribution $P^{(0)}(y|x)$ fixed. But since $P^{(0)}(y|\mathbf{x})$ was chosen arbitrarily and $Q(x)$ is just the known empirical distribution of the data we really haven't "learned" anything at all.

(2-2) (10pts) We now impose the restriction that the components of $\mathbf{x} = (x_1, \ldots, x_d)$ are conditionally independent given y, e.g. that $P(\mathbf{x}|y) = \prod_{j=1}^d P(x_j|y)$. Our solution is intended to explicate the underlying structure of EM (you may have provided a more concise argument).

E-step. The E-step is the same as in the preceding problem, except that we now appeal to conditional independence to compute $P(\mathbf{x}|y) = \prod_j P(x_j|y)$:

$$Q(y|\mathbf{x}) = \frac{P(y)\prod_j P(x_j|y)}{\sum_{y'} P(y')\prod_j P(x_j|y')} \tag{25}$$

We use this formula to compute $Q(y|\mathbf{x}_i)$ for all y, i.

M-Step. Again, we consider two possible perspectives to solve the M-step.

KL Projection. For any $P \in \mathcal{M}$ we may write $P(\mathbf{x}, y) = P(y)\prod_j P(x_j|y)$. Consequently, we may decompose the (joint) Kullback-Leibler divergence as:

$$\begin{aligned} J_{\mathrm{x,y}}(Q,P) &= E_Q\{\log \frac{Q(\mathrm{x,y})}{P(\mathrm{x,y})}\} && (26)\\ &= E_Q\{\log \frac{Q(\mathrm{y})}{P(\mathrm{y})}\} + \sum_{j=1}^{d} E_Q\{\log \frac{Q(\mathrm{x}_j|y)}{P(\mathrm{x}_j|y)}\} + E_Q\{\log \frac{Q(\mathrm{x|y})}{\prod_j Q(\mathrm{x}_j|y)}\} && (27)\\ &= J_{\mathrm{y}}(Q,P) + \sum_j J_{\mathrm{x}_j|y}(Q,P) + I_Q(\mathrm{x}_1;\ldots;\mathrm{x}_d|y) && (28) \end{aligned}$$

Hence the optimization problem is separable s.t. we choose $P(y)$ to minimize $J_{\mathrm{y}}(Q,P)$, the marginal KL-divergence between $Q(y)$ and $P(y)$, and choose each $P(x_j|y)$ (for each j) to minimize $J_{\mathrm{x}_j|\mathrm{y}}(Q,P)$, the conditional divergence between $Q(x_j|y)$ and $P(x_j|y)$ averaged over $y \sim Q(y)$. These are Kullback-Leibler divergences so that, by the information inequality (Jensen's inequality), the minimum is zero and is achieved when:

$$\begin{aligned} P(y) &= Q(y) \;\; \forall y && (29)\\ P(x_j|y) &= Q(x_j|y) \;\; \forall j, x_j, y && (30) \end{aligned}$$

which is the solution to the M-step we sought. Note, the minimized KL-divergence is then $I_Q(\mathrm{x}_1;\ldots;\mathrm{x}_d|y)$, the average mutual information between d variables $x_1, \ldots, x_d$ after conditioning on y.

Parametric Approach. We can also arrive at this result by explicitly parameterizing $P_\theta(x, y)$ as

$$P_\theta(x,y) = \theta_y \cdot \prod_{j=1}^{d} \theta_{x_j|y} \tag{31}$$

and maximizing the objective:

$$J(\theta) = E_{(x,y)\sim Q(x,y)}\{\log P_\theta(x,y)\} \tag{32}$$

w.r.t. θ subject to the constraints $\sum_y \theta_y = 1$ and $\sum_{x_j} \theta_{x_j|y}$ for all j, y. This model has $2d+1$ independent parameters. Without the assumption of conditional independence we would need $2^d - 1$ parameters. For $d > 2$ this results in a reduction in model complexity (a very significant reduction as d becomes large).

Introducing a Lagrange multiplier for each constraint, we define the Lagrangian objective:

$$J(\theta;\lambda) = J(\theta) - \lambda_0 \sum_y \theta_y - \sum_{j,y} \lambda_{j,y} \sum_{x_j} \theta_{x_j|y} \tag{33}$$

Differentiating w.r.t. $\theta_{x_j|y}$ we obtain:

$$\frac{\partial J(\theta;\lambda)}{\partial \theta_{x_j|y}} = \frac{Q(x_j,y)}{\theta_{x_j|y}} - \lambda_{j,y} \tag{34}$$

Set this to zero and solve for $\hat{\theta}_{x_j|y}(\lambda_{j,y}) = Q(x_j,y)/\lambda_{j,y}$. Solve for the Lagrange multiplier to satisfy the constraint $\sum_{x_j} \hat{\theta}_{x_j|y} = 1$ which gives $\lambda = \sum_{x_j} Q(x_j,y) = Q(y)$ so that

$$\hat{\theta}_{x_j|y} = Q(x_j|y) \tag{35}$$

Similarly, differentiating w.r.t θ_y, we obtain:

$$\frac{\partial J(\theta;\lambda)}{\partial \theta_y} = \frac{Q(y)}{\theta_y} - \lambda_0 \tag{36}$$

which gives $\hat{\theta}_y(\lambda_0) = \frac{Q(y)}{\lambda_0}$. But, requiring $\sum_y \hat{\theta}_y(\lambda_0) = 1$, we must have $\lambda_0 = 1$ s.t.

$$\hat{\theta}_y = Q(y) \tag{37}$$

Hence, we arrive at the final answer $\hat{P}(x,y) = Q(y) \cdot \prod_{j=1}^{d} Q(x_j|y)$ which is the same answer we got previously.

Calculation of $Q(y)$ and $Q(x_j|y)$. At this point, we have essentially specified the EM algorithm. However, we still need to explicitly show how to compute $Q(y)$ and $Q(x_j|y)$ as required in the M-step. One way to compute these probabilities exploits the fact that $Q(x)$ is the empirical distribution so that our calculations may be expressed in terms of the probabilities $Q(y|\mathbf{x}_i)$ for all samples $i = 1,\ldots,n$. We may express $Q(\mathbf{x},y)$ as:

$$\begin{aligned} Q(\mathbf{x},y) &= Q(y|\mathbf{x})Q(\mathbf{x}) && (38)\\ &= Q(y|\mathbf{x}) \left(\frac{1}{n} \sum_i \delta(\mathbf{x}|\mathbf{x}_i) \right) && (39)\\ &= \frac{1}{n} \sum_i Q(y|\mathbf{x}_i)\delta(\mathbf{x}|\mathbf{x}_i) && (40) \end{aligned}$$

Then, $Q(y)$ is given by:

$$\begin{aligned} Q(y) &= \sum_{\mathbf{x}} Q(\mathbf{x},y) && (41)\\ &= \frac{1}{n} \sum_i Q(y|\mathbf{x}_i) \underbrace{\left(\sum_{\mathbf{x}} \delta(\mathbf{x}|\mathbf{x}_i) \right)}_{1} && (42)\\ &= \frac{1}{n} \sum_i Q(y|\mathbf{x}_i) && (43)\\ &= \frac{n(y)}{n} && (44) \end{aligned}$$

where $n(y) = \sum_i Q(y|\mathbf{x}_i)$. Similarly, we compute $Q(x_j, y)$ by summing $Q(x, y)$ over all possible values of $\mathbf{x}_{-j}$, i.e. by summing over all possible values of the other components of $\mathbf{x}$ excluding the jth component x_j:

$$\begin{aligned}
Q(x_j, y) &= \sum_{\mathbf{x}_{-j}} Q(x, y) && (45)\\
&= \frac{1}{n} \sum_i Q(y|\mathbf{x}_i) \underbrace{\left(\sum_{\mathbf{x}_{-j}} \delta(\mathbf{x}|\mathbf{x}_i) \right)}_{\delta(\mathbf{x}_{ij}|x_j)} && (46)\\
&= \frac{1}{n} \sum_{i|\mathbf{x}_{ij}=x_j} Q(y|\mathbf{x}_i) && (47)\\
&= \frac{n(x_j, y)}{n} && (48)
\end{aligned}$$

where $n(x_j, y) = \sum_{i|\mathbf{x}_{ij}=x_j} Q(y|\mathbf{x}_i)$. The sum is over all samples with jth component equal to x_j, i.e. where $\mathbf{x}_{ij} = x_j$. Finally, $Q(x_j|y)$ is given by:

$$\begin{aligned}
Q(x_j|y) &= \frac{Q(x_j, y)}{Q(y)} && (49)\\
&= \frac{n(x_j, y)}{n(y)} && (50)
\end{aligned}$$

Using these formulas, EM then has the form given in lecture. Our implementation for this EM algorithm is given in `discrete_em.m` reproduced below.

```
% discrete_em.m
function [P_y,P_xj_given_y,loglik] = discrete_em(X,N)

% N is number of possible values of y = 1,...,N
X=X+1; % elements of X are now 1 or 2 (more convenient for indexing)
n = size(X,1); % number of samples
d = size(X,2); % number components in each sample
% initialize model P
P_y = ones(N,1)/N; % uniform pmf over y
P_xj_given_y = cell(d,1); % store one transition prob matrix P(xj|y) for each j=1,...,d
for j=1:d  % indexes components of x
    P = rand(2,N); % transition probabilities from y=1,...,N to xk=0,1
    P = P * diag(sum(P).^(-1)); % normalize columns to sum to one.
    P_xj_given_y{j} = P;  % conditional pmf for xj given y
end
% allocate storage for Q(y|xi), y=0,1, i=1,...,n.
Q = zeros(2,n);
% run EM algorithm...
loglik = zeros(100,1);
for k = 1:100
    % E-step: compute Q(y|xi) for y=0,1 and i=1,...,n from P(y|x)
    for i=1:n % iterate over samples
```

```
        x = X(i,:);
        P_x = 0.0;
        for y=1:N  % calc prob of y given sample i from P
            Q(y,i)=P_y(y);
            for j=1:d
                Q(y,i)=Q(y,i)*P_xj_given_y{j}(x(j),y);
            end
            P_x = P_x + Q(y,i);
        end
        loglik(k) = loglik(k) + log(P_x);
        Q(:,i) = Q(:,i)/P_x; % normalize column to sum to one
    end
    loglik(k) = loglik(k)/n;  % avg log-lik per sample
    fprintf('iter: %d, loglik: %f\n',k,loglik(k));
    % M-step: update P based on Q
    for y=1:N
        n_y = sum(Q(y,:)');
        P_y(y) = n_y/n;
        for j=1:d
            for xj=1:2
                n_y_given_xj = sum(Q(y,find(X(:,j)==xj)));
                P_xj_given_y{j}(xj,y)=n_y_given_xj/n_y;
            end
        end
    end
end
```

We also wrote a script `hw5prob2_2.m` which runs EM several times and selects the best model. Plots of the log-likelihood vs. the number of EM iterations are shown in Figure 3. The best model was selected and is reproduced below:

```
max_loglik =
   -2.2166

P_y =
    0.3836
    0.6164

P_xj_given_y{1} =
    0.0135    0.9228
    0.9865    0.0772

P_xj_given_y{2} =
    0.0911    0.2305
    0.9089    0.7695

P_xj_given_y{3} =
    0.1378    0.7449
```

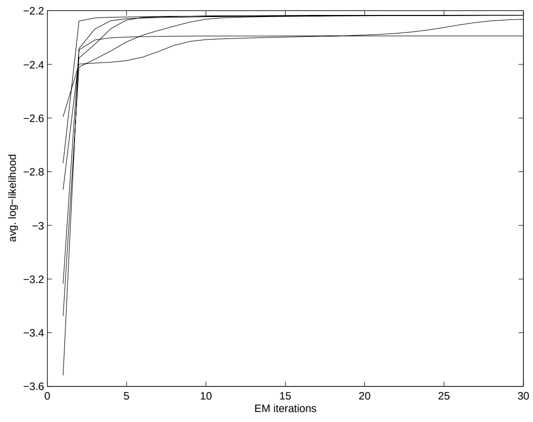

Figure 2: Plot for problem (2-2).

```
    0.8622    0.2551

P_xj_given_y{4} =
    0.8855    0.6640
    0.1145    0.3360
```

(2-3) (optional) The analysis and MATLAB code given in the previous problem applies here as well (N=2 previously). By increasing N, the number of hidden states y, we can refine our model to obtain a better fit to the data. However, we also run the risk of overfitting the data. To select the appropriate value of N, we maximize the BIC. The form of the BIC appropriate here is:

$$BIC(N) = \sum_{i=1}^{n} \log \hat{P}_N(\mathbf{x}_i) - \frac{1}{2}\log(n)K_N \tag{51}$$

where the likelihood of sample $\mathbf{x}_i$ under model N is

$$\hat{P}_N(\mathbf{x}_i) = \sum_{y=1}^{N} \hat{P}_N(y) \prod_{j=1}^{d} \hat{P}_N(\mathbf{x}_{ij}|y) \tag{52}$$

and the model complexity K_N (the number of independent model parameters) is:

$$K_N = (N-1) + Nd \tag{53}$$

Your MATLAB code should run EM for $m = 2, 3, 4$ and select the model with the highest BIC. (MATLAB code/results omitted).

Gaussian Mixtures and EM

(2-4) (10pts) The following script `hw5prob2_4.m` runs `em_mix.m` six times for each $m = 2, 3, 4, 5$, select the best model for each m, and evaluates the BIC.

```
% hw5prob2_4.m
clear;
close all;
load hw5em2.mat;
X = data2;
n = size(X,1);
d = size(X,2);

for m=2:5
  fprintf('m = %d\n',m);
  max_ll = -inf;
  for k=1:6
    [param,hist,ll] = em_mix(X,m);
    if (ll(end)>max_ll)
      max_ll = ll(end);
      eval(sprintf('print -depsc hw5prob2_4_fig%d.eps',m));
    end
  end
  model_complexity = m*(d + d*(d+1)/2) + (m-1);
  bic = max_ll - 0.5 * log(n) * model_complexity
end
```

The plots for each m are shown in Figure 3. We found that the BIC (in this run atleast) was maximal for $m = 4$, then $BIC \approx -0.00417$. This appears to provide a good fit for the data. Apparently, there are 3 well-defined clusters with high prior probabilities and one weaker cluster with low prior probability.

(2-5) (optional) The modified EM procedure is the same except that we now calculate the single covariance matrix Σ_0 according to the formula:

$$\Sigma_0^{(k+1)} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} P^{(k)}(j|i)(\mathbf{x}_i - \mu_j^{(k+1)})(\mathbf{x}_i - \mu_j^{(k+1)})' \tag{54}$$

where $P^{(k)}(j|i) = P^{(k)}(y = j|\mathbf{x} = \mathbf{x}_i)$ is computed from the previous estimate of the mixture model. Once we have estimated a mixture model for each m, we evalute the BIC as a function of m according to the formula:

$$BIC(m) = \sum_i \hat{P}_m(\mathbf{x}_i) - \frac{1}{2}\log(n)\{(m-1) + m \cdot (\frac{1}{2}d(d+1) + d)\} \tag{55}$$

and then select the model with the highest BIC. (MATLAB code/results omitted).

Problem 3: Clustering

(3-1) (10pts) Here is the modified code `kmeans.m`:

```
y = zeros(n,1);
```

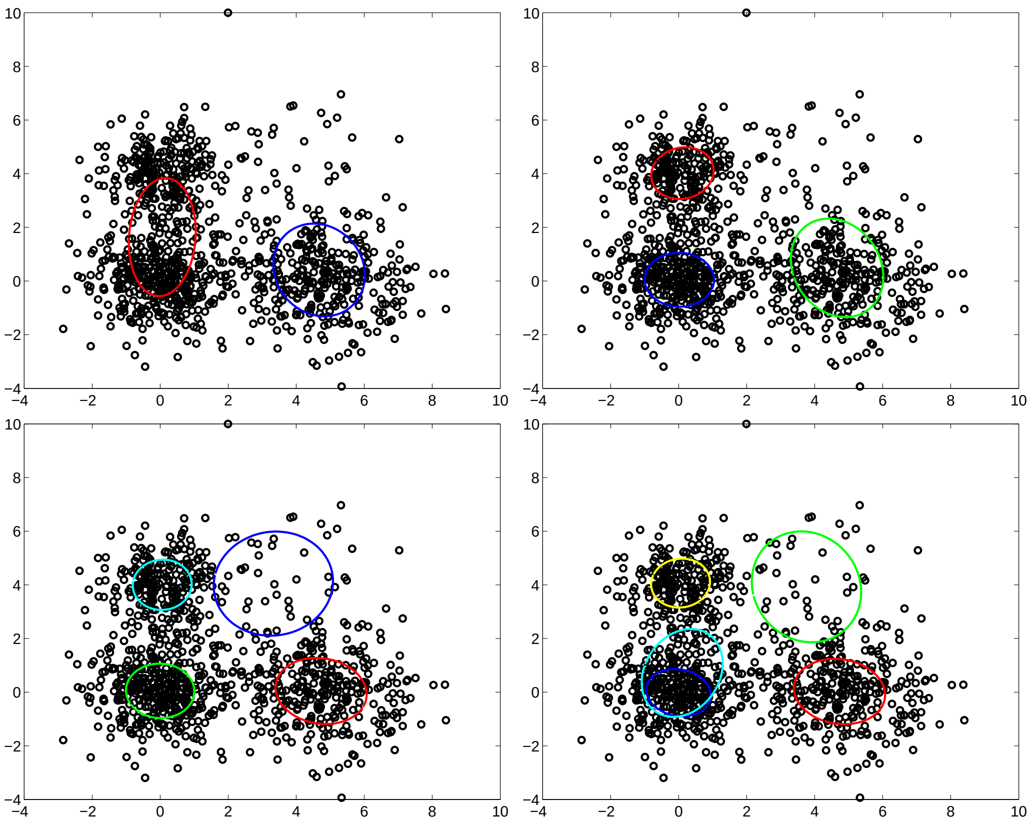

Figure 3: Plots for (2-4) of EM estimated Gaussian mixture models for $m = 2, 3, 4, 5$ (resp. top-left, top-right, bottom-left and bottom-right).

```
for t = 1:100
  % Fill in the k-means updates here
  % assign each input to the nearest cluster
  for i=1:n
    d = zeros(k,1);
    for j=1:k
      d(j) = norm(X(i,:)-centers(j,:));
    end
    [dmin,y(i)] = min(d);
  end
  % recompute the cluster means
  for j=1:k
    Ij = find(y==j);
    if (length(Ij))
       centers(j,:) = mean(X(Ij,:));
    end
  end
end
```

Here is our script `hw5prob3_1.m`:

```
clear;
close all;
load clustdata.mat;

X=X1;
for k=2:5
  y = kmeans(X,k);
  plotclust(X,y);
  eval(sprintf('print -depsc hw5prob3_1_X1_%d.eps',k));
end

X=X2;
for k=2:5
  y = kmeans(X,k);
  plotclust(X,y);
  eval(sprintf('print -depsc hw5prob3_1_X2_%d.eps',k));
end

X=X3;
for k=2:5
  y = kmeans(X,k);
  plotclust(X,y);
  eval(sprintf('print -depsc hw5prob3_1_X3_%d.eps',k));
end
```

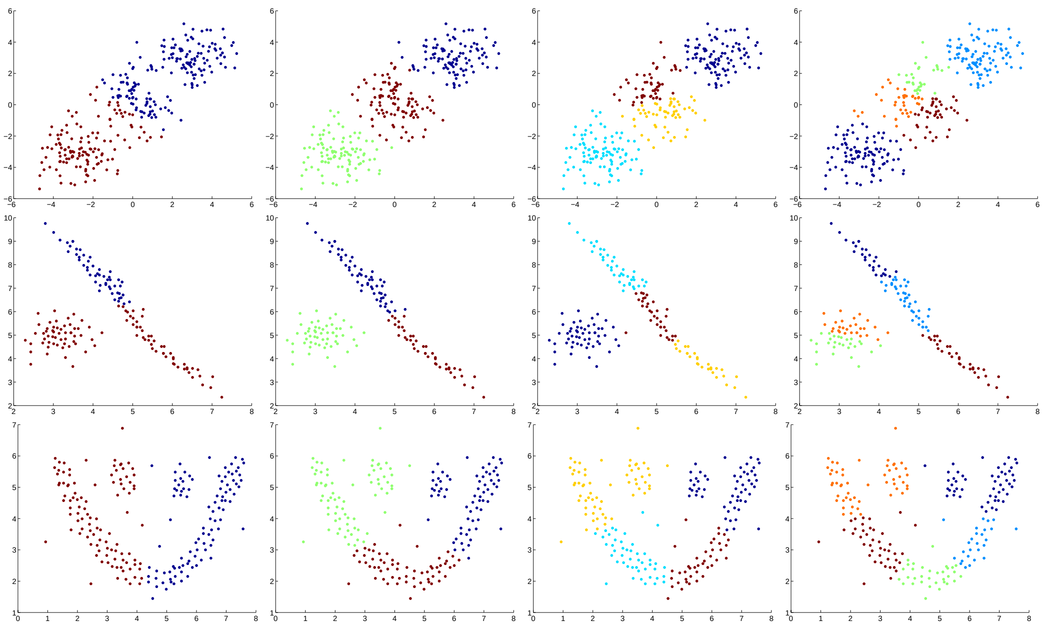
Figure 4: Plots for problem (3-1) for $k = 2, 3, 4, 5$ (left to right) and $X1, X2, X3$ (top to bottom).

The plots generated by this script are shown in Figure 4.

(3-2) (5pts) To choose between the various possible stable points of the k-means algorithm, we should select the clustering which minimizes the average squared distance of each point from its associated mean:

$$J(a, \mu) = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i - \mu_{a(i)}\|^2 \tag{56}$$

Where μ_j for $j = 1, \ldots, k$ are the cluster means and $a : \{1, \ldots, n\} \to \{1, \ldots, k\}$ assigns samples to clusters. Note that the "e-step" of k-means assigns each point i to cluster $a(i)$ so as to minize $J(a, \mu)$ for fixed means μ while the "m-step" of k-means resets the means μ so as to minimize $J(a, \mu)$ subject to fixed associations a.

Here is our code to compute this metric:

```
% kmeans_metric.m

function J = kmeans_metric(y,X)

[n,d] = size(X);
k = max(y);

J = 0.0;
for j=1:k
  Ij = find(y==j);
  mu_j = mean(X(Ij,:));
  for i=1:n
    del = (X(i,:) - mu_j)';
    J = J + del'*del;
  end
end
```

Here is our script for this problem:

```
% hw5prob3_2.m

clear;
close all;
load clustdata.mat;

X=X1;
for k=2:5
  J_min = +inf;
  for trial=1:5
    y = kmeans(X,k);
    J = kmeans_metric(y,X);
    if (J < J_min)
      J_min = J;
      plotclust(X,y);
      refresh;
      eval(sprintf('print -depsc hw5prob3_2_X1_%d.eps',k));
    end
  end
end


X=X2;
for k=2:5
  J_min = +inf;
  for trial=1:5
    y = kmeans(X,k);
    J = kmeans_metric(y,X);
    if (J < J_min)
      J_min = J;
      plotclust(X,y);
      refresh;
      eval(sprintf('print -depsc hw5prob3_2_X2_%d.eps',k));
    end
  end
end
```

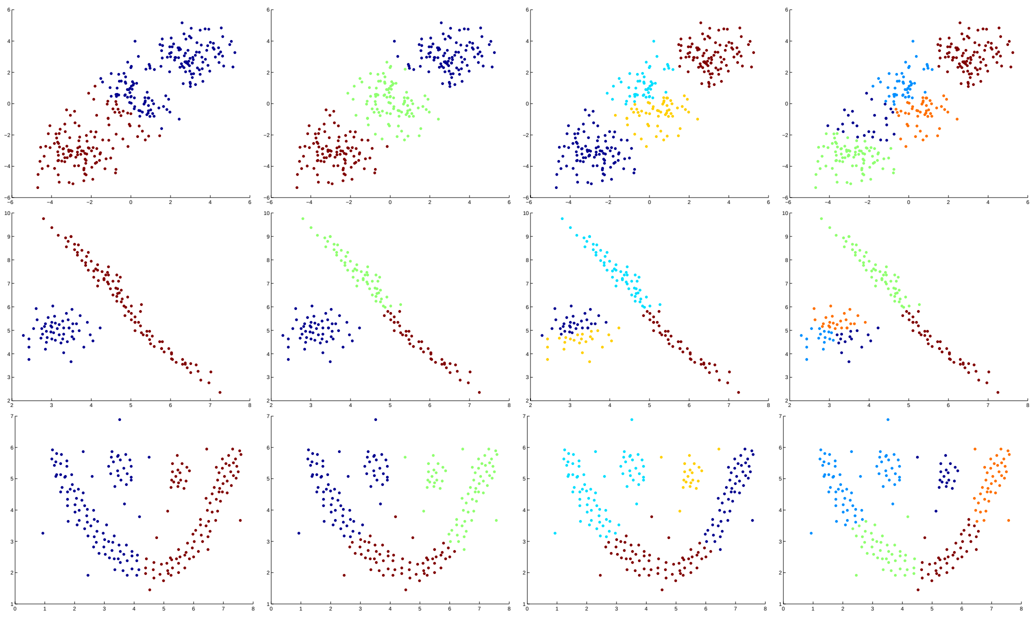

Figure 5: Plots for problem (3-2) for $k = 2, 3, 4, 5$ (left to right) and $X1, X2, X3$ (top to bottom).

```
X=X3;
for k=2:5
  J_min = +inf;
  for trial=1:5
    y = kmeans(X,k);
    J = kmeans_metric(y,X);
    if (J < J_min)
      J_min = J;
      plotclust(X,y);
      refresh;
      eval(sprintf('print -depsc hw5prob3_2_X3_%d.eps',k));
    end
  end
end
```

The plots generated by this script are shown in Figure 5.

(3-3) (5pts) It probably makes more sense to run k-means many times (for fewer iterations) rather then fewer times (for many iterations) since this should give us a coarse estimate of the global minimum rather than a precise estimate of a local minimum.

(3-4) (5pts) No, this metric would always favor higher values of k as we can always decrease J by adding more clusters. For instance, let $k = n$ and set $\mu_j = \mathbf{x}_j$ for $j = 1, \ldots, n$ so that $J = 0$. This would certainly tend to overfit the data.

(3-5) (5pts) Run EM to generate a joint Gaussian mixture model for (x, y) with $y = 1, \ldots, k$. Then, for each sample x, estimate $\hat{y}(x) = \arg\max_y P(y|x)$. These estimates then produce a clustering of the input samples.

(3-6) (5pts) First, we show that the vector $\pi = (1, \ldots, 1)'$ is an eigenvector of the transittion probability matrix P, with entries $P_{ij} = P(j|i)$, and has eigenvalue $\lambda = 1$, i.e. $P \cdot \pi = \pi$.

$$\begin{aligned} (P \cdot \pi)_i &= \sum_i P_{ij} \pi_j & (57) \\ &= \sum_i P_{ij} 1 & (58) \\ &= \sum_i P(j|i) & (59) \\ &= 1 & (60) \\ &= \pi_i & (61) \end{aligned}$$

Hence, $P \cdot \pi = \lambda\pi$ with $\lambda = 1$ as was to be shown.

Now, we argue that $\lambda = 1$ must be the largest eigenvalue. Suppose there existed a vector π s.t. $P \cdot \pi = \lambda\pi$ with $\lambda > 1.0$. Then, the t-step transition probability matrix P^t must have some eigenvalues going to infinity as t becomes large. But this violates P^t being a transition probability matrix (with entries between 0 and 1). Hence, $\lambda = 1$ is the maximum eigenvalue of P.

Moreover, the symmetric matrix considered in spectral clustering is similar to P and hence has the same eigenvalues as P (with maximum eigenvalue 1 as claimed in lecture).

(3-7) (10pts) Here is the code we modified in `spectral.m`:

```
% modified code...
if (k<2)
    error('Only works for k>=2')
end

...

% removed code...
% V = V(:,I(end-1)); % eigenvector corresp. to second largest eigenvalue
% y = (3+sign(V))./2;

% added code...
Xk = V(:,[2:k]);
J_min = +inf;
y=[];
for trial=1:5
  y_trial = kmeans(Xk,k);
  J = kmeans_metric(y,Xk);
  if (J < J_min)
```

```
    J_min = J;
    y = y_trial;
  end
end
```

I got good results by using $r = 4$ nearest neighbors and setting $\beta = 0.1$ (roughly the inter-point distance between nearest neighbors). Here is our automated script for this problem:

```
% hw5prob3_7.m

clear;
close all;
load clustdata.mat;

r=4;
beta=0.1;

X=X1;
for k=2:5
  y = spectral(X,k,r,beta);
  refresh;
  eval(sprintf('print -depsc hw5prob3_7_X1_%d.eps',k));
end

X=X2;
for k=2:5
  y = spectral(X,k,r,beta);
  refresh;
  eval(sprintf('print -depsc hw5prob3_7_X2_%d.eps',k));
end

X=X3;
for k=2:5
  y = spectral(X,k,r,beta);
  refresh;
  eval(sprintf('print -depsc hw5prob3_7_X3_%d.eps',k));
end
```

The plots generated by this script are shown in Figure 6.

(3-8) (5pts) We need to recompute the means $\mu_j = \frac{1}{n_j}\sum_{i:a(i)=j} \mathbf{x}_i$ where $n_j = |\{x_i : a(i) = j\}|$ at each iteration of the k-means algorithm. We can't recover these from just the inter-point distances $D_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$ for all i, j.

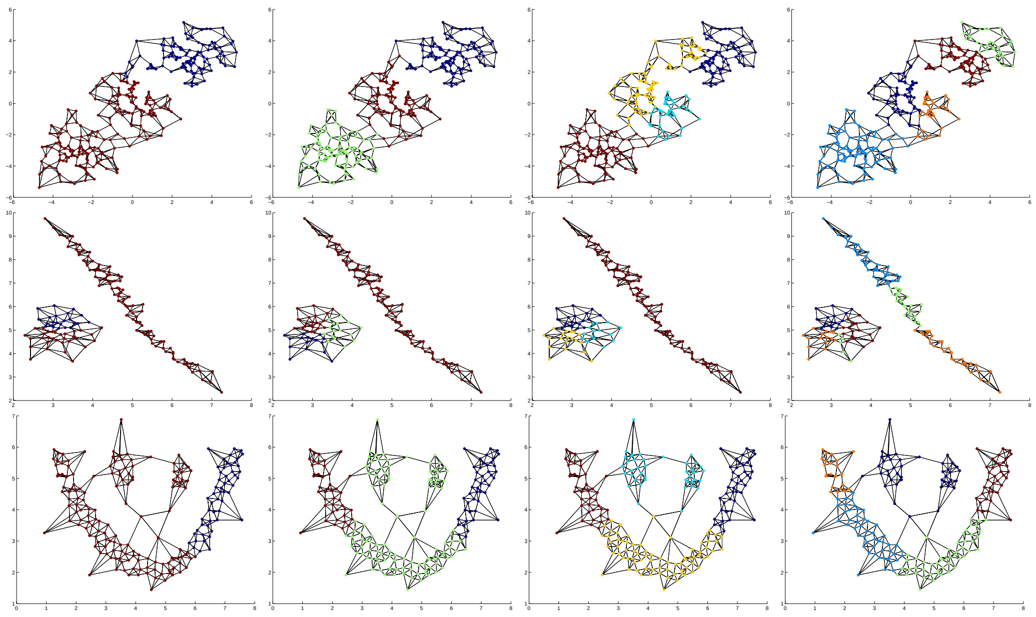

Figure 6: Plots for problem (3-7) for $k = 2, 3, 4, 5$ (left to right) and $X1, X2, X3$ (top to bottom).