10701/15781 Machine Learning, Spring 2006: Homework 5

Due: Wednesday, April 19, beginning of the class.

Please refer your questions to TAs.

1 [10 points] K-Means [Andreas]

1. Consider the data set in Figure 1. The '+' symbols indicate data points, the (centers of the) squares A, B, C indicate the current cluster centers. Please show the progress of the K-Means algorithm by showing how the class centers move with each iteration until convergence. To do this, use the remaining figures, and for each iteration, indicate which data points will be associated with each of the clusters, as well as the updated class centers. Show your construction. If during the cluster update step, a cluster center has no points associated with it, it will not move. Use as many figures you need until convergence of the algorithm.

2. Now repeat the same construction for the data set in Figure 2.

3. What does this imply about the behavior of the K-Means algorithm?

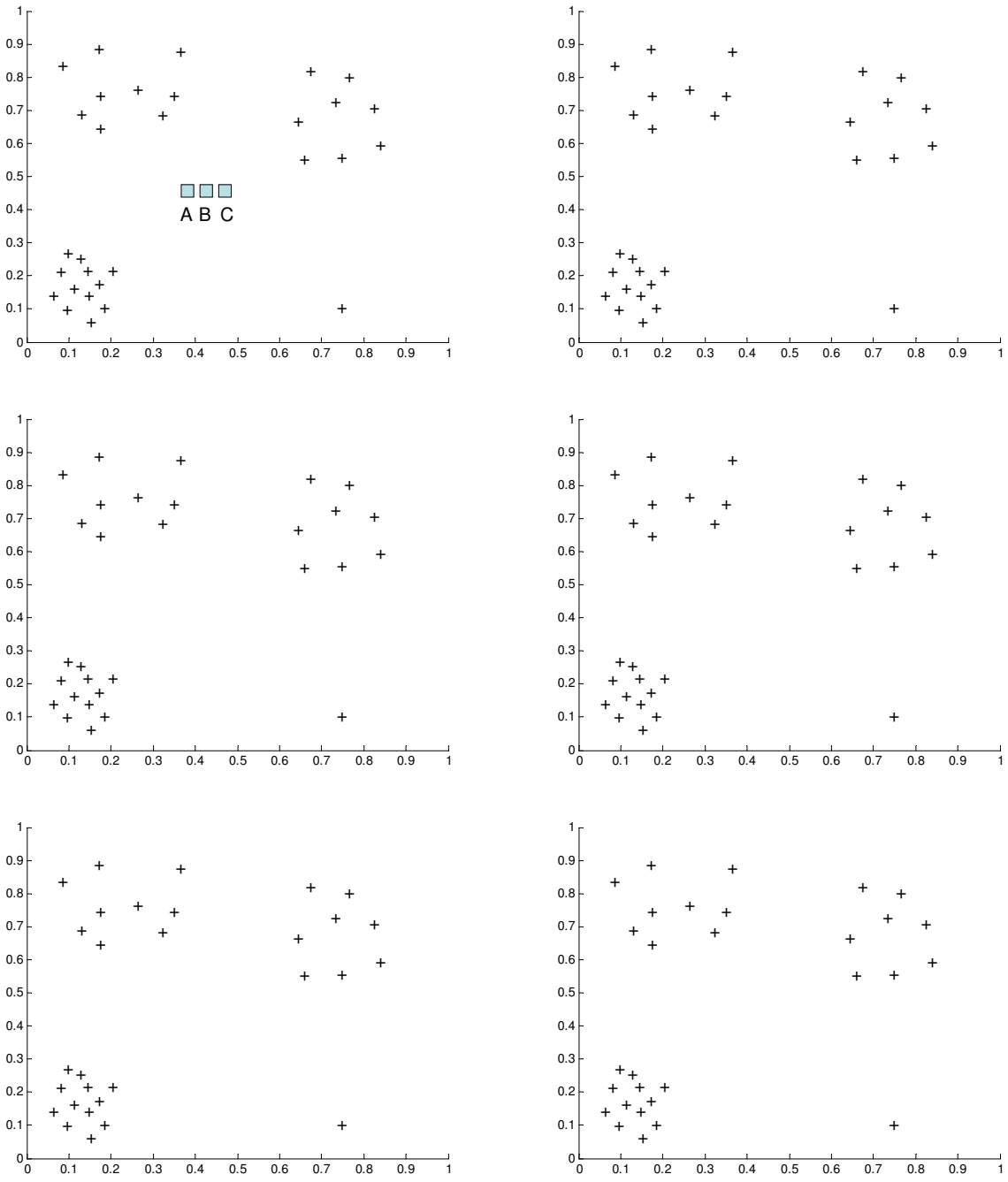

Figure 1: K-Means data set 1.

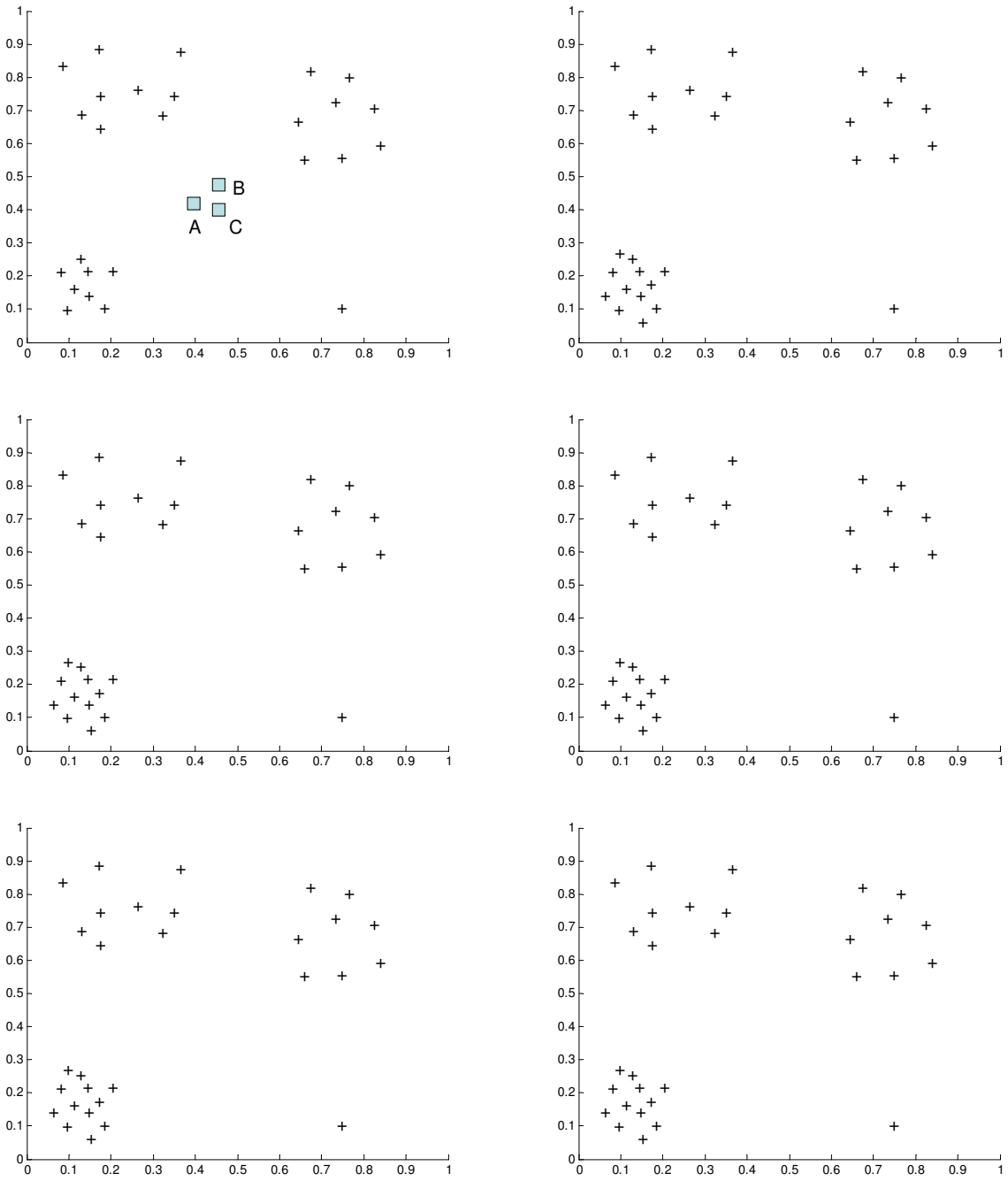

Figure 2: K-Means data set 2.

2 [40 points] HMMs and the Baum-Welch algorithm [Jure]

In this question, you will implement the Baum-Welch algorithm for learning the parameters of a Hidden Markov Model when the state path (a sequence of hidden states) is unknown.

You will implement Baum-Welch for the case of *semi-supervised learning*. First you will use a

small set of labeled examples to get the "prior" model parameters (pseudo-counts). You will then be given a set of unlabeled training examples over which you will run the Baum-Welch algorithm. Incorporating evidence from unlabeled and labeled data, you will learn a model and observe how the performance changes as you increase the amount of unlabeled data. The hope is that by combining the unlabeled data with the labeled data, the overall classification accuracy will improve. As in homework 4, you will work with the character recognition dataset.

For all implementation questions, please submit your source code to

`/afs/andrew.cmu.edu/course/10/701/Submit/your_andrew_id/HW5/`

and provide pseudo-code in your answers. The support code and the dataset for this question are available on the class website.

2.1 Baum-Welch (EM algorithm for HMMs)

When fitting a model we generally want to choose parameters that will maximize the likelihood of our data. Expectation Maximization (EM) algorithm works by guessing initial parameter values, then estimating the likelihood of the data under the current parameters. These likelihoods can then be used to re-estimate the parameters, iteratively until a local maximum is reached.

Here we give a brief introduction to EM for HMMs. First we define two hidden variables. Given training example n (a word in our case), $\xi_{n,t}(i,j)$ is defined as the probability of being in state i at time index t and in state j at time index $t+1$:

$$\xi_{n,t}(i,j) = P(x_{n,t} = i, x_{n,t+1} = j|O)$$

and $\xi_{n,t}(i)$ is the posteriori probability

$$\gamma_{n,t}(i) = P(x_{n,t} = i|O)$$

where O is the sequence of observations (in our case images of letters).

In the E-step of the EM algorithm, we calculate the distribution of hidden variables $\xi_{n,t}(i,j)$ and $\gamma_{n,t}(i)$:

$$\xi_{n,t}(i,j) = \frac{\alpha_{n,t}(i)a_{i,j}b_j(o_{n,t+1})\beta_{n,t+1}(j)}{P(O)}$$

and

$$\gamma_{n,t}(i) = \frac{\alpha_{n,t}(i)\beta_{n,t}(i)}{\sum_i \alpha_{n,t}(i)\beta_{n,t}(i)}$$

where $\alpha_{n,t}$ and $\beta_{n,t}$ are the factors from Forward-Backward algorithm. $a_{i,j}$ and $b_j(o_{n,t+1})$ are the parameters of the model: $a_{i,j}$ is the probability of making a transition from state i to state j; $o_{n,t+1}$ is a $t+1$-th letter of n-th word; and $b_j(o_{n,t+1})$ is the probability of emitting letter $o_{n,t+1}$ if we are in state j.

In the M step we re-estimate the model parameters: the starting state distribution, $P(X_0 = i) = \pi(i)$:

$$\pi(i) = \frac{\sum_n^N \gamma_{n,1}(i)}{N}, \tag{1}$$

and the transition probabilities:

$$a_{i,j} = \frac{\sum_n^N \sum_{t=1}^{T-1} \xi_{n,t}(i,j)}{\sum_n^N \sum_{t=1}^{T-1} \gamma_{n,t}(i)} \tag{2}$$

Note that we should also update the emission probabilities $b_j(o_t)$. To make your life easier (and implementation less messy) we will assume $b_j(o_t)$ stay fixed and we don't update them.

2.2 Dataset and the code

From the class website download the code and the dataset.

You will be using the same OCR data set dataset as in homework 4. It consists of a sequence of words, one character per row.[1]

The file `ocrtable.mat` contains the OCR data. The columns have the following meaning:

Col. 1: character ID (same as the row number)

Col. 2: character code (1-26), 1='a', 2='b', etc.

Col. 3: the code of the previous character or -1 if this is the first character in a word)

Col. 4: word id

Col. 5: the position of the character in the word

Col. 6: cross-validation fold (ignore)

Cols. 7-70: pixel value (0/1).

We also give you a complete set of functions to work with factors (basically, the functionality you implemented in homework 4, optimized for speed):

- `table_factor`: creates a new table factor.
- `assignment`: creates a new assignment structure (representing an assignment for a set of variables). An assignment consists of an array of variable IDs and the corresponding values.
- `value`: Given an assignment that includes the factor's arguments, returns the value of the factor for this assignment of variables.
- `restrict`: Fixes one or more arguments of the factor to a particular value. We have seen this operation in the example above: for example, if a factor $\psi_1(X, Y)$ represents the joint distribution $p(X, Y)$, restricting ψ_1 to $Y = 0$ returns a factor $\psi_2(X)$ that corresponds to the distribution $p(X, Y = 0)$.
- `multiply`: given two table factors f_1 and f_2 over two sets of variables $\mathbf{U}$ and $\mathbf{V}$, respectively, returns a table factor over $\mathbf{U} \cup \mathbf{V}$ that represents the product $f_1 \times f_2$.
- `maximum`: given a factor and a variable X in the factor's scope, computes the maximum over all variables, other than X.
- `argmax`: given a factor f over a single variable V, returns the value of V that maximizes f. Note that function takes the variable that should be left in the factor, as opposed to the variable that should be maximized-out.

Besides this we also provide the following functions to work with HMMs:

- `split_data`: gives a split of the data into 3 sets: labeled, training, test. Supply it the table loaded from `ocrtable.mat` file.

[1] This dataset is a modified version of the dataset at `http://ai.stanford.edu/~btaskar/ocr/`, which has been sub-sampled and slightly altered to simplify the processing.

- `hmm_init`: initializes the HMM model to random parameters (you will need this for Baum-Welch). It also initializes emission probabilities to the true values. As a parameter give it the table loaded from `ocrtable.mat` file.
- `hmm_train`: trains a HMM model on the *labeled* data.
- `hmm_eval`: calculates accuracy of the HMM model.
- `hmm_fwdBck`: Forward-Backward algorithm.
- `get_counts`: Returns the counts for the prior and transitions of HMM model.

2.3 Questions

1. Implement Baum-Welch algorithm for learning parameters of the HMM. Use function `split_data` to get a split of the data in 3 sets: *labeled, train* and *test*. By using function `get_counts` obtain pseudo-counts of the *labeled* set.

   Now implement the EM part of the algorithm. Use the *training* part of the data to run the algorithm. Use `hmm_init` to initialize the model. To obtain αs and βs in E-step use the Forward-Backward algorithm (function `hmm_fwdBck`). Since some of the hidden variables are labeled, we are in a semi-supervised learning setting. Hence, in the M-step, when updating the model parameters, both the pseudo-counts you estimated above, as well as the true counts which you obtained on the *labeled* set, have to be included.

   Turn in the pseudo-code.

2. How does this change the formulas (1) and (2) for estimating $\pi(i)$ and $a_{i,j}$? Please give the equations.

3. What is the formula for computing the log-likelihood of the observed data $\log P(O)$?

4. For one of the splits of the data run the EM algorithm and plot the log-likelihood of the training data $\log P(O)$ as a function of the number of iterations of the EM algorithm. Run the EM algorithm for at least 10 steps.

5. Plot the accuracy of the final HMM (use function `hmm_eval`) on labeled, train and test sets as you increase the amount of unlabeled training data (use function `split_data`). Vary the parameter of the `split_data` function which will produce splits with various amounts of unlabeled data (number of words). Vary the size of train data from 10 to 100 in steps of 10 and plot the model accuracy on all 3 sets.

   What do you observe from the plot? What happens to the accuracy on the labeled set? What with the accuracy on the training and test sets?

3 [25 points] Bayes Net structure learning [Stano]

3.1 Mutual information

Consider the following dataset with four binary variables A, B, C, D:

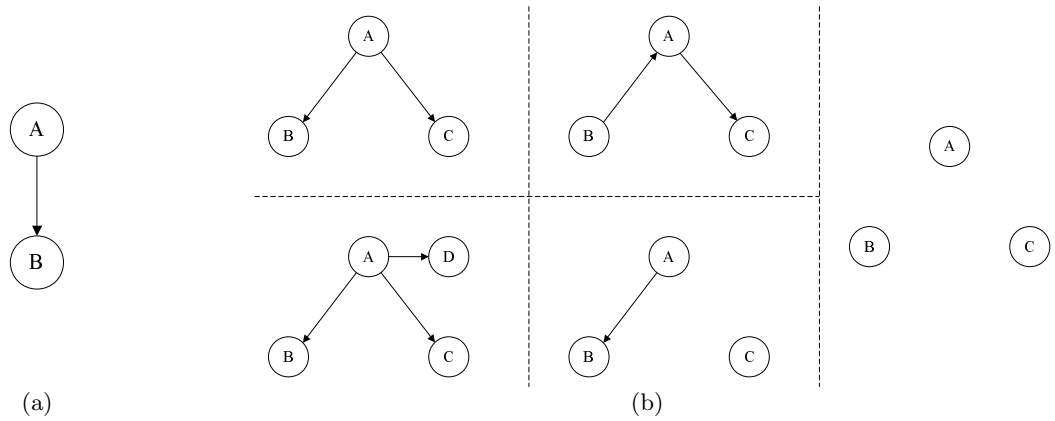

Figure 3: Bayesian networks for Question 3.2.1.

A	B	C	D
1	0	1	0
1	0	1	1
1	0	0	1
0	1	0	1
1	0	1	0

Your goal is to learn a Bayesian network structure for this data set.

1. Compute the mutual information $I(X, Y)$ for each pair of variables X, Y.
2. Using the computed mutual informations, find a *tree* Bayesian network that maximizes the likelihood of the training data.
3. Find a Bayesian network $\mathcal{G}$ that maximizes the likelihood of the training data s.t. at most one node in $\mathcal{G}$ has two parents and the other nodes have one or no parents. Justify your answer. *Hint: you do* **not** *need to compute the log-likelihood of data exhaustively for all possible BN structures.*

3.2 Learning forests with BIC score

In this part, you will derive a variation of the Chow-Liu algorithm for learning Bayesian networks with forest graph structure using the BIC score.

1. Recall that the BIC score is

$$\text{score}_{\text{BIC}}(\mathcal{G}; \mathcal{D}) = \log p(\mathcal{D} \mid \mathcal{G}, \theta_{\mathcal{G}}) - \frac{\text{NumberParams}(\mathcal{G})}{2} \log M,$$

   where $\mathcal{G}$ is a Bayesian network, $\mathcal{D}$ is the dataset, $\theta_{\mathcal{G}}$ are the parameters that represent the CPDs in $\mathcal{G}$, and NumberParams($\mathcal{G}$) is the total number of independent parameters in the CPDs of $\mathcal{G}$. For example, for the Bayesian network in Figure 3(a), the number of parameters is $N_A - 1 + N_A(N_B - 1) = N_A N_B - 1$, where $N_X = |\text{Val}[X]|$ is the number of possible values for variable X.

   Write down the number of independent parameters in the CPDs for each Bayesian network in Figure 3(b), in terms of N_A, N_B, N_C, and N_D. Show your work.

2. A forest is a graph that consists of one or more connected components, each of which is a tree (but not a poly-tree, i.e., each node has at most one parent). Prove that the number of independent parameters for any forest $\mathcal{G}$ can be written as

$$\text{NumberParams}(\mathcal{G}) = \sum_{\{X,Y\} \in \text{E}_\mathcal{G}} (N_X - 1)(N_Y - 1) + \left(\sum_{X \in \text{V}_\mathcal{G}} N_X \right) - |\text{V}_\mathcal{G}|. \tag{3}$$

Here, $\text{E}_\mathcal{G}$ are the undirected edges of $\mathcal{G}$ and $\text{V}_\mathcal{G}$ are its vertices (variables).

Hint: In a forest $\mathcal{G}$, $|V_\mathcal{G}| = |E_\mathcal{G}| + \#components_G$, where $\#components_G$ is the number of connected components in $\mathcal{G}$.

3. Using (3), describe an efficient algorithm for learning a forest Bayesian network that maximizes the BIC score. (*Hint: This algorithm is a small modification of Chow Liu.*)

4 [25 points] EM for supervised learning [Anton]

In class you have learned about EM in an unsupervised learning setting. It turns out that EM is also sometimes useful for supervised learning problems. Suppose you have a classification problem with a binary class $Y \in \{1, 2\}$ and one feature $X \in \mathbb{R}$.

The true model, from which the data is generated, is a *mixture model*: given the value of class $Y = y$, X is drawn from a normal distribution $N(\mu_{y,a}, \sigma_{y,a})$ with probability β_y or from $N(\mu_{y,b}, \sigma_{y,b})$ with probability $(1 - \beta_y)$. In other words, X is drawn from a mixture of 2 Gaussians, and the parameters of the mixture depend on the class:

$$\begin{aligned} P(Y = 1) = \alpha, \quad P(Y = 2) = 1 - \alpha \\ p(X|Y = 1) \sim \beta_1 N(\mu_{1,a}, \sigma_{1,a}) + (1 - \beta_1) N(\mu_{1,b}, \sigma_{1,b}) \\ p(X|Y = 2) \sim \beta_2 N(\mu_{2,a}, \sigma_{2,a}) + (1 - \beta_2) N(\mu_{2,b}, \sigma_{2,b}) \end{aligned} \tag{4}$$

1. You are given a dataset $\mathcal{D} =< X_n, Y_n >, n = 1 \dots N$ generated by the mixture model in Equation 4. However, you decide that you want to learn a simple Gaussian Naive Bayes classifier:

$$\begin{aligned} P(Y = 1) = \alpha, \quad P(Y = 2) = 1 - \alpha \\ p(X|Y = 1) = N(\mu_1, \sigma_1) \\ p(X|Y = 2) = N(\mu_2, \sigma_2) \end{aligned} \tag{5}$$

What are the MLE parameter values of this GNB classifier in terms of the dataset?

2. Now you will derive an EM algorithm for this supervised learning problem. Suppose you know that $\sigma_{i,j} = 1 \; \forall i, j$ and need to learn $\alpha, \beta_i, \mu_{i,j}$. You have a dataset $\mathcal{D} =< X_n, Y_n >, n = 1 \dots N$. Derive E and M steps for the unknown parameters.

Hint: To derive EM updates, it might be useful to introduce a hidden variable $Z \in \{a, b\}$ that denotes the component of the Gaussian mixture from which a given X has been drawn. Then the distribution from Equation 4, represented by a BN in Fig. 4(a) can be seen as a marginalization of a distribution represented by BN Fig. 4(b):

$$p(X, Y) = \sum_z p(X, Y, Z = z).$$

Figure 4: Bayesian networks for Question 4.2.

Instead of learning parameters of the original distribution directly, you will then learn parameters of BN in Fig. 4(b), where

$$P(Y = 1) = \alpha, \quad P(Y = 2) = 1 - \alpha$$
$$P(Z = 1|Y) = \beta_Y$$
$$p(X|Y, Z) = N(\mu_{Y,Z}, 1).$$

3. Now consider the data set in Figure 5. Filled circles correspond to data points of class 1, empty circles correspond to data points of class 2. In Figure 5, sketch class-conditional densities $p(X|Y)$: First, sketch the class-conditional density using the Gaussian Naive Bayes model from part 1. Then sketch the class density using the class-conditional mixture model estimated using the EM algorithm in part 2, assuming that EM finds the globally optimum solution. Which model fits this data better?

4. Construct a one-dimensional data set, similar to the one in Figure 5, with the following property: if we train a Gaussian Naive Bayes classifier on this dataset and then use the same data for testing, all data points from one class would be misclassified, but for the class-conditional mixture model from part 2, the training error on this dataset would be 0.

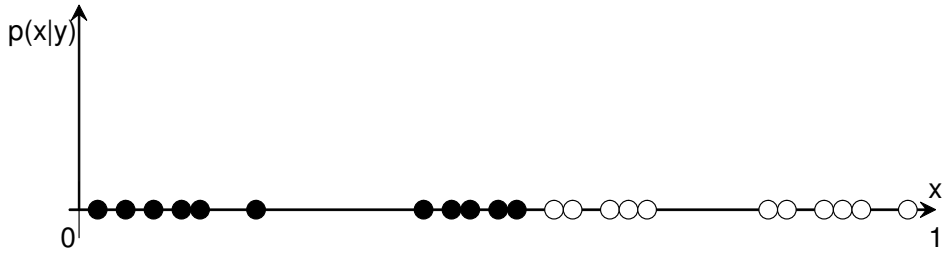

Figure 5: EM data set for Question 4.3.