10701 Machine Learning - Spring 2012

Monday, May 14th 2012 Final Examination **180 minutes**

Name: ______________________ Andrew ID: ______________________

Instructions.

1. Make sure that your exam has 20 pages and is not missing any sheets, then write your full name and Andrew ID on this page (and all others if you want to be safe).
2. Write your answers in the space provided below the problem. If you make a mess, clearly indicate your final answer.
3. The exam has 9 questions, with a maximum score of 100 points. The problems are of varying difficulty. The point value of each problem is indicated.
4. This exam is open book and open notes. You may use a calculator, but any other type of electronic or communications device is not allowed.

Question	Points	Your Score
Q1	8	
Q2	12	
Q3	8	
Q4	12	
Q5	15	
Q6	11	
Q7	10	
Q8	12	
Q9	12	
TOTAL	100	

1 Decision trees and KNNs [8 points]

In the following questions we use Euclidian distance for the KNN.

1.1 [3 points]

Assume we have a decision tree to classify binary vectors of length 100 (that is, each input is of size 100). Can we specify a 1-NN that would result in exactly the same classification as our decision tree? If so, explain why. If not, either explain why or provide a counter example.

Solution: Yes. A simple solution would be to generate all 2^100 possible strings and use the decision tree to classify each of them. Then let the 1-NN be the entire collection of these length 100 binary strings.

1.2 [5 points]

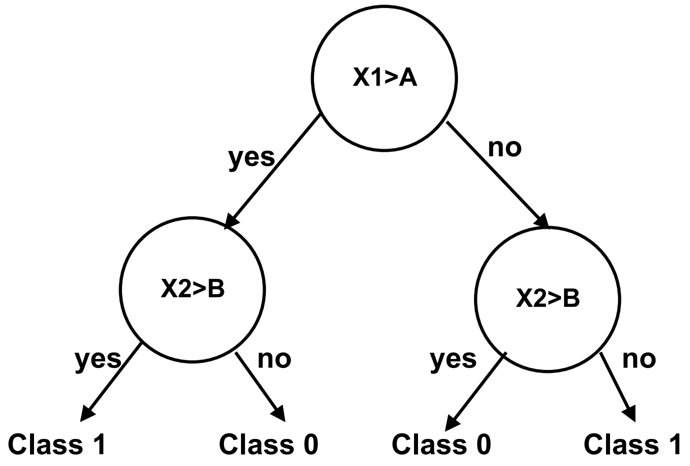


Figure 1: The decision tree for Problem 1.

Assume we have the decision tree in Figure 1 which classifies two dimensional vectors $\{X1, X2\} \in R \setminus \{A, B\}$. In other words, the values A and B are never used in the inputs. Can this decision tree be implemented as a 1-NN? If so, explicitly say what are the values you use for the 1-NN (you should use the minimal number possible). If not, either explain why or provide a counter example.

Solution: Yes. The following 4 points are enough to specify a 1-NN that has the exact same outcome as the decision tree:

$$\{A+1, B+1\} - 1 \tag{1}$$
$$\{A+1, B-1\} - 0 \tag{2}$$
$$\{A-1, B+1\} - 0 \tag{3}$$
$$\{A-1, B-1\} - 1 \tag{4}$$

2 Neural Networks [12 points]

Consider neural networks where each neuron has a *linear* activation function, i.e., each neurons output is given by $g(x) = c + b\frac{1}{n}\sum_{i=1}^{n} W_i x_i$, where b and c are two fixed real numbers and n is the number of incoming links to that neuron.

1. [**3 points**] Suppose you have a single neuron with a linear activation function $g()$ as above and input $\mathbf{x} = x_0, \ldots, x_n$ and weights $\mathbf{W} = W_0, \ldots, W_n$. Write down the squared error function for this input if the true output is a scalar y, then write down the weight update rule for the neuron based on gradient descent on this error function.

   Solution: Error function: $(y - \mathbf{W}^\top \mathbf{x})^2$.
   Update rule: $W_i \leftarrow W_i + \lambda 2 x_i (y - \mathbf{W}^\top \mathbf{x})$.

2. [**3 points**] Now consider a network of linear neurons with one hidden layer of m units, n input units, and one output unit. For a given set of weights $w_{k,j}$ in the input-hidden layer and W_j in the hidden-output layer, write down the equation for the output unit as a function of $w_{k,j}$, W_j, and input x. Show that there is a single-layer linear network with no hidden units that computes the same function.

   Solution: $y \approx \sum_j W_j \sum_k w_{k,j} x_k = \sum_k \left(\sum_j W_j w_{k,j}\right) x_k = \sum_k \beta_k x_k$ Or, $\mathbf{W}^\top \mathbf{w}^\top \mathbf{x} = \beta^\top \mathbf{x}$ where $\beta^\top = \mathbf{W}^\top \mathbf{w}^\top$

3. [**3 points**] Now assume that the true output is a *vector* $\mathbf{y}$ of length o. Consider a network of linear neurons with one hidden layer of m units, n input units, and o output units. If $o > m$, can a single-layer linear network with no hidden units be *trained* to compute the same function? **Briefly** explain why or why not.

   Solution: No. The hidden layer effectively imposes a rank constraint on the learned coefficients that a single layer network will not be able to enforce durning training.

4. [**3 points**] The model in 3) combines dimensionality reduction with regression. One could also reduce the dimensionality of the inputs (e.g. with PCA) and then use a linear network to predict the outputs. **Briefly** explain why this might not be as effective as 3) on some data sets.

   Solution: If some of the linearly independent input dimensions are *not* correlated with the output, then PCA on the inputs alone will not regularize effectively. The model in 3) reduces dimensionality based on the *predictive* capacity of the input dimensions.

3 Gaussian mixtures models [8 points]

3.1 [3 points]

The E-step in estimating a GMM infers the probabilistic membership of **each data point in each component** Z: $P(Z_j|X_i), i = 1, ..., n, j = 1, ..., k$, where i indexes data and j indexes components. Suppose a GMM has **two components** with known variance and an equal prior distribution

$$N(\mu_1, 1) \times 0.5 + N(\mu_2, 1) \times 0.5 \tag{5}$$

The observed data are $x_1 = 2$, and the current estimates of μ_1 and μ_2 are 2 and 1 respectively. Compute the component memberships of **this observed data point** for the next E-step (*hint: normal densities for standardized variable* $y_{(\mu=0,\sigma=1)}$ *at* $0, 0.5, 1, 1.5, 2$ *are* $0.4, 0.35, 0.24, 0.13, 0.05$).

Solution: the tricks here are 1) spot the memberships must sum up to 1 so only need to compute a single probability for each data point 2) the normal densities in the hint can be used to simplify computations by variable transformation $\frac{x-\mu}{\sigma}$*, and also that the density is symmetric about 0. Thus the memberships are*

$$\begin{aligned}
p(z_1|x_1) &= \frac{p(x_1|z_1)}{p(x_1|z_1) + p(x_1|z_2)} = \frac{(.4)(.5)}{(.4)(.5) + (.24)(.5)} = \frac{5}{8} && (6)\\
p(z_2|x_1) &= 1 - p(z_1|x_1) = \frac{3}{8} && (7)\\
p(z_1|x_2) &= \frac{p(x_2|z_1)}{p(x_2|z_1) + p(x_2|z_2)} = \frac{(.13)(.5)}{(.13)(.5) + (.35)(.5)} = \frac{13}{48} && (8)\\
p(z_2|x_2) &= 1 - p(z_1|x_2) = \frac{35}{48} && (9)
\end{aligned}$$

3.2 [5 points]

The Gaussian mixture model (GMM) and the k-means algorithm are closely related—the latter is a special case of GMM. The likelihood of a GMM with Z denoting the latent components can be expressed typically as

$$P(X) = \sum_z P(X|Z)P(Z) \tag{10}$$

where $P(X|Z)$ is the (multivariate) Gaussian likelihood conditioned on the mixture component and $P(Z)$ is the prior on the components. Such a likelihood formulation can also be used to describe a k-means clustering model. Which of the following statements are true—choose *all* correct options if there are multiple ones.

a) $P(Z)$ is uniform in k-means but this is not necessarily true in GMM.

b) The values in the covariance matrix in $P(X|Z)$ tend towards zero in k-means but this is not so in GMM.

c) The values in the covariance matrix in $P(X|Z)$ tend towards infinity in k-means but this is not so in GMM.

d) The covariance matrix in $P(X|Z)$ in k-means is diagonal but this is not necessarily the case in GMM.

Solution: a), b), d). The k-means algorithm is a special case of GMM where the covariance in the Gaussian likelihood function is diagonal with elements tending towards zero. The prior on the components is uniform in k-means.

4 Semi-Supervised learning [12 points]

4.1 [6 points]

Assume we are trying to classify stocks to predict whether the stock will increase (class 1) or decrease (class 0) based on the stock closing value in the last 5 days (so our input is a vector with 5 values). We would like to use logistic regression for this task. We have both labeled and unlabeled data for the learning task. For each of the 4 semi-supervised learning methods we discussed in class, say whether it can be used for this classification problem or not. If it can, briefly explain if and how you should change the logistic regression target function to perform the algorithm. If it cannot, explain why.

1. Re-weighting

   Solution: Yes. As discussed in class and the problem set, we can use re-weighting to obtain a different target function in logistic regression classification and so we will just re-weight the labeled data based on the distribution of the unlabeled data points and solve the revised target function.

2. Co-Training

   Solution: No. As we mentioned for co-training, the features need to be independent for co-training for work. However you cut time series data, it wont be independent (strong autocorrelation exist between consecutive day values) and so co-training is not appropriate for this data.

3. EM based

   Solution: No. That method was specifically discussed in the context of GMMs and Bayes classification methods.

4. Minimizing overfitting

   Solution: Yes. For example, we can use the unlabeled data to choose an appropriate polynomial transformation for the input vectors.

4.2 [3 points]

Assume we are using co-training to classify the rectangles and circles in Figure 2 a). The ? represents unlabeled data. For our co-training we use linear classifiers where the first classifier uses only the x coordinate values and the second only the y axis values. Choose from the answers below the number of iterations that will be performed until our co-training converges for this problem and briefly explain:

1. 1
2. 2
3. more than 2
4. Impossible to tell.

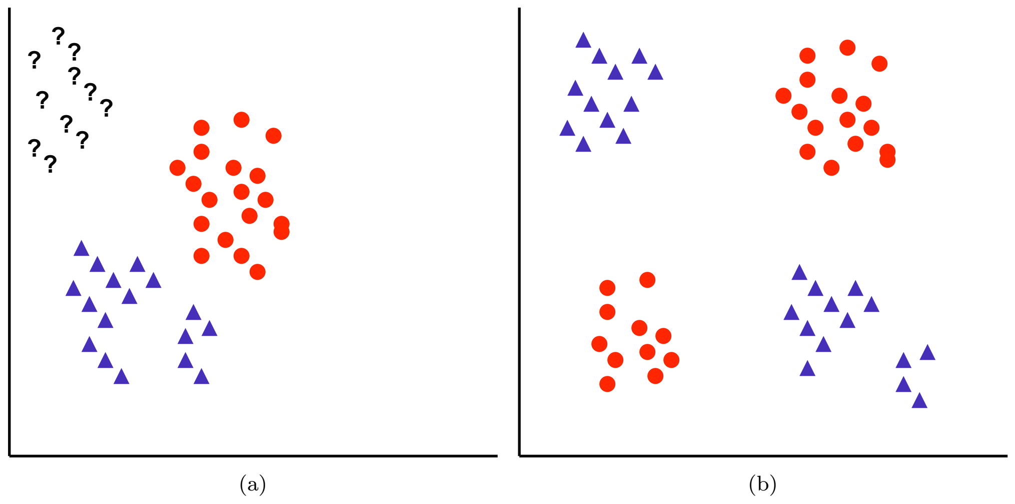


Figure 2: The datapoints of questions 4.2 and 4.3.

Solution: A. Note that co-training depends on agreement between the two classifiers to perform the next step. Thus, we need at least one point in the intersection of both classifiers (that is, an unlabeled point on which both classifiers agree). However, for this data the x axis classifier would classify all unlabeled points as triangles whereas the y axis classifier would classify all unlabeled points as circles and so there is no points for a second iterations.

4.3 [3 points]

Now assume that we are using boosting (with linear classifiers, so each classifier is a line in the two dimensional plane) to classify the points in Figure 2 b). We terminate the boosting algorithm once we reach a t such that $\epsilon_t = 0$ or after 100 iterations. How many iterations do we need to perform until the algorithm converges? Briefly explain.

1. 1
2. 2
3. more than 2
4. Impossible to tell.

Solution: C. As we discussed in class, even if the overall error (which is based on the collection of classifiers we learned so far) goes to 0, we may still have misclassified points for the current classifier (t). In this case, clearly both classifiers (in iteration 1 and 2) would make mistakes and so ϵ_t would not be 0 for either one and we will continue.

5 Bayesian networks [15 points]

5.1 [5 points]

1. [**2 points**] Show that $a \perp\!\!\!\perp b, c \mid d$ (a is conditionally independent of $\{b, c\}$ given d) implies $a \perp\!\!\!\perp b \mid d$ (a is conditionally independent of b given d).

   Solution:

$$p(a, b \mid d) = \sum_c p(a, bc \mid d) \tag{11}$$
$$= \sum_c p(a \mid d) p(b, c \mid d) \tag{12}$$
$$= p(a \mid d) \sum_c p(b, c \mid d) \tag{13}$$
$$= p(a \mid d) p(b \mid d) \tag{14}$$

   You can also show by d-separation.

2. [**3 points**] Define the skeleton of a BN $\mathcal{G}$ over variables $\mathcal{X}$ as the *undirected* graph over $\mathcal{X}$ that contains an edge $\{X, Y\}$ for every edge (X, Y) in $\mathcal{G}$. Show that if two BNs $\mathcal{G}$ and $\mathcal{G}'$ over the same set of variables $\mathcal{X}$ having the same skeleton and the same set of v-structures, encode the same set of independence assumptions. V-structure is a structure of 3 nodes X, Y, Z such as $X \to Y \leftarrow Z$.

   Hints: It suffices to show that for any independence assumption $A \perp\!\!\!\perp B \mid C$ (A, B, C are mutually exclusive sets of variables), it is encoded by $\mathcal{G}$ if and only if it is also encoded $\mathcal{G}'$. Show by d-separation.

Solution: Consider an independence assumption$A \perp\!\!\!\perp B \mid C$ (A, B, C are mutually exclusive sets of variables), which is encoded by $\mathcal{G}$. For any path from A to B in $\mathcal{G}$, it also exists in $\mathcal{G}'$, since they have the same skeleton. We need to show that this path is active in $\mathcal{G}$ if and only if it is also active in $\mathcal{G}'$. Consider any three consecutive variables along the path:X, Y, Z. There are two cases:

1. $X \to Y \to Z$, $X \leftarrow Y \leftarrow Z$, $X \leftarrow Y \to Z$ or not a v-structure in $\mathcal{G}$: since $\mathcal{G}$ and $\mathcal{G}'$ have the same set of v-structures, this is not a v-structure $\mathcal{G}'$, therefore this part of the path is active in $\mathcal{G}$ if and only if it is also active in $\mathcal{G}'$.

2. $X \to Y \leftarrow Z$ is a v-structure in $\mathcal{G}$: again since $\mathcal{G}$ and $\mathcal{G}'$ have the same set of v-structures, this is also a v-structure $\mathcal{G}'$, therefore this part of the path is active in $\mathcal{G}$ if and only if it is also active in $\mathcal{G}'$.

5.2 [5 points]

For each of the following pairs of BNs in Figure 3, determine if the two BNs are equivalent, e.g they have the same set of independence assumptions. When they are equivalent, state *one* independence/conditional independence assumption shared by them. When they are not equivalent, state *one* independence/conditional independence assumption satisfied by one but not the other.

Solution: The previous question should give you hints on how to do this quickly!

1. Yes, e.g $A \perp\!\!\!\perp C \mid B$.

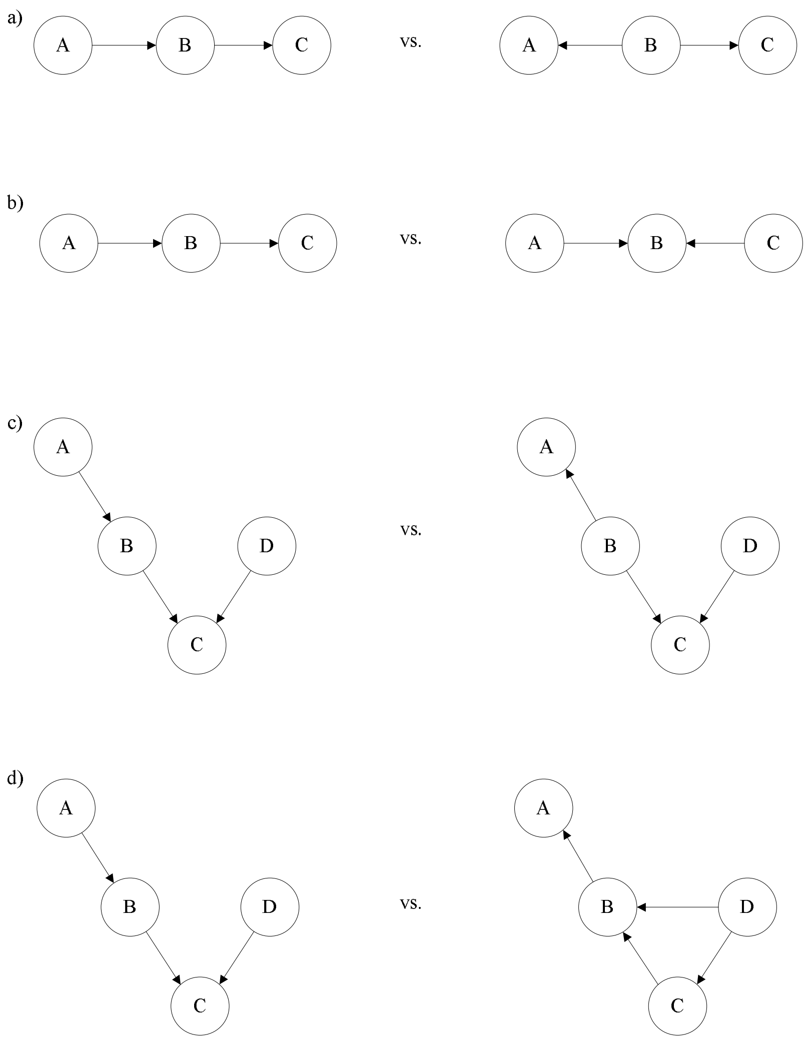


Figure 3: The pairs of Bayesian networks for 5.2.

2. No, e.g $A \perp\!\!\!\perp C \mid B$ is valid in the first one, but not the second.
3. Yes, e.g $B \perp\!\!\!\perp D$.
4. No, e.g $B \perp\!\!\!\perp D$ is valid in the first one, but not the second.

5.3 [5 points]

We refer to the Figure 4 in this question.

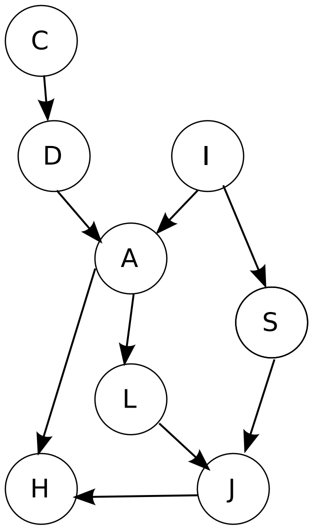


Figure 4: A Bayesian network for 5.3.

1. What is Markov blanket of $\{A, S\}$?

   Solution: $\{D, I, H, L, J\}$

2. (T/F) For each of the following independence assumptions, please state whether it is true or false:

   (a) $D \perp\!\!\!\perp S$.
   (b) $D \perp\!\!\!\perp S \mid H$.
   (c) $C \perp\!\!\!\perp J \mid H$.
   (d) $C \perp\!\!\!\perp J \mid A$.

   Solution:

   (a) $D \perp\!\!\!\perp S$: T
   (b) $D \perp\!\!\!\perp S \mid H$: F
   (c) $C \perp\!\!\!\perp J \mid H$: F
   (d) $C \perp\!\!\!\perp J \mid A$: F

6 Hidden Markov Models [11 points]

6.1 [3 points]

Assume we have temporal data from two classes (for example, 10 days closing prices for stocks that increased / decreased on the following day). How can we use HMMs to classify this data?

Solution: We would need to learn two HMMs, one for each class. For a new test data vector, we will run the Viterby algorithm for this vector in both HMMs and chose the HMM with higher likelihood as the class for this vector.

6.2 [3 points]

Derive the probablity of:

$$P(o_1, \ldots, o_t, q_{t-1} = s_v, q_t = s_j) \tag{15}$$

You may use any of the model parameters (starting, emission and transition probabilities) and the following constructs (defined and derived in class) as part of your derivation:

$$\begin{aligned} p_t(i) &= p(q_t = s_i) && (16)\\ \alpha_t(i) &= p(o_1, \ldots, o_t, q_t = s_i) && (17)\\ \delta_i(i) &= \max_{q_1,\ldots,q_{t-1}} p(q_1, \ldots, q_{t-1}, q_t = s_i, O_1, \ldots, O_t) && (18) \end{aligned}$$

Note that you may not need to use all of these constructs to fully define the function listed above.

Solution:

$$\begin{aligned} &P(o_1, \ldots, o_t, q_{t-1} = s_v, q_t = s_j) && (19)\\ =&P(o_t, q_t = s_j \mid o_1, \ldots, o_{t-1}, q_{t-1} = s_v)P(o_1, \ldots, o_{t-1}, q_{t-1} = s_v) && (20)\\ =&P(o_t, q_t = s_j \mid; q_{t-1} = s_v)\alpha_{t-1}(v) && (21)\\ =&P(o_t \mid q_t = s_j, q_{t-1} = s_v)P(q_t = s_j \mid q_{t-1} = s_v)\alpha_{t-1}(v) && (22)\\ =&b_j(o_t)a_{v,j}\alpha_{t-1}(v) && (23) \end{aligned}$$

6.3 [5 points]

The following questions refer to figure 5. In that figure we present a HMM and specify both the transition and emission probabilities. Let p_A^t be the probability of being in state A at time t. Similarly define p_B^t and p_C^t.

1. What is p_C^3?

   Solution: p_C^3 = A1A2.

2. Define p_2^t as the probability of observing 2 at time point t. Express p_2^t as a function of p_B^t and p_C^t (you can also use any of the model parameters defined in the figure if you need).

   Solution: $0.3 * (1 - p_B^t - p_C^t)$.

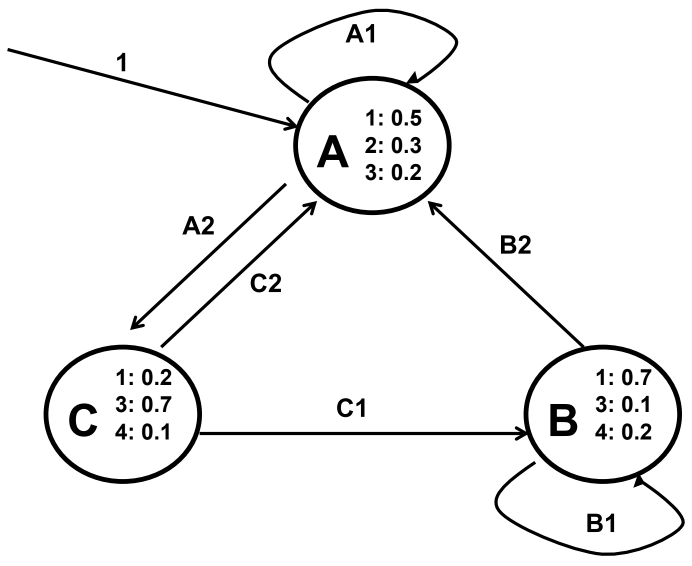


Figure 5: The HMM for 6.3.

7 Dimension Reduction [10 points]

7.1 [3 points]

Which of the following unit vectors expressed in coordinates $(X1, X2)$ correspond to *theoretically* correct directions of the 1st (p) and 2nd (q) principal components (*via linear PCA*) respectively for the data shown in Figure 6? Choose *all* correct options if there are multiple ones.

a) **i)** p$(1,0)$ q$(0,1)$ **ii)** p$(\frac{1}{\sqrt{(2)}}, \frac{1}{\sqrt{(2)}})$ q$(\frac{1}{\sqrt{(2)}}, \frac{-1}{\sqrt{(2)}})$

b) **i)** p$(1,0)$ q$(0,1)$ **ii)** p$(\frac{1}{\sqrt{(2)}}, \frac{-1}{\sqrt{(2)}})$ q$(\frac{1}{\sqrt{(2)}}, \frac{1}{\sqrt{(2)}})$

c) **i)** p$(0,1)$ q$(1,0)$ **ii)** p$(\frac{1}{\sqrt{(2)}}, \frac{1}{\sqrt{(2)}})$ q$(\frac{-1}{\sqrt{(2)}}, \frac{1}{\sqrt{(2)}})$

d) **i)** p$(0,1)$ q$(1,0)$ **ii)** p$(\frac{1}{\sqrt{(2)}}, \frac{1}{\sqrt{(2)}})$ q$(\frac{-1}{\sqrt{(2)}}, \frac{-1}{\sqrt{(2)}})$

e) All of the above are correct.

Solution: a) and c). Three points are tested: PCs are ordered according to variability; PCs are orthogonal; PC axes are potentially unidentifiable.

7.2 [4 points]

In linear PCA, the covariance matrix of the data $\mathbf{C} = \mathbf{X}^T\mathbf{X}$ is decomposed into weighted sums of its eigenvalues (λ) and eigenvectors $\mathbf{p}$:

$$\mathbf{C} = \sum_i \lambda_i \mathbf{p}_i \mathbf{p}_i^T \tag{24}$$

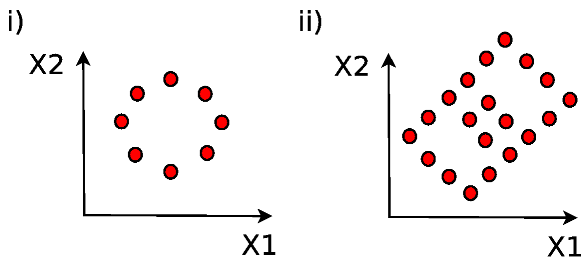


Figure 6: Data in two-dimensions spanned by $X1$ and $X2$.

Prove mathematically that the first eigenvalue λ_1 is identical to the variance obtained by projecting data into the first principal component $\mathbf{p}_1$ (*hint: PCA maximizes variance by projecting data onto its principal components*).

Solution: the variance in the first PC is $v = \mathbf{p}_1^T \mathbf{C} \mathbf{p}_1$. *Since* λ_1 *is the eigenvector,* $\lambda_1 \mathbf{p}_1 = \mathbf{C}\mathbf{p}_1 \Rightarrow \lambda_1 \mathbf{p}_1^T \mathbf{p}_1 = \mathbf{p}_1^T \mathbf{C} \mathbf{p}_1 \Rightarrow \lambda_1 \cdot 1 = v \because \mathbf{p}_1^T \mathbf{p}_1 = 1$ *(Q.E.D.).*

7.3 [3 points]

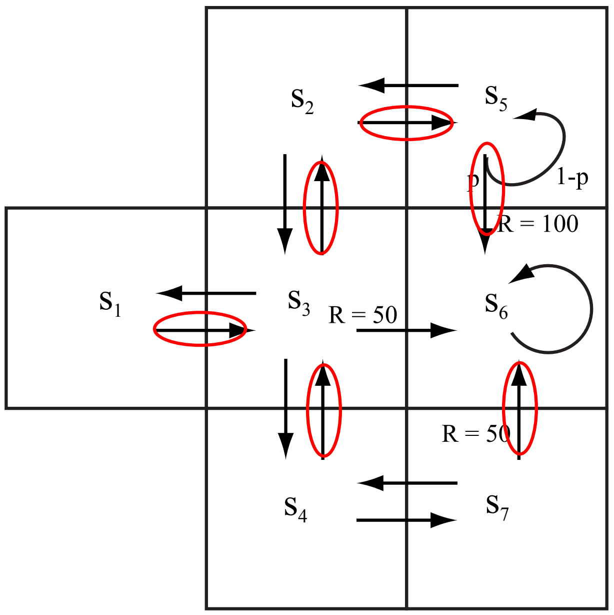
The key assumption of a naive Bayes (NB) classifier is that features are independent, which is not always desirable. Suppose that linear principal components analysis (PCA) is first used to transform the features, and NB is then used to classify data in this low-dimensional space. Is the following statement true? Justify your answers.

> The independent assumption of NB would now be valid with PCA transformed features because all principal components are orthogonal and hence uncorrelated.

Solution: This statement is false. First, uncorrelation is not equivalent to independence. Second, transformed features are not necessarily uncorrelated if the original features are correlated in a nonlinear way.

8 Markov Decision Processes [12 points]

Consider the following Markov Decision Process (MDP), describing a simple robot grid world. The values of the *immediate rewards* R are written next to the transitions. Transitions with no value have an immediate reward of 0. Note that the result of the action "go south" from state S_5 results in one of two outcomes. With probability p the robot succeeds in transitioning to state S_6 and receives immediate reward 100. However, with probability $(1-p)$ it gets stuck in sand, and remains in state S_5 with zero immediate reward. **Assume the discount factor** $\gamma = 0.8$. **Assume the probability** $p = 0.9$.



1. **[3 points]** Mark the state-action transition arrows that correspond to one *optimal* policy. If there is a tie, always choose the state with the smallest index.

   Solution: See figure.

2. **[3 points]** Is it possible to change the value for γ so that the optimal policy is changed? If yes, give a new value for γ and describe the change in policy that it causes. Otherwise **briefly** explain why this is impossible.

   Solution: $\gamma = .7$. The optimal policy now takes action $S_3 \rightarrow S_6$.

3. **[3 points]** Is it possible to change the immediate reward function so that V^* changes but the optimal policy π^* remains unchanged? If yes, give such a change and describe the resulting changes to V^*. Otherwise **briefly** explain why this is impossible.

Solution: Double each reward. V^* is also doubled but the policy remains unchanged.

4. [**3 points**] How sticky does the sand have to get before the robot will prefer to completely avoid it? Answer this question by giving a probability for p below which the optimal policy chooses actions that completely avoid the ice, even choosing the action "go west" over "go south" when the robot is in state S_5.

Solution:

$$50\gamma^2 = \frac{100p}{1-\gamma(1-p)} \implies p = \frac{\gamma^2-\gamma^3}{2-\gamma^3} = \frac{8}{93} \approx 0.086$$

9 Boosting[12 points]

9.1 [4 points]

AdaBoost can be understood as an optimizer that minimizes an exponential loss function $E = \sum_{i=1}^{N} \exp(-y_i f(x_i))$ where $y = +1$ or -1 is the class label, x is the data and $f(x)$ is the weighted sum of weak learners. Show that the loss function E is strictly greater than and hence an upper bound on a $0-1$ loss function $E_{0-1} = \sum_{i=1}^{N} 1 \cdot (y_i f(x_i) < 0)$ (*hint:* E_{0-1} *is a step function that assigns value 1 if the classifier predicts correctly and 0 otherwise*).

Solution: $E_{0-1} = \sum_{i=1}^{N} 1 \cdot (y_i f(x_i) < 0) = \sum_{i=1}^{N} 1 \cdot (-y_i f(x_i) > 0)$*, which can be easily transformed to derive the upper bound*

$$\sum_{i=1}^{N} \log(\exp(-y_i f(x_i) > 0)) \leq \sum_{i=1}^{N} \exp(-y_i f(x_i) > 0) \leq \sum_{i=1}^{N} \exp(-y_i f(x_i)) = E$$

Q.E.D.

9.2 [4 points]

The AdaBoost algorithm has two caveats. Answer the following questions regarding these.

a) Show mathematically why a weak learner with $<$ 50% predictive accuracy presents a problem to AdaBoost.

b) AdaBoost is susceptible to outliers. Suggest a simple heuristic that relieves this.

Solution: a) the weight assigned to the weak learners $\alpha = \frac{1-e}{e}$*, if* $e > .5$ *then the weight becomes negative and the algorithm breaks b) since each (misclassified) data point receives a penalty weight, one way to ignore outliers is to threshold on the weight vector and prunes those points that exceeds certain weights.*

9.3 [4 points]

Figure 7 illustrates the decision boundary (the middle intersecting line) after the first iteration in an AdaBoost classifier with decision stumps as the weak learner. The square points are from class -1 and the circles are from class 1. Draw (roughly) in a solid line the decision boundary at the second iteration. Draw in dashed line the ensemble decision boundary based on decisions at iteration 1 and 2. State your reasoning.

Solution: the decision boundary based on stumping at the second iteration should be towards left of the middle line because there are more circles mispredicted than squares–the line should at least include a few more circles in its right-hand side region. The ensemble decision boundary should be between the two decision boundaries from iteration 1 and 2 because AdaBoost predicts by weighting the individual weak learners.

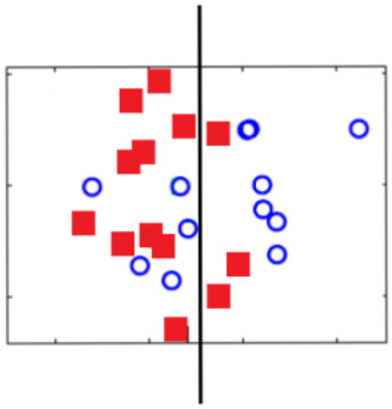


Figure 7: The solid line in the middle is the decision boundary after the first iteration in AdaBoost. The classifier predicts points to the left of the line as class -1 and those to the right as class 1.