10-701 Final Exam, Spring 2006

1. Write your name and your email address below.
   - Name:
   - Andrew account:
2. There should be 22 numbered pages in this exam (including this cover sheet).
3. You may use any and all books, papers, and notes that you brought to the exam, but not materials brought by nearby students. Calculators are allowed, but no laptops, PDAs, or Internet access.
4. If you need more room to work out your answer to a question, use the back of the page and clearly mark on the front of the page if we are to look at what's on the back.
5. Work efficiently. Some questions are easier, some more difficult. Be sure to give yourself time to answer all of the easy ones, and avoid getting bogged down in the more difficult ones before you have answered the easier ones.
6. Note there are extra-credit sub-questions. The grade curve will be made without considering students' extra credit points. The extra credit will then be used to try to bump your grade up without affecting anyone else's grade.
7. You have 180 minutes.
8. Good luck!

Question	Topic	Max. score	Score
1	Short questions	6 + 3.6 extra	
2	Gaussian Naive Bayes and Logistic Regression	8	
3	Boosting	12	
4	HMM	12	
5	Bayesian Networks and Independence	10	
6	Bayesian Network Structure Learning	10	
7	MDPs and Reinforcement Learning	12 + 6 extra	
8	EM Algorithm	12	
9	Dimensionality reduction - PCA	8	
10	Dimensionality reduction - neural nets and PCA	10	

1 [6 points + 3.6 extra credit] Short questions

1. [3 points] In Figure **??**, draw a partially labeled data set, for which transductive SVMs achieve higher classification accuracy than regular SVMs. Use linear kernels. To do that, mark the positive labels by +, negative labels by -, indicate the true decision boundary between the classes, as well as the decision boundaries computed by the regular and transductive SVMs.

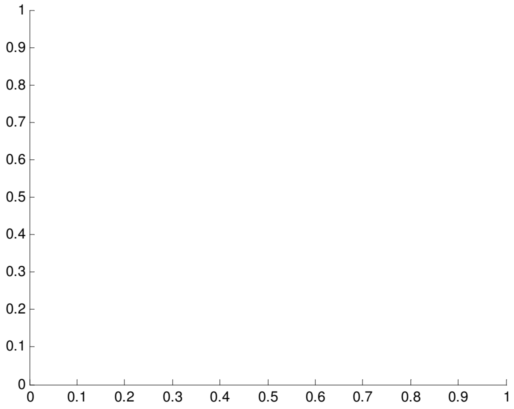


Figure 1: Please draw the data set for Question 1.1 here.

2. [3 points] Distribute the numbers 1,2,3,4 to the following four methods for classifying 2D data, such that 1 indicates highest variance and lowest bias, and 4 indicates lowest variance and highest bias.

   - Boosted logistic regression (10 rounds of boosting)
   - 1-nearest neighbor classifier
   - Decision trees of depth 10
   - Logistic regression

3. [3 points extra credit] Consider two random variables, X_1 and X_2. X_1 can take values A, B, C, D, E, and X_2 can take values $1, 2, 3, 4, 5$. Data sets of samples from (X_1, X_2) consist of combinations of assignments to X_1 and X_2, such as e.g., $(A, 3)$ or $(B, 5)$. Consider the bipartite graphs (with no edges yet) drawn below. A data set can be visualized by the presence of a set of edges between X_1 and X_2.

   On the graphs below, draw a data set (i.e., a set of edges) of at most 10 elements, such that, when using co-training, a single labeled example will lead to (left) no improved classification, or (right) perfectly classified data (on all possible combinations of (X_1, X_2)). An edge determines whether a particular combination of features are present in the data.

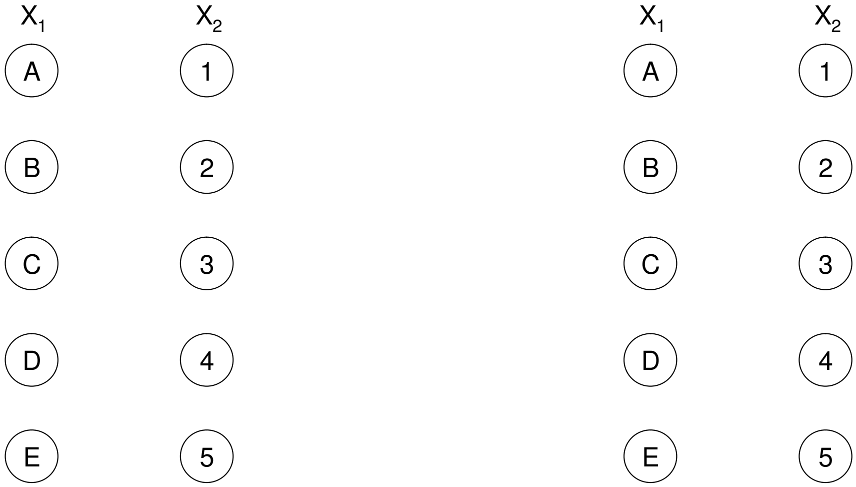


Figure 2: Please answer the cotraining question here.

4. [0.6 points extra credit] Connect the dots to get a prediction of your grade:

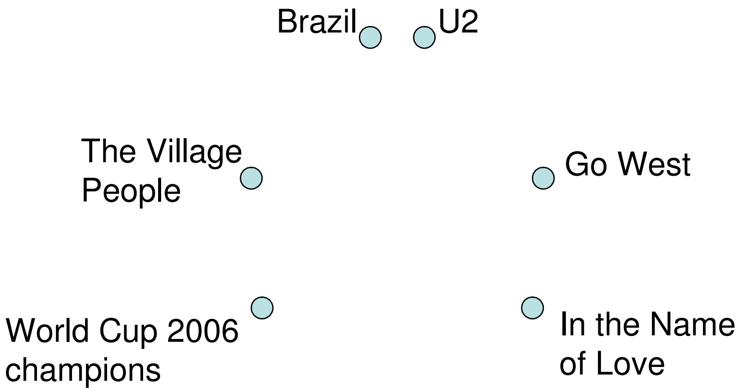


Figure 3: Connect the dots.

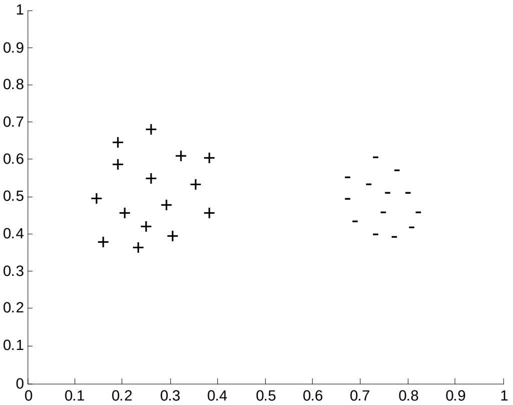
2 [8 points] Gaussian Naive Bayes and Logistic Regression

In this question you will compare the decision boundaries of Gaussian Naive Bayes (GNB) and Logistic Regression (LR). For each of the datasets circle true (**T**) or false (**F**), based on whether the method can separate the two classes.

• If the answer for Logistic Regression is true, draw the decision boundary.

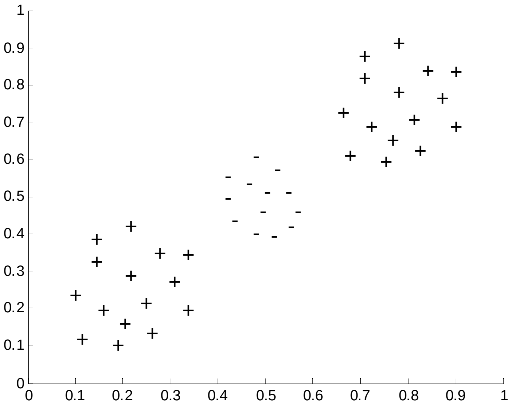
• If the answer for Gaussian Naive Bayes is true, draw the shape of the two Gaussian bumps (center and the variance).

• If the answer for any of Gaussian Naive Bayes or Logistic Regression is false, please give a one sentence explanation.



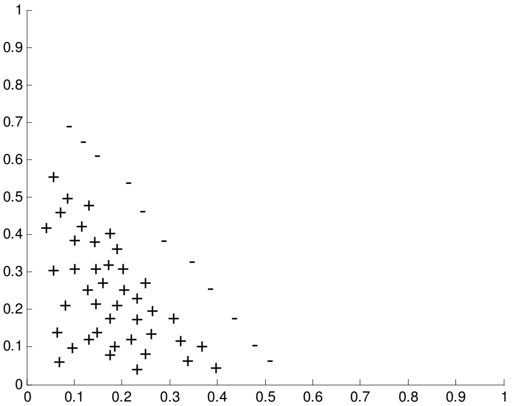
GNB can separate: T F

LR can separate: T F



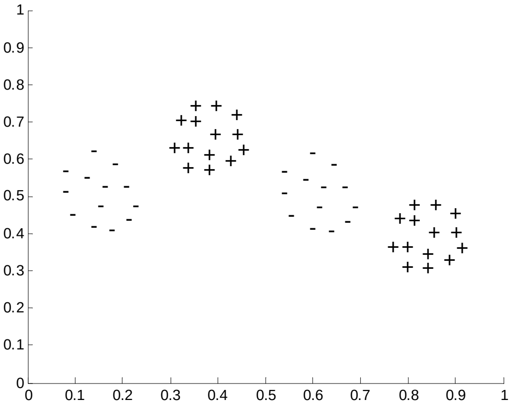
GNB can separate: T F

LR can separate: T F



GNB can separate: T F

LR can separate: T F



GNB can separate : T F

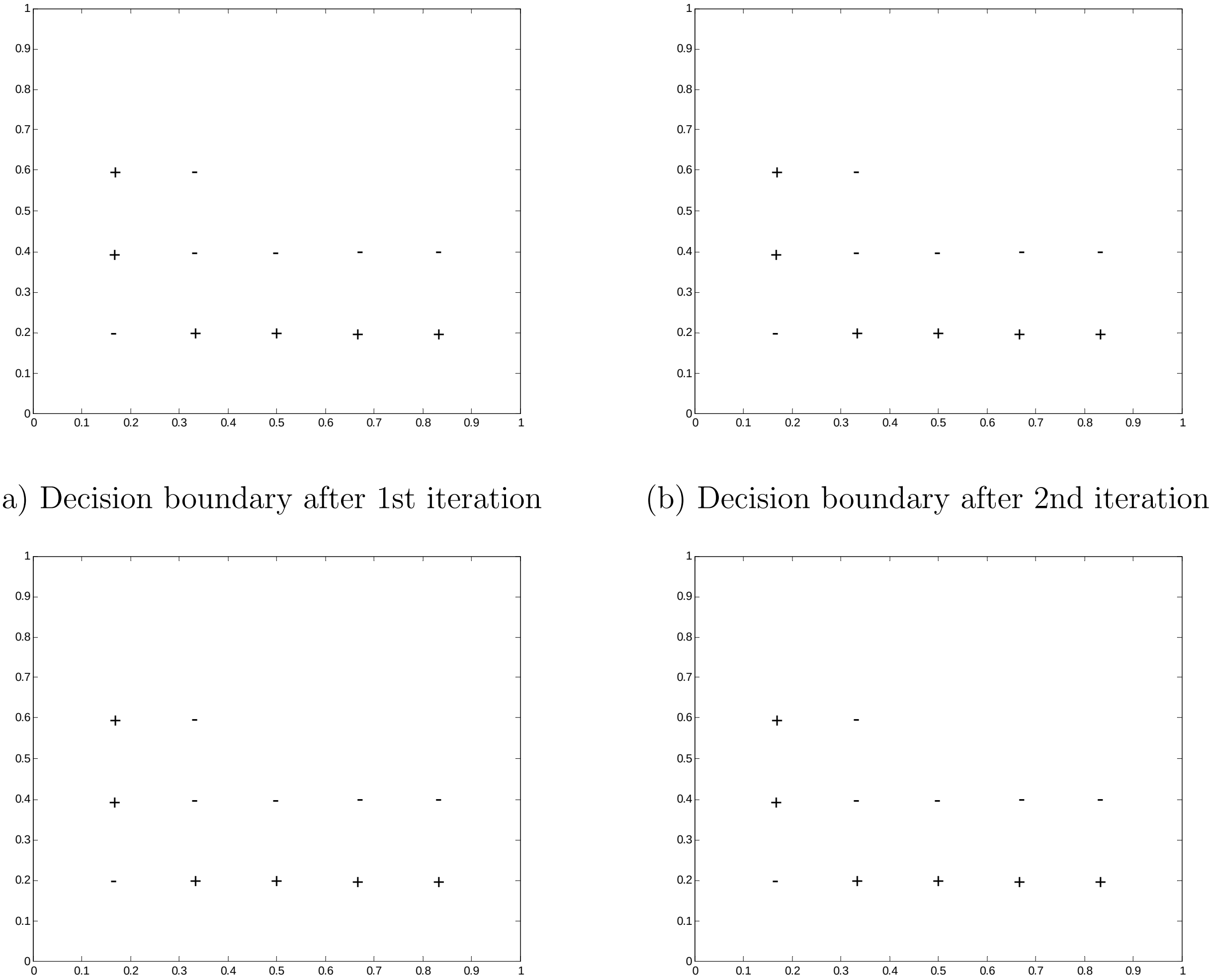
LR can separate: T F

3 [12 points] Boosting

Given a small dataset on figure **??** we want to learn a classifier that will separate pluses from minuses, using the AdaBoost algorithm. Each datapoint x_i has a class $y_i \in \{+1. -1\}$.

We will be using weak learners where the weak learner's decision boundary is *always parallel* to one of the axis, i.e., the separating plane is either vertical or horizontal. You can think of weak learners as decision stumps where we split either on X or Y coordinate.

When training the new weak learner $h_t(x)$, we will choose the one that maximizes the weighted classification accuracy with respect to the current weights D_t, i.e., choose h_t that maximizes $\sum_i D_t(i)\delta(h_t(x_i) = y_i)$. Note that $h_t(x)$ only takes values in $\{+1, -1\}$, depending on whether it classifies x as positive or negative.



(a) Decision boundary after 1st iteration

(b) Decision boundary after 2nd iteration

(c) Example with *lowest* weight

(d) Example with *highest* weight

Figure 4: Dataset for question **??** on boosting.

1. [1 point] Draw the decision boundary of boosting after the first iteration (after the first weak learner is chosen) on Figure **??**(a). Don't forget to mark which parts of the plane get classified as "+" and which as "−".

2. [1 point] Now we do second iteration of boosting. Draw decision boundaries of both weak learners in Figure **??**(b). Boosting combines the decisions of both weak learners. Mark which parts of the plane boosting with 2 weak learners classifies as "+" and which as "−". Use Figure **??**(b).

3. [2 points] In AdaBoost, we choose α_t as the weight of the t-th weak learner, where $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$. Hereby, $\varepsilon_t = P_{x\sim D_t}[h_t(x) \neq y]$, i.e. the weighted fraction of examples misclassified by the t-th weak learner. Which one is larger, α_1 or α_2? Give a one sentence explanation of your answer.

4. [2 points] After two iterations of boosting, how many training examples are misclassified?

5. [2 points] Mark which training example(s) have the *lowest* weight (D_t) after the two iterations of boosting, $t = 2$? Use Figure **??**(c).

6. [2 points] Mark which training example(s) have the *highest* weight (D_t) after the second iteration of boosting, $t = 2$? Use Figure **??**(d).

7. [2 points] Using the dataset from figure **??** and our weak learners, will Boosting ever achieve zero training error? Give a one sentence explanation of your answer.

4 [12 points] HMM

Consider the Hidden Markov Model defined by the transition and observation probabilities in the table below. The hidden variables are X_1, X_2, X_3, X_4, and the observations are O_1, O_2, O_3, O_4, i.e., the hidden and observed sequences have length 4. The hidden variables X_i can take one of six values, $\{s_1, \ldots, s_6\}$. The observations O_i are in $\{a, b, c, d\}$.

You know that $P(X_1 = s_1) = 1$, i.e., the hidden sequence starts in state s_1. The transition probabilities are as represented in the following transition diagram:

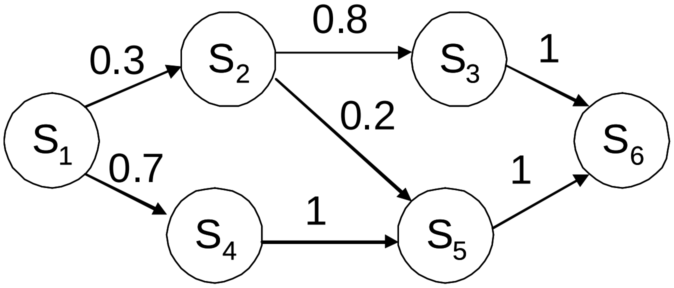


Figure 5: Finite (stochastic) state machine representation of the transitions for the HMM of Question **??**. The transition probabilities are annotated on the edges.

For example, $P(X_{t+1} = s_3 \mid X_t = s_2) = 0.8$, and $P(X_{t+1} = s_4 \mid X_t = s_1) = 0.7$. Edges which are missing correspond to transition probabilities of zero.

The observation probabilities are as follows:

	a	b	c	d
s_1	0.5	0.3	0	0.2
s_2	0.1	0.3	0.5	0.1
s_3	0.2	0.3	0.4	0.1
s_4	0.3	0.2	0.3	0.2
s_5	0	0.3	0.2	0.6
s_6	0.4	0.4	0.1	0.1

For example, $P(O_t = c \mid X_t = s_2) = 0.5$, and $P(O_t = a \mid X_t = s_4) = 0.3$. We will use the following shorthanded notation where we for example write $P(O = abca, X_2 = s_1, X_4 = s_2)$ instead of $P(O_1 = a, O_2 = b, O_3 = c, O_4 = a, X_2 = s_1, X_4 = s_2)$.

[3 points each] For each of the items below, insert <, > or = into the brackets between the left and the right expression. Show your work or justify your answer. *Hint: Thinking before computing might save a lot of time.*

1. $P(O = abca, X_1 = s_1, X_2 = s_2)$ (   ) $P(O = abca | X_1 = s_1, X_2 = s_2)$

   ★ **SOLUTION:** "$<$". The right hand side is the left hand side divided by $P(X_1 = s_1, X_2 = s_2)$ which is less than 1.

2. $P(O = abca, X_1 = s_1, X_4 = s_6)$ (   ) $P(O = abca | X_1 = s_1, X_4 = s_6)$

★ **SOLUTION:** "=". Here, $P(X_1 = s_1, X_4 = s_6) = 1$, hence we have equality.

3. $P(O = acdb, X_2 = s_2, X_3 = s_3)$ ( ) $P(O = acdb, X_2 = s_4, X_3 = s_5)$

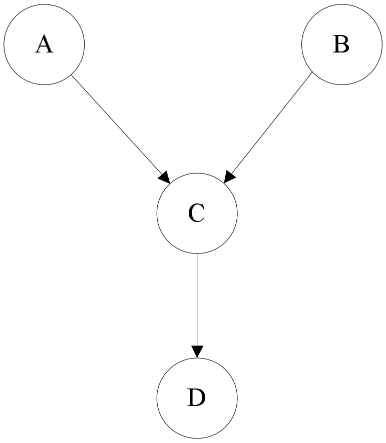
★ **SOLUTION:** "<". We work out $P(O = acdb, X_2 = s_2, X_3 = s_3) = CP(X_2 = s_2 \mid X_1 = s_1), P(X_3 = s_3 \mid X_2 = s_2)P(O_2 = c \mid S_2 = s_2)P(O_3 = d \mid S_3 = s_3) = C{\cdot}.3{\cdot}.8{\cdot}.5{\cdot}.1$ which is less than $P(O = acdb, X_2 = s_4, X_3 = s_5) = CP(X_2 = s_4 \mid X_1 = s_1), P(X_3 = s_5 \mid X_2 = s_2)P(O_2 = c \mid S_2 = s_2)P(O_3 = d \mid S_3 = s_3) = C \cdot .7 \cdot 1 \cdot .3 \cdot .6$ for some constant C.

4. $P(O = acdb)$ ( ) $P(O = acdb|X_2 = s_4, X_3 = s_5)$

★ **SOLUTION:** "<". We work out the probabilities $P(O = acdb, X_2 = s_2, X_3 = s_3)$, $P(O = acdb, X_2 = s_2, X_3 = s_5)$, $P(O = acdb, X_2 = s_4, X_3 = s_5)$ and sum them to get $P(O = acdb)$. To get $P(O = acdb \mid S_2 = s_4, S_3 = s_5)$, we can divide $P(O = acdb, X_2 = s_4, X_3 = s_5)$ by $P(X_2 = s_4, X_3 = s_5) = .7$.

5 [10 points] Bayesian Networks and Independence

1. [4 points] For the following Bayesian network graph, construct a set of CPDs $P(a)$, $P(b)$, $P(c \mid a, b)$, $P(d \mid a)$ s.t. A is not independent of B, given D. Justify your answer, e.g., by computing some conditional probabilities in your solution.



★ **SOLUTION:** The constructed distribution needs to satisfy

$$p(a, b \mid d) \neq p(a \mid d)p(b \mid d) \tag{1}$$

or, equivalently,

$$p(a \mid b, d) \neq p(a \mid d). \tag{2}$$

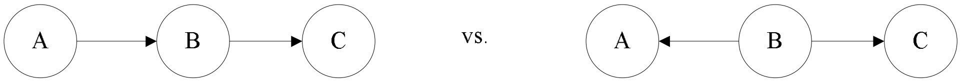
Given a particular distribution p, the terms in Equations ?? and ?? can be computed by marginalizing out one or more variables from $p(a, b, c, d) = p(a)p(b)p(c \mid a, b)p(d \mid c)$. Most choices of $p(a)$, $p(b)$, $p(c \mid b, c)$, and $p(d \mid c)$ satisfy Eq. ?? and ?? (see Thm ? in Chap 2 of Koller, Friedman). With binary A, B, C, D, a simple choice is $p(a) = p(b) = 0.5$, $C = A$ XOR B, $D = C$ with probability 1. In this case, $p(A = 1 \mid D = 1, B = 1) = 0$, whereas $p(A = 1 \mid D = 1, B = 0) = 1$, hence it cannot be the case that

$$p(A = 1 \mid D = 1) = p(A = 1 \mid D = 1, B = 1) = p(A = 1 \mid D = 1, B = 0),$$

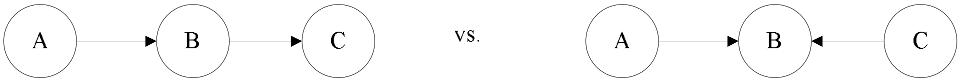
which concludes the proof.

2. [1.5 points each] For each of the following pairs of Bayesian networks, determine if the two Bayesian networks are equivalent, if one is strictly more expressive than the other, or if each represents a different set of independence assumptions. When the two networks are equivalent, state one independence assumption satisfied by them; when the two networks are not equivalent, state an independence assumption satisfied by one, but not the other.

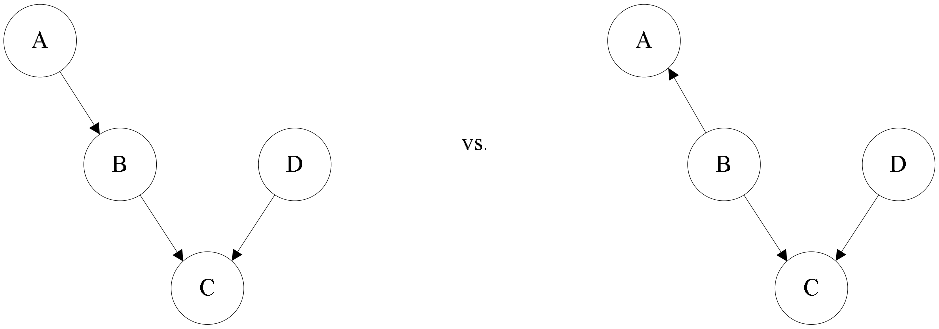
a)



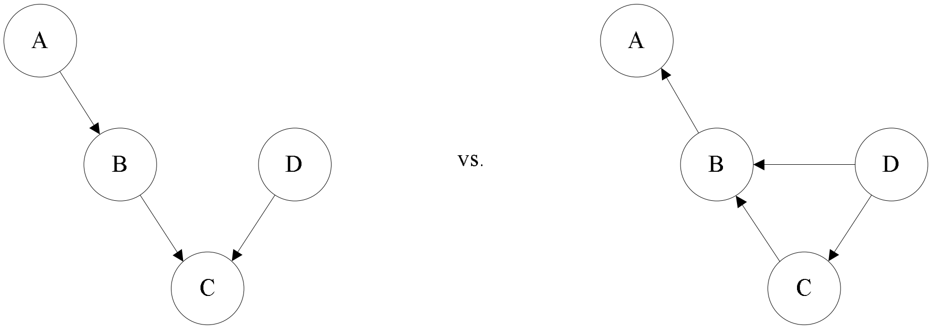
b)



c)



d)



6 [10 points] Bayesian Network Structure Learning

Consider the following Bayesian network tree $\mathcal{T}$:

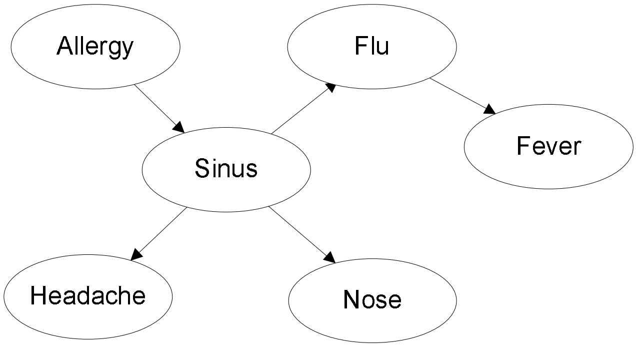


Figure 6: A Flu-Allergy-Sinus Bayesian network.

Note that the edge between Flu and Sinus goes from **Sinus to Flu**, rather than the other way around. You are given a fully observed training set $\mathcal{D} = \{x^1, \ldots, x^N\}$, where $x^i = \langle a^i, s^i, f^i, h^i, n^i, r^i \rangle$ is the i-th data point (f stands for flu, while r stands for fever).

1. [3 points] Write down the log-likelihood score of the tree network in Figure **??**, $\log p(\mathcal{D} \mid \mathcal{T}, \theta_{\mathcal{T}})$.

2. [4 points] Now suppose that that the direction of the edge between Sinus and Flu reverses, introducing a v-structure at Sinus. Denote the new graph with $\mathcal{G}$. What is the change in the log-likelihood

$$\log p(\mathcal{D} \mid \mathcal{G}, \theta_{\mathcal{G}}) - \log p(\mathcal{D} \mid \mathcal{T}, \theta_{\mathcal{T}})?$$

3. [3 points] Describe a general algorithm that, given a directed tree $\mathcal{T}$, finds a Bayesian network structure with maximum log-likelihood score that differs from $\mathcal{T}$ in the direction of exactly one edge.

7 [12 points + 6 extra credit] MDPs and Reinforcement Learning

Consider the following scenario. You are reading email, and you get an offer from the CEO of Marsomania Ltd., asking you to consider investing into an expedition which plans to dig for gold on Mars. You can either choose to invest, with the prospect of either getting money or fooled, or you can instead choose to ignore your emails and go to a party. Of course your first thought is to model this as Markov Decision Process, and you come up with the MDP presented in Figure **??**.

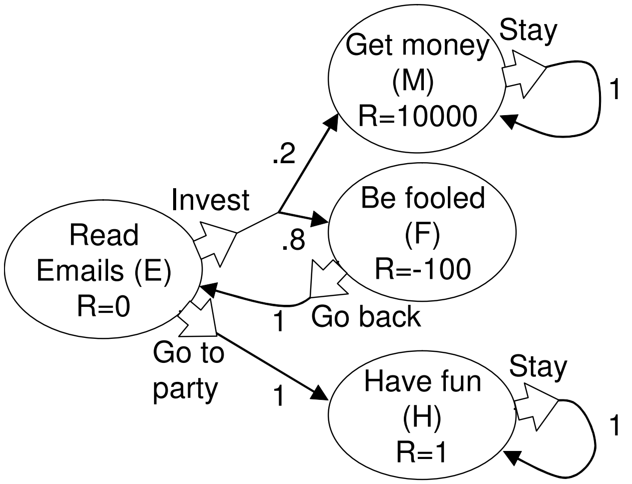


Figure 7: MDP for question **??** on MDPs and RL.

Your MDP has four states: Read emails (E), Get money (M), Be fooled (F) or Have fun (H). The actions are denoted by fat arrows, the (probabilistic) transitions are indicated by thin arrows, annotated by the transition probabilities. The rewards only depend on the state and are indicated on Figure **??**, for example, the reward in state E is 0, in state M it is 10,000.

1. [2 points] What are the possible (deterministic) policies in this MDP?

   ★ **SOLUTION:** There are 2 policies: both map state M to Stay, F to Go back and H to Stay. The only difference is in state E, where we have two choices, either to Invest or to Go to party.

2. [5 points] For each policy, compute the value function (infinite horizon, discount factor 1/2). What is the optimal policy?

   ★ **SOLUTION:** For the Invest policy, we first compute $V_1(M) = 10000\sum_{i=0}^{\infty}(1/2)^i = 20000$. We find $V_1(E) = \frac{1}{2}(.2V_1(M) + .8V_1(F))$. Similarly, $V_1(F) = -100 + \frac{1}{2}V_1(E)$.

Plugging that into $V_1(E)$, we get

$$V_1(E) = \frac{1}{2}(.2V_1(M) + .8(-100 + \frac{1}{2}V_1(E))).$$

Solving for $V_1(E)$, we get

$$V_1(E) = \frac{5}{4}\left(\frac{V_1(M)}{10} - 40\right) = 2450. \tag{3}$$

Hence $V_1(F) = 1125$. For the party policy we get, by similar arguments, $V_2(H) = 2$, $V_2(E) = 1$.

3. [5 points] Now assume you know everything about this model, except the reward state M (i.e., $R(M)$ is unknown). Using the "optimism in the face of uncertainty" heuristic and the *Rmax* algorithm, what is the minimum value for *Rmax*, such that when you initialize $R(M) = Rmax$ with this value, you will prefer to *explore* by choosing the Invest action, instead of *exploiting* by choosing the Go to Party action in state E?

   ★ **SOLUTION:** We need to set $V(M)$ such that, using Eq. **??**,

   $$V_1(E) = \frac{5}{4}\left(\frac{V_1(M)}{10} - 40\right) \geq V_2(E) = 1,$$

   hence $V_2(M) = 2R_{max} \geq 408$, hence $R_{max} = 204$ is the least possible reward which makes investing attractive over partying.

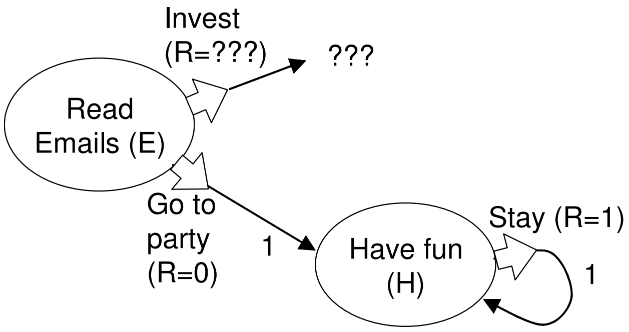


Figure 8: MDP for question **??**.4 on MDPs and RL.

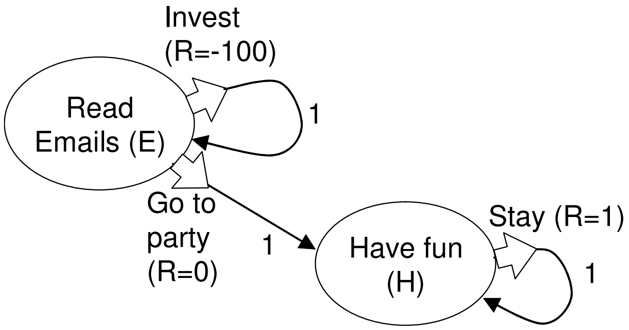


Figure 9: MDP for question **??**.4, full (but unknown) model.

4. [**6 extra credit**] Now assume we know even less about the model: We only know what happens if we go to the party, but not if we choose to invest. This partially explored MDP is presented in Figure **??**. Unlike in the from part 1-3, here the reward depends on the taken action. We know that for staying in state H we get reward 1. Hence we know that $Q(E, \text{Go to party}) = 0 + \gamma \sum_{m=0}^{\infty} \gamma^m = \frac{1}{2} \cdot 2 = 1$ for $\gamma = \frac{1}{2}$.

   We do not know how much reward we get for Investing in state E, and where this action will lead us. Hence we do not know Q(E,Invest). We will use Q-learning to learn this value. Again using the "optimism in the face of uncertainty" heuristic, we assign an optimistic reward $Rmax = 10,000$ to $Q(E, \text{Invest})$. **Unlike in the model from part 1-3**, in this problem, investing on such a dubious email will never pay off; we will always incur the negative reward of -100, and directly go back to the Read Email state. The full model (which we do not know), is presented in Figure **??**.

   Assuming that both the learning rate α and the discount factor γ are 1/2, how many times do we explore the investment strategy, until we decide that going to the party might be the better strategy? *Hint: The formula* $\sum_{m=0}^{k-1} \beta^m = \frac{1-\beta^k}{1-\beta}$ *might come in handy.*

Please work out the answer to Question **??**.4 here.

★ **SOLUTION:** Defining $Q^{(t)}$ the $Q(E, Invest)$ part of the Q-function after iteration t, we initialize $Q^{(0)} = R_{max}$. Using the usual Q learning update, we can work out the recurrence to be

$$Q^{(t)} = \left(\frac{3}{4}\right)^t Q^{(0)} - 50\frac{1-\left(\frac{3}{4}\right)^t}{1-\frac{3}{4}},$$

which has to be greater than $V_2(E) = 1$. Solving for t we get

$$t \geq \left\lceil \frac{\log \frac{R_{max}+50}{201}}{\log \frac{3}{4}} \right\rceil = 14.$$

8 [12 points] EM Algorithm

In this problem, you will derive an EM algorithm for estimating the mixing parameter for a mixture of arbitrary probability densities f_1 and f_2. For example, $f_1(x)$ could be a standard normal distribution centered at 0, and $f_2(x)$ could be the uniform distribution between $[0, 1]$. You can think about such mixtures in the following way: First, you flip a coin. With probability λ (i.e., the coin comes up heads), you will sample x from density f_1, with probability $(1 - \lambda)$ you sample from density f_2.

More formally, let $f_\lambda(x) = \lambda f_1(x) + (1-\lambda) f_2(x)$, where f_1 and f_2 are arbitrary probability density functions on $\mathbb{R}$, and $\lambda \in [0, 1]$ is an unknown mixture parameter.

1. [3 points] Given a data point x, and a value for the mixture parameter λ, compute the probability that x was generated from density f_1.

2. [3 points] Now you are given a data set $\{x_1, \ldots, x_n\}$ drawn i.i.d. from the mixture density, and a set of coin flips $\{c_1, c_2, \ldots, c_n\}$, such that $c_i = 1$ means that x_i is a sample from f_1, and $c_i = 0$ means that x_i was generated from density f_2. For a fixed parameter λ, compute the complete log-likelihood of the data, i.e., $\log P(x_1, c_1, x_2, c_2, \ldots, x_n, c_n \mid \lambda)$.

3. [6 points] Now you are given only a sample $\{x_1, \dots, x_n\}$ drawn i.i.d. from the mixture density, without the knowledge about which component the samples were drawn from (i.e., the c_i are unknown). Using your derivations from part 1 and 2, derive the E- and M-steps for an EM-algorithm to compute the Maximum Likelihood Estimate of the mixture parameter λ. Please describe your derivation of the E- and M-step clearly in your answer.

9 [8 points] Dimensionality reduction - PCA

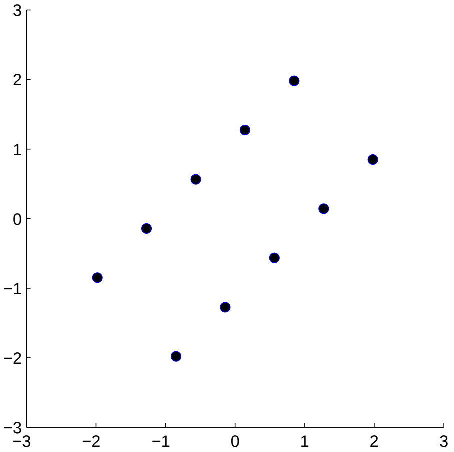
Recall that PCA should be used with caution for classification problems, because it does not take information about classes into account. In this problem you will show that, depending on the dataset, the results may be **very** different.

Suppose that the classification algorithm is 1-nearest-neighbor, the source data is 2-dimensional and PCA is used to reduce the dimensionality of data to 1 dimension. There are 2 classes (+ and -). The datapoints (without class labels) is pictured on plots below (the two plots are identical).

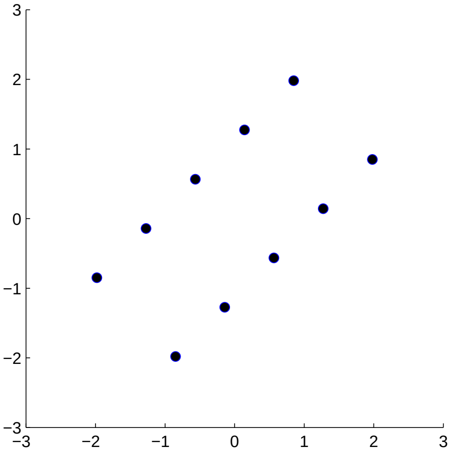
1. [2 points] On one of the plots draw a line that PCA will project the datapoints to.

2. [6 points] For each of the plots, label the source datapoints so that 1-NN will have the following leave-one-out cross-validation error:

2D data: 100% error
1D data from PCA: 0% error



2D data 0% error
1D data from PCA: 100% error

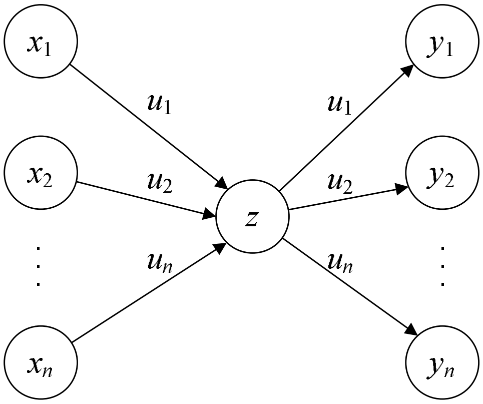


10 [10 points] Dimensionality reduction - neural networks and PCA

It turns out that neural networks can be used to perform dimensionality reduction. We are given a dataset $\mathbf{x}^1, \ldots, \mathbf{x}^m$, where each point is a n-dimensional vector: $\mathbf{x}^i = (\mathbf{x}^i_1, \ldots, \mathbf{x}^i_n)$. Suppose that the dataset is centered:

$$\bar{\mathbf{x}} = \sum_{i=1}^{m} \mathbf{x}^i = 0.$$

Consider the following network with 1 node in the hidden layer:



In this network, the hidden and the output layer share the weight parameters, i.e., the edge $x_j \rightarrow z$ uses the same weight u_j as the edge $z \rightarrow y_j$. The nodes have linear response functions:

$$z = \sum_{j=1}^{n} u_j x_j, \quad y_j = u_j z$$

.

Typically, in neural networks, we have training examples of the form $(\mathbf{x}^i, \mathbf{t}^i)$. Here, the network is trained as follows: for each point $\mathbf{x}^i$ of our dataset, we construct a training example $(\mathbf{x}^i, \mathbf{x}^i)$ that assigns the point to both the inputs and the outputs, i.e., $\mathbf{t}^i = \mathbf{x}^i$.

1. [2 points] Suppose that we minimize a square loss function,

$$l(\mathbf{w};\mathbf{x}) = \sum_i \sum_j \ (\mathbf{t}_j^i - \mathrm{out}_j(\mathbf{x}^i;\mathbf{w}))^2,$$

where out($\mathbf{x}$;$\mathbf{w}$) is the prediction $\mathbf{y}$ of the network on input $\mathbf{x}$, given weights $\mathbf{w}$. Write down the loss function in terms of the network parameters $\mathbf{u}$ and the way we are training this neural network.

2. [4 points] Recall that, in principal component analysis with m components, we approximate a data point $\mathbf{x}^i$, by projecting it onto a set of basis vectors $(\mathbf{u}_1, \ldots, \mathbf{u}_k)$:

$$\hat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j,$$

where $z_j^i = \mathbf{x}^i \cdot \mathbf{u}_j$, in order to minimize the squared reconstruction error:

$$\sum_{i=1}^{m} ||\mathbf{x}^i - \hat{\mathbf{x}}^i||_2^2,$$

where $||v||_2^2 = \sum_{j=1}^{n}(v_j)^2$. Relate the optimization problem from part 1 to the problem optimized in PCA.

3. [4 points] Construct a neural network that will result in the same reconstruction of data as PCA with k first principal components. Write down both the structure of the network and the node response functions.