Bayes Inference

- Bayes Rule:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- Bayesian Decision:
  - Maximum a Posterior (MAP)

$$a = \arg\max_a P(Y = a|E = e)$$
$$= \text{argmax} \frac{P(E = e|Y = a)P(Y = a)}{P(E = e)}$$
$$= \text{argmax}\, P(E = e|Y = a)P(Y = a)$$

Thus, $P(Y = a|E = e) \propto P(E = e|Y = a)P(Y = a)$

posterior — likelihood — prior

  - Maximum Likelihood (ML) decision:

$$a = \arg\max_a P(E = e|Y = a)$$

Naïve Bayes Model

- Suppose we have many different types of observations (symptoms, features) $E_1, \ldots, E_n$ that we want to use to obtain evidence about an underlying hypothesis Y

- MAP decision:

$$a = \text{argmax}\, p(Y = a|E_1 = e_1, \ldots, E_n = e_n)$$

$$= \text{argmax}\, p(Y = a)p(E_1 = e_1, \ldots, E_n = e_n|Y = a)$$

$$\approx \text{argmax}\, p(Y = a)p(y_1|a)p(y_2|a) \ldots p(y_n|a)$$

(given that the different features are conditionally independent:)

$$P(e_1, \ldots, e_n|y) \approx P(e_1|y)P(e_2|y) \ldots P(e_n|y)$$

- "Conditionally independent" is Naïve Bayes Assumption.

Bayes Learning Bag of Word Model

- The "bag of words model" has the following parameters:
  - $\lambda_{cw} \equiv P(W = w|C = c)$
  - $\pi_c \equiv P(C = c)$
- The training data are a set of documents, $E = [D_1, \ldots, D_m]$, each with its associated class label, $Y = [C_1, \ldots, C_m]$.
- Each document is a sequence of words, $D_i = [W_{1i}, \ldots, W_{ni}]$.

$$P(E, Y) = \prod_{i=1}^{m} P(C_i = c_i) \prod_{j=1}^{n} P(W_{ji} = w_{ji}|C_i = c_i) = \prod_{i=1}^{m} \pi_{c_i} \prod_{j=1}^{n} \lambda_{c_i w_{ji}}$$

- The data likelihood $P(E, Y)$ is maximized if we choose:

$$\lambda_{cw} = \frac{\#\text{ occurrences of word } w \text{ in documents of type } c}{\text{total number of words in all documents of type } c}$$

$$\pi_c = \frac{\#\text{ documents of type } c}{\text{total number of documents}}$$

Laplace Smoothing

- Laplace Smoothing is introduced to solve the problem of zero probability.
- The basic idea: add 1 "unobserved observation" to every possible event

$$P(Y = C) = \frac{\sum_{i=1}^{N} I(y_i = C) + \lambda}{N + K\lambda}$$

  - K denotes the number of different values in y
  - Usually, λ in the formula equals to 1.

Linear Classifier

- A linear classifier cannot learn an XOR function
- Multi-Class Linear Classification:

$$y = \arg\max_c \left(\beta_c + \sum_{w=1}^{V} \alpha_{cw} f_{cw} \right)$$

where f_{cw} are the features (binary, integer, or real), α_{cw} are the feature weights, and β_c is the offset

- Binary classifier, for example:

$$y = 1 \quad \text{if} \quad \beta_c + \sum_{w=1}^{V} \alpha_{cw} f_{cw} > 0$$
$$y = 0 \quad \text{if} \quad \beta_c + \sum_{w=1}^{V} \alpha_{cw} f_{cw} < 0.$$

Perceptron

- Need to be differentiable
- Basic idea: replace the non-differentiable decision function

$$y' = \text{sign}(\vec{w}^T \vec{f})$$

with a differentiable decision function

$$y' = \tanh(\vec{w}^T \vec{f})$$

- Same idea works for multi-class perceptrons. We replace the non-differentiable decision rule $c = argmax_c \boldsymbol{w}_c \cdot \boldsymbol{f}$ with the differentiable decision rule $c = softmax_c \boldsymbol{w}_c \cdot \boldsymbol{f}$, where the softmax function is defined as

$$p(c|\vec{f}) = \frac{e^{\vec{w}_c \cdot \vec{f}}}{\sum_{k=1}^{\#\, classes} e^{\vec{w}_k \cdot \vec{f}}}$$

Product Rule

- Definition of conditional probability: $P(A \mid B) = \frac{P(A,B)}{P(B)}$
- Sometimes we have the conditional probability and want to obtain the joint:

$$P(A, B) = P(A \mid B)P(B) = P(B \mid A)P(A)$$

- The chain rule:

$$P(A_1, \ldots, A_n) = P(A_1)P(A_2 \mid A_1)P(A_3 \mid A_1, A_2) \ldots P(A_n \mid A_1, \ldots, A_{n-1})$$
$$= \prod_{i=1}^{n} P(A_i \mid A_1, \ldots, A_{i-1})$$

Useful Version of BR:

This version is what you memorize. →

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

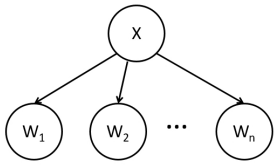
- Remember, P(B|A) is easy to measure (the probability that light hits our solar cell, if the sun still exists and it's daytime). Let's assume we also know P(A) (the probability the sun still exists).
- But suppose we don't really know P(B) (what is the probability light hits our solar cell, if we don't really know whether the sun still exists or not?)

This version is what you actually use. →

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\neg A)P(\neg A)}$$

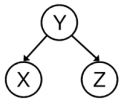
Bayesian Network Structure on Parameters

- Nodes: random variables
  - X: document class
  - $W_1, \ldots, W_n$: words in the document
- Arcs: interactions
  - An arrow from one variable to another indicates direct influence
  - Must form a directed, *acyclic* graph



Conditional Independence (≠ Independence)

Common cause: Conditionally Independent



Y: Project due
X: Newsgroup busy
Z: Lab full

Are X and Z independent? **No**

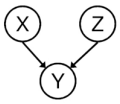
$$P(Z,X) = \sum_Y P(Z|Y)P(X|Y)P(Y)$$

$$P(Z)P(X) = \left(\sum_Y P(Z|Y)P(Y)\right)\left(\sum_Y P(X|Y)P(Y)\right)$$

Are they conditionally independent given Y? **Yes**

$$P(Z,X|Y) = P(Z|Y)P(X|Y)$$

Common effect: Independent



X: Raining
Z: Ballgame
Y: Traffic

Are X and Z independent? **Yes**

$$P(X,Z) = P(X)P(Z)$$

Are they conditionally independent given Y? **No**

$$P(Z,X|Y) = \frac{P(Y|X,Z)P(X)P(Z)}{P(Y)} \neq P(Z|Y)P(X|Y)$$

Hidden Markov Model (HMM)

- At each time slice t, the state of the world is described by an unobservable variable X_t and an observable *evidence* variable E_t
- Transition model: distribution over the current state given the whole past history:

$$P(X_t|X_0,\dots,X_{t-1}) = P(X_t|X_{0:t-1}) \Rightarrow P(X_t|X_{t-1})$$

(since the current state is conditionally independent of all the other states given the state in the previous time step)

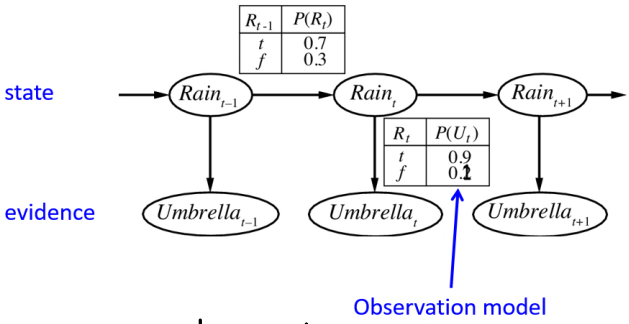
- Observation model:

$$P(Et\ |X_{0:t}, E_{0:t-1}) \Rightarrow P(E_t|X_t)$$

(since the evidence at time t depends only on the state at time t)

Example:

Transition model



Inference Tasks and Learning:

- Inference tasks
  - Filtering: what is the distribution over the current state X_t given all the evidence so far, $e_{1:t}$
  - Smoothing: what is the distribution of some state X_k given the entire observation sequence $e_{1:t}$?
  - Evaluation: compute the probability of a given observation sequence $e_{1:t}$
  - Decoding: what is the most likely state sequence $X_{0:t}$, given the observation sequence $e_{1:t}$?
- Learning
  - Given a training sample of sequences, learn the model parameters (transition and emission probabilities)
    - EM (expectation maximization) algorithm

Markov Decision Process (MDP)

- S - finite set of domain states
- A - finite set of actions
- $P(s' \mid s, a)$ - state transition function
- $R(s), R(s,a), or\ R(s,a,s')$ - reward function
  Could be negative to reflect cost
- S_0 - initial state
- The Markov assumption:

$$P(s_t\ |s_{t-1}, s_{t-2}, \dots, s_1, a) = P(s_t|s_{t-1}, a)$$

Computing Optimal policy the Bellman Equation

- Optimal Policy:
  An Optimal Policy is a policy where you are always choosing the action that maximizes the "return"/"utility" of the current state

$$\pi^*(s) = \arg\max_a \sum_{s'} P(s'|s,a) \cdot U(s')$$

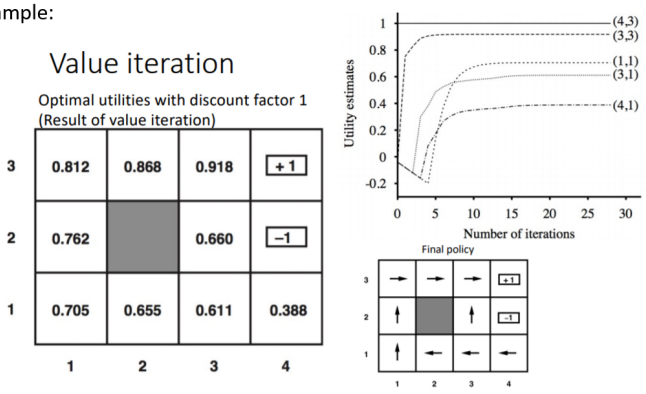
- Utility function:

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a)U(s')$$

Value Iteration and Policy Iteration

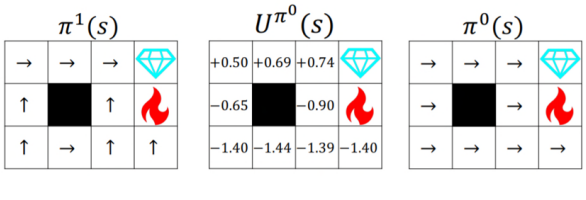
- Value Iteration:
  initialize U'
  repeat
  $U \leftarrow U'$
  for each state s do
  $U'[s] \leftarrow R[s] + \gamma \max_a \sum_{s'} P(s'|s,a)U(s')$
  end until CloseEnough (U, U')
  return greedy policy with respect to U'
- Policy Iteration:
  repeat
  $\pi \leftarrow \pi'$
  $U \leftarrow ValueDetermination(\pi)$ reverse from value iteration
  for each state s do
  $\pi'[s] \leftarrow \max_a \sum_{s'} P(s'|s,a)U(s')$
  end until $\pi = \pi'$

VI Example:



PI Example:

$$\pi^{i+1}(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s'|s,a)U^{\pi_i}(s')$$



Q-Learning

- Q-learning:
  learn an action-utility function $Q(s,a)$ that tells us the value of doing action a in state s

$$U(s) = \max_a Q(s,a)$$

- Equilibrium constraint on Q values:

$$Q(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a')$$

- Transformation and Application:
  - Temporal Difference (TD) Learning
  - State-Action-Reward-State-Action (SARSA)

Loss Function

Testing the Neural Net

- sum-squared error (SSE):

$$L_{SSE} = \sum_{i=1}^{N}\sum_{j=1}^{M}(Y_{ij} - F_{ij})^2 \quad A_{ij}^{(0)} = X_{ij}$$

where $\vec{F}_i = [F_{i1}, \ldots, F_{iM}]$

- cross-entropy: the log probability (softmax) of the correct class

$$L_{CE} = -\sum_{i=1}^{N} \ln F_{i,Y_i}$$

where $F_{ij} = P(Y_i = j^{th} \text{ type of category}) = \frac{e^{z_{ij}^{(L)}}}{\sum_k e^{z_{ik}^{(L)}}}$

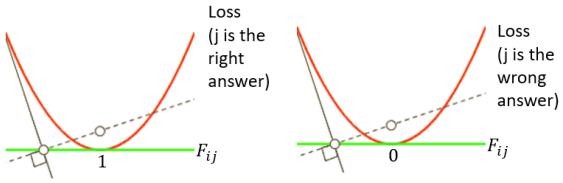
Back-Propagation

Training the Neural Net

A neural net is trained according to gradient descent:

$$W_{jk}^{(l)} = W_{jk}^{(l)} - \eta \frac{\partial L}{\partial W_{jk}^{(l)}}$$

So that the loss function, L, gradually approaches a local minimum

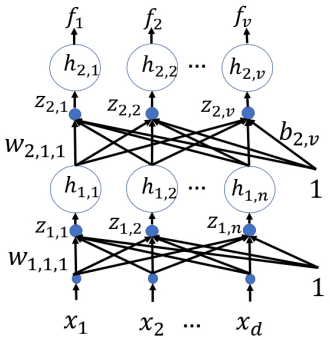


its chain Rules

$$\frac{\partial \mathcal{L}}{\partial h_{1,j}} = \sum_{k=1}^{v} \frac{\partial \mathcal{L}}{\partial z_{2,k}} \times \frac{\partial z_{2,k}}{\partial h_{1,j}}$$

$$\frac{\partial \mathcal{L}}{\partial z_{1,j}} = \frac{\partial \mathcal{L}}{\partial h_{1,j}} \times \frac{\partial h_{1,j}}{\partial z_{1,j}}$$

$$\frac{\partial \mathcal{L}}{\partial w_{1,j,k}} = \frac{\partial \mathcal{L}}{\partial z_{1,j}} \times \frac{\partial z_{1,j}}{\partial w_{1,j,k}}$$



The key to backpropagation is that each individual derivative is easy. For example, if $\mathcal{L}$ is cross-entropy and $f(x) = \sigma(z_2)$, then you already know that

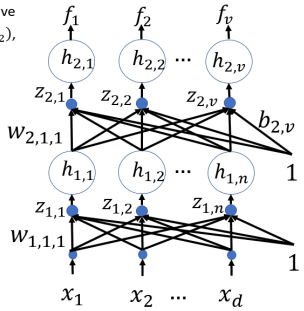
$$\frac{\partial \mathcal{L}}{\partial z_{2,k}} = f_k(x) - \mathbb{1}_{y=k}$$

Since $z_2 = w_2 h_1$ is just a matrix multiplication,

$$\frac{\partial z_{2,k}}{\partial h_{1,j}} = w_{2,k,j}$$

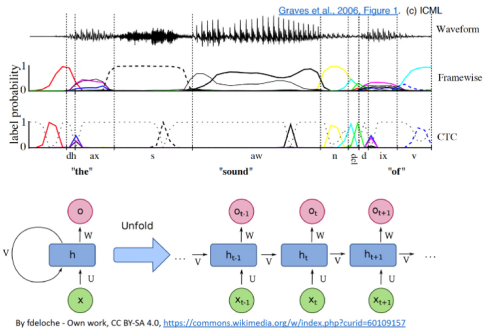
... and the derivative of ReLU is the unit step:

$$\frac{\partial h_{1,j}}{\partial z_{1,j}} = u(z_{1,j})$$



NLP

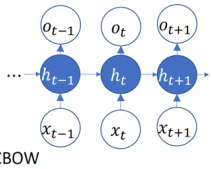
Recurrent neural network



By fdeloche - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=60109157

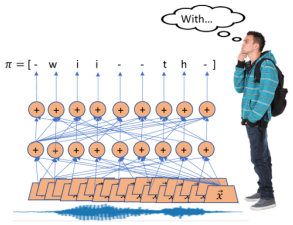
- In a recurrent neural network (RNN), the hidden node activation vector, h_t, depends on the value of the same vector at time $t-1$.
- From 2014-2017, the best speech recognition and machine translation used RNNs.
- The input is x_t=speech or input-language text
- The output is o_t=text in the target language

Example: Part of speech tagging



- x_t =vector representation of the t[th] word, e.g., trained using CBOW
- h_t =hidden state vector
- $o_t = \text{softmax}(h_t^T x_t) = [P(Y_t = \text{Noun}|X_1, \ldots, X_t), P(Y_t = \text{Verb}|X_1, \ldots, X_t), \ldots]$

The Transformer: "Attention is all you need"



- In 2017, researchers proposed that the short-term memory buffer should contain raw signals, not processed signals.
- All processing is done using a model of bottom-up attention.

Under what circumstances is a difference-of-Gaussians filter more useful for edge detection than a simple pixel difference?

Solution: A difference-of-Gaussians filter first smooths the input image (using a Gaussian smoother), then computes a pixel difference. The smoothing step can reduce random noise. Therefore, this procedure is more useful if the input image has some random noise in it.

In a context-free grammar, what is a terminal symbol? What is a non-terminal symbol?

Solution: A terminal symbol is one that is observed in the data, for example, a word. A non-terminal is a generalization, for example, a phrase or a part of speech, that is never observed in the data, but that can generate one or more terminals.

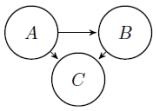
An image classification algorithm is being trained using the multiclass perceptron learning rule. There are 10 classes, each parameterized by a weight vector $\vec{w}_k$, for $0 \le k \le 9$. During the last round of training, all of the training tokens were correctly classified. Which of the weight vectors were updated, and why?

Solution: None. The perceptron learning rule updates the weight vectors only if the classifier makes a mistake.

What is a linear classifier?

Solution: A classifier that computes its decision based on a linear function of the data. If the feature vector is $(x_1, \ldots, x_d)$, then the classification function is given by $\text{sgn}(w_1 x_1 + \ldots + w_d x_d + b)$.

Consider the following Bayes network (all variables are binary):



$P(A) = 0.4$

A	$P(B\|A)$
False	0.1
True	0.2

A, B	$P(C\|A, B)$
False,False	0.9
False,True	0.3
True,False	0.7
True,True	0.5

(a) What is $P(C)$? Write your answer in numerical form, but you don't need to simplify.

Solution:

$$\begin{aligned} P(C) &= P(\neg A, \neg B, C) + P(\neg A, B, C) + P(A, \neg B, C) + P(A, B, C) \\ &= (0.6)(0.9)(0.9) + (0.6)(0.1)(0.3) + (0.4)(0.8)(0.7) + (0.4)(0.2)(0.5) \end{aligned}$$

(b) What is $P(A|B = True, C = True)$? Write your answer in numerical form, but you don't need to simplify.

Solution:

$$\begin{aligned} P(A|B, C) &= \frac{P(A, B, C)}{P(A, B, C) + P(\neg A, B, C)} \\ &= \frac{(0.4)(0.2)(0.5)}{(0.4)(0.2)(0.5) + (0.6)(0.1)(0.3)} \end{aligned}$$

Observation	A	B	C
1	True	True	False
2	False	True	True
3	False	True	False
4	False	False	True

(a) Given the data in the table, what are the maximum likelihood estimates of the model parameters? If there is a model parameter that cannot be estimated from these data, mark it "UNKNOWN."

Solution:

if Laplace smoothing

$$\begin{aligned} P(A) &= 1/4 \\ P(B|\neg A) &= 2/3 \\ P(B|A) &= 1/1 \\ P(C|\neg A, \neg B) &= 1/1 \\ P(C|\neg A, B) &= 1/2 \\ P(C|A, \neg B) &= UNKNOWN \\ P(C|A, B) &= 0/1 \end{aligned}$$

$$\begin{aligned} P(A) &= 2/6 \\ P(B|\neg A) &= 3/5 \\ P(B|A) &= 2/3 \\ P(C|\neg A, \neg B) &= 2/3 \\ P(C|\neg A, B) &= 2/4 \\ P(C|A, \neg B) &= 1/2 \\ P(C|A, B) &= 1/3 \end{aligned}$$

Consider the following probabilistic context-free grammar:

S→	NP VP	$P = 1.0$
NP→	N	$P = 0.9$
NP→	J N	$P = 0.1$
VP→	V	$P = 0.3$
VP→	V NP	$P = 0.7$
J→	beautiful	$P = 0.4$
J→	complicated	$P = 0.6$
N→	birds	$P = 0.8$
N→	flowers	$P = 0.2$
V→	enjoy	$P = 0.5$
V→	grow	$P = 0.5$

Consider the sentence

"Complicated flowers enjoy."

What is the probability that this sentence would be generated by the grammar shown above? (Ignore capitalization and punctuation.)

Solution: The rules that fire are

S→	NP VP	$P = 1.0$
NP→	J N	$P = 0.1$
J→	complicated	$P = 0.6$
N→	flowers	$P = 0.2$
VP→	V	$P = 0.3$
V→	enjoy	$P = 0.5$

The product of these probabilities is

$$P = (1.0)(0.1)(0.6)(0.2)(0.3)(0.5)$$

Consider a Nave Bayes classifier with 100 feature dimensions. The label Y is binary with $P(Y = 0) = P(Y = 1) = 0.5$. All features are binary, and have the same conditional probabilities: $P(X_i = 1|Y = 0) = a$ and $P(X_i = 1|Y = 1) = b$ for $i = 1, \ldots, 100$. Given an item X with alternating feature values ($X_1 = 1, X_2 = 0, X_3 = 1, ..., X_{100} = 0$), compute $P(Y = 1|X)$.

Solution:

$$\begin{aligned} P(Y = 1|X) &= \frac{P(Y = 1) \prod_{i=1}^{100} P(X_i|Y = 1)}{P(Y = 1) \prod_{i=1}^{100} P(X_i|Y = 1) + P(Y = 0) \prod_{i=1}^{100} P(X_i|Y = 0)} \\ &= \frac{0.5b^{50}(1-b)^{50}}{0.5b^{50}(1-b)^{50} + 0.5a^{50}(1-a)^{50}} \\ &= \frac{b^{50}(1-b)^{50}}{b^{50}(1-b)^{50} + a^{50}(1-a)^{50}} \end{aligned}$$

A particular hidden Markov model (HMM) has state variable X_t, and observation variables E_t, where t denotes time. Suppose that this HMM has two states, $X_t \in \{0, 1\}$, and three possible observations, $E_t \in \{0, 1, 2\}$. The initial state probability is $P(X_1 = 1) = 0.3$. The transition and observation probability matrices are

X_{t-1}	$P(X_t = 1\|X_{t-1})$	X_t	$P(E_t = 0\|X_t)$	$P(E_t = 1\|X_t)$
0	0.6	0	0.4	0.1
1	0.4	1	0.1	0.6

Suppose that, in a particular test of the HMM, the observation sequence is

$$\{E_1, E_2\} = \{2, 1\}$$

(a) What is the total probability $P(E_1 = 2, E_2 = 1)$?

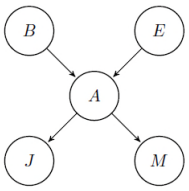
Solution:

$$\begin{aligned} P(E_1 = 2, E_2 = 1) &= \sum_{X_1, X_2} P(X_1)P(E_1 = 2|X_1)P(X_2|X_1)P(E_2 = 1|X_2) \\ &= (0.7)(0.5)(0.4)(0.1) + (0.7)(0.5)(0.6)(0.6) + (0.3)(0.3)(0.6)(0.1) + (0.3)(0.3)(0.4)(0.6) \end{aligned}$$

(b) If it is observed that $X_2 = 1$, what is the most likely value of X_1? In other words, what is $\arg\max_{X_1} P(X_1, E_1 = 2, X_2 = 1, E_2 = 1)$?

$$\begin{aligned} \arg\max_{X_1} P(X_1, E_1 = 2, X_2 = 1, E_2 = 1) &= \arg\max_{X_1} P(X_1)P(E_1 = 2|X_1)P(X_2 = 1|X_1) \\ &= \arg\max_{0,1} ((0.7)(0.5)(0.6), (0.3)(0.3)(0.4)) \\ &= 0 \end{aligned}$$

Consider the "Burglary" Bayesian network:



(a) How many independent parameters does this network have? How many entries does the full joint distribution table have?

Solution: There are five binary variables: two with no parents (B and E, one parameter each), one with two parents (A, four parameters), and two with one parent each (J and M, two parameters each), for a total of ten independent parameters. The full joint distribution table has $2^5 - 1 = 31$ parameters.

(b) If no evidence is observed, are B and E independent?

Solution: Yes, because they have no common ancestors.

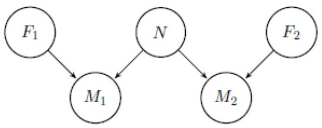
(c) Are B and E conditionally independent given the observation that A =True?

Solution: No. Knowing that Earthquake=True makes Burglary less probable.

Two astronomers in different parts of the world make measurements M_1 and M_2 of the number of stars N in some small region of the sky, using their telescopes. Under normal circumstances, this experiment has three possible outcomes: either the measurement is correct, or the measurement overcounts the stars by one (one star too high), or the measurement undercounts the stars by one (one star too low). There is also the possibility, however, of a large measurement error in either telescope (events F_1 and F_2, respectively), in which case the measured number will be **at least** three stars too low (regardless of whether the scientist makes a small error or not), or, if N is less than 3, fail to detect any stars at all.

(a) Draw a Bayesian network for this problem.

Solution: A solution must include the variables N, M_1, M_2 with the dependencies shown below. The variables F_1, F_2 are optional:



(b) Write out a conditional distribution for $P(M_1|N)$ for the case where $N \in \{1, 2, 3\}$ and $M_1 \in \{0, 1, 2, 3, 4\}$. Each entry in the conditional distribution table should be expressed as a function of the parameters e and/or f.

Solution:

	M_1				
N	0	1	2	3	4
1	$e + f$	$1 - 2e - f$	e	0	0
2	f	e	$1 - 2e - f$	e	0
3	f	0	e	$1 - 2e - f$	e

(c) Suppose $M_1 = 1$ and $M_2 = 3$. What are the possible numbers of stars if you assume no prior constraint on the values of N?

Solution: $N = 2$ is possible, if both made small mistakes. $N = 4$ is possible, if M_2 made a small and M_1 a big mistake. $N \geq 6$ is possible, if both M_1 and M_2 made big mistakes.

(d) What is the most likely number of stars, given the observations $M_1 = 1, M_2 = 3$? Explain how to compute this, or if it is not possible to compute, explain what additional information is needed and how it would affect the result.

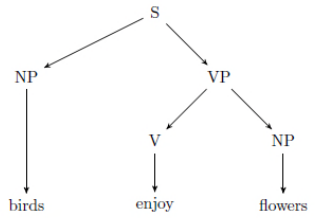
Solution: We need to find the value of N that maximizes $P(N, M_1 = 1, M_2 = 3)$. We have that $P(N = 2, M_1 = 1, M_2 = 3) = P(N = 2)e^2$. We know that $P(N = 4, M_1 = 1, M_2 = 3) \leq P(N = 4)fe$; we don't know exactly how much it is, because we don't know $P(M_1 = 1|N = 4)$, but we know that $P(M_1 = 1|N = 4) \leq f$. So if $P(N = 2)e > P(N = 4)f$, $N = 2$ is the most probable value. If $P(N = 2)e \leq P(N = 4)f$, then it depends on the way in which big errors are distributed among the various values that are "at least three stars" too small.

Consider the following probabilistic context-free grammar:

S→	NP VP	$P = 1.0$
NP→	birds	$P = 0.5$
NP→	flower	$P = 0.5$
VP→	V	$P = 0.5$
VP→	V NP	$P = 0.5$
V→	enjoy	$P = 0.5$
V→	grow	$P = 0.5$

(a) Draw a tree showing how the S nonterminal can produce the sentence "birds enjoy flowers".

Solution:



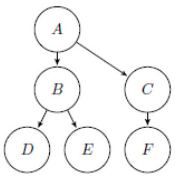
(b) What is the probability, according to this model, of the sentence "birds enjoy flowers"?

Solution:

S→	NP VP	$P = 1.0$
NP→	birds	$P = 0.5$
NP→	V NP	$P = 0.5$
V→	enjoy	$P = 0.5$
NP→	flower	$P = 0.5$

Multiplying the probabilities of the five production rules, we get $P = 1/16$.

Consider the following Bayes network (all variables are binary):



$P(A) = 0.8$

A	$P(B\|A)$	$P(C\|A)$
0	0.2	0.6
1	0.5	0.8

B	$P(D\|B)$	$P(E\|B)$
0	0.5	0.8
1	0.5	0.8

C	$P(F\|C)$
0	0.01
1	0.2

(a) Are D and E independent?

Solution: Yes. This is a trick question. The structure of the Bayes net shows them to be conditionally independent given B, but not independent. However, in the probability table, notice that $P(D|B) = P(D|\neg B)$, therefore D is independent of B, despite the arrow shown in the Bayes net. Similarly, $P(E|B) = P(E|\neg B)$, therefore E is independent of B, despite the arrow shown in the Bayes net. Since there is no other path connecting D to E except the one going through B, they are independent.

(b) Are D and E conditionally independent given B?

Solution: Yes. This is not a trick question. The structure of the Bayes net shows that they are conditionally independent given B.

(c) If you did not know the Bayesian network, how many numbers would you need to represent the full joint probability table?

Solution: There are 2^6 possible combinations of 6 binary variables, so you'd need $2^6 - 1 = 63$ numbers.

(d) If you knew the Bayes network as shown above, but the variables were ternary instead of binary, how many values would you need to represent the full joint probability table and the conditional probability tables, respectively?

Solution: Conditional probability tables: For each variable, the number of trainable parameters is (# possible values of the variable, minus 1)×(# possible values of its parents). $P(A)$ would need 2 trainable parameters, each of the other five variables would need $2 \times 3 = 6$ trainable parameters, for a total of $2 + 5 \times 2 \times 3 = 32$ trainable parameters.

Full joint probability table: there are 3^6 possible combinations of the variables, so you would need to store $3^6 - 1$ parameters.

(e) Write down the expression for the joint probability of all the variables in the network, in terms of the model parameters given above.

Solution:

$$P(A,B,C,D,E,F) = P(A)P(B|A)P(C|A)P(D|B)P(E|B)P(F|C)$$

(f) Find $P(A = 0, B = 1, C = 1, D = 0)$.

Solution:

$$P(A = 0, B = 1, C = 1, D = 0) = (0.2)(0.2)(0.6)(0.5) = \frac{3}{250}$$

(g) Find $P(B|A = 1, D = 0)$.

Solution:

$$P(B|A = 1, D = 0) = \frac{P(A = 1, B = 1, D = 0)}{P(A = 1, B = 1, D = 0) + P(A = 1, B = 0, D = 0)} = \frac{(0.8)(0.5)(0.5)}{(0.8)(0.5)(0.5) + (0.8)(0.5)(0.5)} = \frac{1}{2}$$

The real world contains two parallel infinite-length lines, whose equations, in terms of the coordinates (x, y, z), are parameterized as $ax + by + cz = d$ and $ax + by + cz = e$; in addition, both of these lines are on the ground plane, $y = g$, for some constants (a, b, c, d, e, g). Show that the images of these two lines, as imaged by a pinhole camera, converge to a vanishing point, and give the coordinates (x', y') of the vanishing point.

Solution: The pinhole camera equations are

$$x' = \frac{-fx}{z}, \quad y' = \frac{-fy}{z}$$

From which we derive

$$x = \frac{-zx'}{f}, \quad y = \frac{-zy'}{f}$$

So the equations of the two lines are

$$-\frac{ax'}{f} - \frac{by'}{f} + c = \frac{d}{z}$$
$$-\frac{ax'}{f} - \frac{by'}{f} + c = \frac{e}{z}$$

As $z \to \infty$, the right-hand-sides of these two equations both go to zero, and the equations of both lines converge to

$$ax' + by' = cf$$

In addition, we have $y = g$, so $y' = -fg/z \to 0$, and therefore $x' = cf/a$. The coordinates are $(x', y') = (cf/a, 0)$.

In a pinhole camera, a light source at (x, y, z) is projected onto a pixel at $(x', y', -f)$ through a pinhole at $(0, 0, 0)$. Write $\sqrt{(x')^2 + (y')^2}$ in terms of x, y, z, and f.

Solution: The pinhole camera equations are

$$x' = \frac{-fx}{z}, \quad y' = \frac{-fy}{z}$$

from which we derive

$$\sqrt{(x')^2 + (y')^2} = \frac{f}{z}\sqrt{x^2 + y^2}$$

Consider a hidden Markov model (HMM) whose hidden variable denotes part of speech (POS), $X_t \in \{N, V\}$ where N =noun, V =verb, the initial state probability is $P(X_1 = N) = 0.8$, and the transition probabilities are $P(X_t = N|X_{t-1} = N) = 0.1$ and $P(X_t = V|X_{t-1} = V) = 0.1$. Suppose we have the observation probability matrix given in Table 1.

E_t	rose	bill	likes
$P(E_t\|X_t = N)$	0.4	0.4	0.2
$P(E_t\|X_t = V)$	0.2	0.2	0.6

Table 1: Observation probabilities for a simple POS HMM.

You are given the sentence "bill rose." You want to figure out whether each of these two words, "bill" and "rose", is being used as a noun or a verb.

(a) List the four possible combinations of (X_1, X_2). For each possible combination, give $P(X_1, E_1, X_2, E_2)$.

Solution:

$P(X_1, E_1, X_2, E_2)$	$X_2 = N$	$X_2 = V$
$X_1 = N$	(0.8)(0.4)(0.1)(0.4)	(0.8)(0.4)(0.9)(0.2)
$X_1 = V$	(0.2)(0.2)(0.9)(0.4)	(0.2)(0.2)(0.1)(0.2)

(b) Find $P(X_2 = V|E_1 = \text{bill}, E_2 = \text{rose})$.

Solution: Using the forward algorithm, we can compute the joint probabilities as

$$\begin{aligned} P(E, X_2 = V) &= P(X_1 = N, E_1, X_2 = V, E_2) + P(X_1 = V, E_1, X_2 = V, E_2) \\ &= (0.8)(0.4)(0.9)(0.2) + (0.2)(0.2)(0.1)(0.2) \\ P(E, X_2 = N) &= P(X_1 = N, E_1, X_2 = N, E_2) + P(X_1 = V, E_1, X_2 = N, E_2) \\ &= (0.8)(0.4)(0.1)(0.4) + (0.2)(0.2)(0.9)(0.4) \end{aligned}$$

Dividing the first row by the sum of the two rows, we get

$$P(X_2 = V|E) = \frac{(0.8)(0.4)(0.9)(0.2) + (0.2)(0.2)(0.1)(0.2)}{(0.8)(0.4)(0.9)(0.2) + (0.2)(0.2)(0.1)(0.2) + (0.8)(0.4)(0.1)(0.4) + (0.2)(0.2)(0.9)(0.4)}$$

(c) Use the Viterbi algorithm to find the most likely state sequence for this sentence.

Solution:

- To find the backpointer from $X_2 = N$, we find the maximum among the two possibilities $P(X_1 = N, E_1, X_2 = N, E_2)$ and $P(X_1 = V, E_1, X_2 = N, E_2)$. The larger of the two is $P(X_1 = V, E_1, X_2 = N, E_2) = (0.2)(0.2)(0.9)(0.4)$, so the backpointer from $X_2 = N$ points to $X_1 = V$.
- To find the backpointer from $X_2 = V$, we find the maximum among the two possibilities $P(X_1 = N, E_1, X_2 = V, E_2)$ and $P(X_1 = V, E_1, X_2 = V, E_2)$. The larger of the two is $P(X_1 = N, E_1, X_2 = V, E_2) = (0.8)(0.4)(0.9)(0.2)$, so the backpointer from $X_2 = V$ points to $X_1 = N$.
- To find the best terminal state, then, we find the maximum among the two possibilities $P(X_1 = V, E_1, X_2 = N, E_2)$ and $P(X_1 = N, E_1, X_2 = V, E_2)$. The larger of the two is $P(X_1 = N, E_1, X_2 = V, E_2) = (0.8)(0.4)(0.9)(0.2)$, so the maximum likelihood state sequence is $(X_1, X_2) = (N, V)$.

Question 5 *(5 points)*

Gradient descent is guaranteed to find a set of model parameters that has the smallest possible loss function on the training corpus.

○ True

√ False

Explain:

Solution: Gradient descent finds a local optimum of the loss function, but it is not guaranteed to find a global optimum.

Question 6 *(5 points)*

A naive Bayes classifier can be implemented as a linear classifier.

√ True

○ False

Explain:

Solution: The log of the naive Bayes probability can be written as

$$\ln P(X = x, F_1 = f_1, \ldots) = \beta + \sum_{i=1}^{N} [F_i = f_i] w_i$$

where $\beta = \ln P(X = x)$ is the prior, $w_i = \ln P(F_i = f_i | X = x)$, and [Proposition] is the unit indicator function (equal to one if Proposition is true, equal to zero otherwise).

You have a two-layer neural network trained as an animal classifier. The input feature vector is $\vec{x} = [x_1, x_2, x_3, 1]$, where x_1, x_2, and x_3 are some features, and 1 is multiplied by the bias. There are two hidden nodes, and three output nodes, $\vec{y}^* = [y_1^*, y_2^*, y_3^*]$, corresponding to the three output classes $y_1^* = \Pr(\text{dog}|\vec{x})$, $y_2^* = \Pr(\text{cat}|\vec{x})$, $y_3^* = \Pr(\text{skunk}|\vec{x})$. The hidden layer uses a sigmoid nonlinearity, the output layer uses a softmax.

(a) (2 points) A Maltese puppy has the feature vector $\vec{x} = [2, 20, -1, 1]^T$. Suppose all weights and biases are initialized to zero. What is $\vec{y}^*$?

Solution: If all weights and biases are zero, then the excitation of each hidden node is $0 \times 2 + 0 \times 20 + 0 \times (-1) + 0 \times 1 = 0$. With zero input, the sigmoid $1/(1 + \exp(-f)) = 0.5$, but weights in the last layer are also all zero, so the excitations at the last layer are all zero. With a softmax nonlinearity, every output node is computing $\exp(0)/\sum_{i=1}^{3} \exp(0) = 1/3$. So

$$\vec{y}^* = \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right]$$

(b) (3 points) Let w_{ij} be the weight connecting the i^{th} output node to the j^{th} hidden node. What is dy_2^*/dw_{21}? Write your answer in terms of y_i^*, w_{ij}, and/or the hidden node activations h_j, for any appropriate values of i and/or j.

Solution: Let's use the notation f_i as the excitation of the i^{th} output node. That allows us to write the softmax as:

$$y_2^* = \frac{\exp(f_2)}{\sum_{j=1}^{3} \exp(f_j)}, \quad f_j = \sum_i w_{ji} h_i$$

Then:

$$\begin{aligned}
\frac{dy_2^*}{dw_{21}} &= \frac{1}{\sum_{i=1}^{3} \exp(f_i)} \frac{d\exp(f_2)}{dw_{21}} + \exp(f_2) \frac{d(1/\sum_i \exp(f_i)))}{dw_{21}} \\
&= \frac{1}{\sum_{i=1}^{3} \exp(f_i)} \exp(f_2) \frac{df_2}{dw_{21}} + \exp(f_2) \left(-\frac{1}{(\sum_{i=1}^{3} \exp(f_i))^2} \right) \frac{d(\sum \exp(f_i)))}{dw_{21}} \\
&= \frac{\exp(f_2)}{\sum_{i=1}^{3} \exp(f_i)} h_1 - \frac{\exp(f_2)}{(\sum_{i=1}^{3} \exp(f_i))^2} \frac{d\exp(f_2)}{dw_{21}} \\
&= \frac{\exp(f_2)}{\sum_{i=1}^{3} \exp(f_i)} h_1 - \frac{\exp(f_2)}{(\sum_{i=1}^{3} \exp(f_i))^2} \exp(f_2) \frac{df_2}{dw_{21}} \\
&= \frac{\exp(f_2)}{\sum_{i=1}^{3} \exp(f_i)} h_1 - \frac{\exp(f_2)^2}{(\sum_{i=1}^{3} \exp(f_i))^2} h_1 \\
&= y_2^* h_1 - (y_2^*)^2 h_1
\end{aligned}$$

Question 13 *(5 points)*

How can randomness be incorporated into a game tree? How about partial observability (imperfect information)?

Solution: Randomness is incorporated using the expectiminimax algorithm, in which max tries to maximize the expected score, min tries to minimize the expected score. Partial observability is incorporated using a minimax state tree in which neither player knows for sure which state they're in; the max player chooses an action that maximizes the minimum payoff over all of the states he might be in.

Question 11 *(5 points)*

In a Markov Decision Process with finite state and action sets, model-based reinforcement learning needs to learn a larger number of trainable parameters than model-free reinforcement learning.

√ True

○ False

Explain:

Solution: Model-based learning needs to learn $P(s'|s,a)$, a set of $N_s^2 N_a$ parameters, where N_s is the number of states, N_a the number of actions. Model-free learning needs to learn $Q(s,a)$, a set of only $N_s N_a$ trainable parameters.

Question 12 *(5 points)*

When we apply the Q-learning algorithm to learn the state-action value function, one big problem in practice may be that the state space of the problem is continuous and high-dimensional. Discuss at least two possible methods to address this.

Solution:

1. Discretize the state space.
2. Design a lower-dimensional set of discrete features to represent the states.
3. Use a parametric approximator (e.g., a neural network) to estimate the Q function values and learn the parameters instead of directly learning the state-action value functions.

You're creating sentiment analysis. You have a training corpus with four movie reviews:

Review #	Sentiment	Review
1	+	what a great movie
2	+	I love this film
3	-	what a horrible movie
4	-	I hate this film

Let $Y = 1$ for positive sentiment, $Y = 0$ for negative sentiment.

(a) What's the maximum likelihood estimate of $P(Y = 1)$?

Solution: Maximum likelihood estimate is

$$P(Y = 1) = \frac{\#\text{ times } Y = 1}{\#\text{ training tokens}} = \frac{2}{4} = \frac{1}{2}$$

(b) Find maximum likelihood estimates $P(W|Y = 1)$ and $P(W|Y = 0)$ for the ten words $W \in \{\text{what,a,movie,I,this,film,great,love,horrible,hate}\}$.

Solution: There are three cases. For the words $W \in \{\text{what,a,movie,I,this,film}\}$, $P(W|Y = 0) = P(W|Y = 1) = 1/8$. For the words $W \in \{\text{great,love}\}$, $P(W|Y = 0) = 0$, and $P(W|Y = 1) = 1/8$. For the words $W \in \{\text{horrible,hate}\}$, $P(W|Y = 1) = 0$, and $P(W|Y = 0) = 1/8$.

(c) Use Laplace smoothing, with a smoothing parameter of $k = 1$, to estimate $P(W|Y = 1)$ and $P(W|Y = 0)$ for the ten words $W \in \{\text{what,a,movie,I,this,film,great,love,horrible,hate}\}$.

Solution: There are three cases. For the words $W \in \{\text{what,a,movie,I,this,film}\}$, $P(W|Y = 0) = P(W|Y = 1) = 2/18$. For the words $W \in \{\text{great,love}\}$, $P(W|Y = 0) = 1/18$, and $P(W|Y = 1) = 2/18$. For the words $W \in \{\text{horrible,hate}\}$, $P(W|Y = 1) = 1/18$, and $P(W|Y = 0) = 2/18$.

(d) Using some other method (unknown to you), your professor has estimated the following conditional probability table:

Y	$P(\text{great}\|Y)$	$P(\text{love}\|Y)$	$P(\text{horrible}\|Y)$	$P(\text{hate}\|Y)$
1	0.01	0.01	0.005	0.005
0	0.005	0.005	0.01	0.01

and $P(Y = 1) = 0.5$. All other words (except great, love, horrible, and hate) can be considered out-of-vocabulary, and you can assume that $P(W|Y) = 1$ for all out-of-vocabulary words. Under these assumptions, what is the probability $P(Y = 1|R)$ that the following 14-word review is a positive review?

$R = \{$"I'm horrible fond of this movie, and I hate anyone who insults it."$\}$

Solution:

$$P(Y = 1|R) = \frac{P(Y = 1, R)}{P(Y = 1, R) + P(Y = 0, R)} = \frac{(0.5)(0.005)(0.005)}{(0.5)(0.005)(0.005) + (0.5)(0.01)(0.01)} = \frac{1}{5}$$

a:	purr		walk	
s:	1	2	1	2
$P(s' = 1\|s,a)$	2/3	1/3	1/3	2/3
$P(s' = 2\|s,a)$	1/3	2/3	2/3	1/3

The cat decides to use policy iteration to find a new optimal policy under this model. It starts with the following policy: $\pi(1) = \text{purr}$, $\pi(2) = \text{walk}$. Now it needs to find the policy-dependent utility, $U^\pi(s)$. Again, because the cat doesn't care about the distant future, it uses a relatively low discount factor ($\gamma = 3/4$). Write two linear equations that can be solved to find the two unknowns $U^\pi(1)$ and $U^\pi(2)$; your equations should have no variables in them other than $U^\pi(1)$ and $U^\pi(2)$.

Solution: The two equations are

$$U^\pi(1) = R(1) + \frac{3}{4} \sum_{s'} P(s'|1, \pi(1)) U^\pi(s')$$

$$U^\pi(2) = R(2) + \frac{3}{4} \sum_{s'} P(s'|2, \pi(2)) U^\pi(s')$$

Plugging in the given values of all variables, we have

$$U^\pi(1) = 2 + \frac{3}{4} \left(\frac{2}{3} U^\pi(1) + \frac{1}{3} U^\pi(2) \right)$$

$$U^\pi(2) = 5 + \frac{3}{4} \left(\frac{2}{3} U^\pi(1) + \frac{1}{3} U^\pi(2) \right)$$

d) (2 points) Since it has some extra time, and excellent python programming skills, the cat decides to implement deep reinforcement learning, using an actor-critic algorithm. Inputs are one-hot encodings of state and action. What are the input and output dimensions of the actor network, and of the critic network?

Solution: The actor network takes a state as input, thus its input dimension is 2 (if the input is a one-hot encoding of two states). It computes the probability that any given action is the best action, so its output dimension is 2 (if there are two possible actions). The critic takes, as input, an encoding of the state (two dimensions), and an encoding of the action (two dimensions, if the action is a one-hot encoding of two possible actions), for a total of 4 input dimensions. It computes, as output, a real-valued score $Q(s,a)$, which is a 1-dimensional (scalar) output.

Question 7 *(5 points)*

We want to implement a classifier that takes two input values, where each value is either 0, 1 or 2, and outputs a 1 if at least one of the two inputs has value 2; otherwise it outputs a 0. Can this function be learned by a Perceptron? If so, construct a Perceptron that does it; if not, why not.

Solution: In this case the input space of all possible examples with their target outputs is:

	0	1	2
2	1	1	1
1	0	0	1
0	0	0	1

Since there is clearly no line that can separate the two classes, this function is not linearly separable and so it cannot be learned by a Perceptron.

Question 8 *(5 points)*

After t iterations of the "Value Iteration" algorithm, the estimated utility $U(s)$ is a summation including terms $R(s')$ for the set of states s' that can be reached from state s in at most $t-1$ steps.

√ **True**

○ False

Explain:

Solution: Value iteration starts with $U(s) = 0$. Each iteration updates $U(s)$ by adding $R(s)$, plus the maximum over all actions of the expected utility $U(s')$ of the state s' that can be reached from state s in one step. In t iterations of this algorithm, one accumulates rewards from states that are up to $t-1$ steps away.

9. (2½ points) In the tree search formulation, why do we restrict step costs to be non-negative?

Solution: If the cost around any loop is negative, then the lowest-cost path is to take that loop an infinite number of times.

10. (2½ points) What is the distinction between a world state and a search tree node?

Solution: A world state contains enough information to know (1) whether or not you've reached the goal, (2) what actions can be performed, (3) what will be the result of each action. A search tree node contains a pointer to the world state, plus a pointer to the parent node.

11. (2½ points) How do we avoid repeated states during tree search?

Solution: By keeping a set of "explored states." If expanding a search node results in a state that has already been explored, we don't add it to the frontier.

13. (2½ points) What are the main challenges of adversarial search as contrasted with single-agent search? What are some algorithmic similarities and differences?

Solution: The biggest difference is that we are unaware of how the opponent(s) will act. Because of this our search cannot simply consider my own moves, it must also figure out how my opponent will act at each level, thus effectively doubling the number of levels over which I have to search. Since the number of levels is the exponent in the computational complexity, this makes computational complexity much harder.

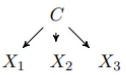
14. (2½ points) What additional difficulties does dice throwing or other sources of uncertainty introduce into a game?

Solution: Uncertainties introduce probabilities into the game. Expectiminimax is used to find solutions for these type of games. Expectiminimax doubles the number of levels as compared to minimax, because there is randomness after every play. Expectiminimax has nasty branching factor and often times defining evaluation functions and pruning algorithms are difficult.

We have a bag of three biased coins, a, b, and c, with probabilities of coming up heads of 20%, 60%, and 80%, respectively. One coin is drawn randomly from the bag (with equal likelihood of drawing each of the three coins), and then the coin is flipped three times to generate the outcomes X1, X2, and X3.

(a) Draw the Bayesian network corresponding to this setup and define the necessary conditional probability tables (CPTs).

Solution: You need an intermediate variable, $C \in \{a, b, c\}$, to specify which coin is drawn, then the graph is



and the CPTs are

C	$P(C)$	$P(X_1 = H\|C)$	$P(X_2 = H\|C)$	$P(X_3 = H\|C)$
a	1/3	0.2	0.2	0.2
b	1/3	0.6	0.6	0.6
c	1/3	0.8	0.8	0.8

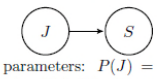
(b) Calculate which coin was most likely to have been drawn from the bag if the observed flips come out heads twice and tails once.

Solution:

$$P(C = a, HHT) = (0.2)(0.2)(0.8)/3 = 32/3000$$
$$P(C = b, HHT) = (0.6)(0.6)(0.4)/3 = 144/3000$$
$$P(C = c, HHT) = (0.8)(0.8)(0.2)/3 = 128/3000$$

The maximum-posterior-probability event is also the maximum-joint-probability event, which is the event $C = b$.

Suppose you have a Bayes net with two binary variables, Jahangir (J) and Shahjahan (S):



This network has three trainable parameters: $P(J) = a$, $P(S|J) = b$, and $P(S|\neg J) = c$. Suppose you have a training dataset in which S is observed, but J is hidden. Specifically, there are N training tokens for which $S =$ True, and M training tokens for which $S =$ False. Given current estimates of a, b, and c, you want to use the EM algorithm to find improved estimates $\hat{a}$, $\hat{b}$, and $\hat{c}$.

(a) Find the following expected counts, in terms of M, N, a, b, and c:

$$E[\#\text{ times } J \text{ True}] =$$
$$E[\#\text{ times } J \text{ and } S \text{ True}] =$$
$$E[\#\text{ times } J \text{ True and } S \text{ False}] =$$

Solution:

$$E[\#\text{ times } J \text{ True}] = \frac{abN}{ab+(1-a)c} + \frac{a(1-b)M}{a(1-b)+(1-a)(1-c)}$$
$$E[\#\text{ times } J \text{ and } S \text{ True}] = \frac{abN}{ab+(1-a)c}$$
$$E[\#\text{ times } J \text{ True and } S \text{ False}] = \frac{a(1-b)M}{a(1-b)+(1-a)(1-c)}$$

(b) Find re-estimated values $\hat{a}$, $\hat{b}$, and $\hat{c}$ in terms of M, N, $E[\#\text{ times } J \text{ True}]$, $E[\#\text{ times } J \text{ and } S \text{ True}]$, and $E[\#\text{ times } J \text{ True and } S \text{ False}]$.

Solution:

$$\hat{a} = \frac{E[\#\text{ times } J \text{ True}]}{M+N}$$
$$\hat{b} = \frac{E[\#\text{ times } J \text{ and } S \text{ True}]}{E[\#\text{ times } J \text{ True}]}$$
$$\hat{c} = \frac{E[\#\text{ times } J \text{ False and } S \text{ True}]}{M+N-E[\#\text{ times } J \text{ False}]}$$

Question 25

Consider the convolution equation

$$Z(x', y') = \sum_m \sum_n h(m, n) Y(x' - m, y' - n)$$

Where $Y(x', y')$ is the original image, $Z(x', y')$ is the filtered image, and the filter $h(m, n)$ is given by

$$h(m,n) = \begin{cases} \frac{1}{21} & 1 \le m \le 3, \quad -3 \le n \le 3 \\ -\frac{1}{21} & -3 \le m \le -1, \quad -3 \le n \le 3 \end{cases}$$

Would this filter be more useful for smoothing, or for edge detection? Why?

Solution: The sum of $h(m, n)$, over all m and n, is 0. So if it is filtering a constant-color region, the output would always be zero, regardless of the input color. So it's not very useful for smoothing.

Any given pixel of $Z(x', y')$ is the difference between the pixels $Y(x', y')$ to its left, minus those to its right. Since it's computing a difference, it would be useful for edge detection.

Question 22

In a pinhole camera, a light source at (x, y, z) is projected onto a pixel at $(x', y', -f)$ through a pinhole at $(0, 0, 0)$. Write $\sqrt{(x')^2 + (y')^2}$ in terms of x, y, z, and f.

Solution: From the idea of similar triangles, we have

$$\frac{x'}{f} = -\frac{x}{z}, \quad \frac{y'}{f} = -\frac{y}{z}$$

from which we derive

$$\sqrt{(x')^2 + (y')^2} = \frac{f}{z}\sqrt{x^2 + y^2}$$

Question 1

What are the main challenges of adversarial search as contrasted with single-agent search? What are some algorithmic similarities and differences?

Solution: The biggest difference is that we are unaware of how the opponent(s) will act. Because of this our search cannot simply consider my own moves, it must also figure out how my opponent will act at each level, thus effectively doubling the number of levels over which I have to search. Since the number of levels is the exponent in the computational complexity, this makes computational complexity much harder.

Question 24

A particular HMM has a binary state variable $X_t \in \{0, 1\}$, and a binary observation variable $E_t \in \{0, 1\}$. Suppose the HMM starts at time $t = 1$ with initial probability $P(X_1 = 1) = 0.2$. The transition probabilities and observation probabilities are given in the following table:

		$P(X_{t+1}=1\|X_t)$	$P(E_t=1\|X_t)$
X_t	0	0.3	0.8
	1	0.3	0.9

What is $P(X_1 = 0, E_1 = 0, E_2 = 0)$? Express your answer as a sum of products; do not simplify.

Solution:

$$(0.8)(0.2)(0.7)(0.2) \mid (0.8)(0.2)(0.3)(0.1)$$

Suppose you have $n = 2$ training samples, $x_1 = [0.2, 0.6]^T$, $y_1 = 1$, $x_2 = [-1.2, 0.3]^T$, $y_2 = 0$. Suppose $P(Y = 1|x)$ is defined as $P(Y = 1|x) = \sigma(w^T x)$, where $\sigma(\cdot)$ is the logistic sigmoid function. This computation has already been performed, therefore you already know that $P(Y = 1|x_1) = 0.2$, $P(Y = 1|x_2) = 0.9$. Find the gradient of $\mathcal{L}$ with respect to the vector w.

Solution:

$$\nabla_w \mathcal{L} = -\frac{1}{2}[(0.8)(0.2) - (0.9)(-1.2), (0.8)(0.6) - (0.9)(0.3)]^T$$