● True ○ False: Let $A$ be the set of all actions and $S$ the set of states for some MDP. Assuming that $|A| \ll |S|$, one iteration of value iteration is generally faster than one iteration of policy iteration that solves a linear system during policy evaluation. One iteration of value iteration is $O(|S|^2|A|)$, whereas one iteration of policy iteration is $O(|S|^3)$, so value iteration is generally faster when $|A| \ll |S|$

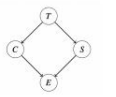
○ True ● False: For any MDP, changing the discount factor does not affect the optimal policy for the MDP. Consider an infinite horizon setting where we have 2 states $A$, $B$, where we can alternate between $A$ and $B$ forever, gaining a reward of 1 each transition, or exit from $B$ with a reward of 100. In the case that $\gamma = 1$, the optimal policy is to forever oscillate between $A$ and $B$. If $\gamma = \frac{1}{2}$, then it is optimal to exit.

## Variable Elimination

● $Q^\pi(s,a) \le Q^*(s,a)$ for all $s, a$

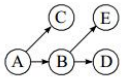
An alternate approach to inference by enumeration, where we eliminate variables one by one in order to reduce the maximum size of any factor generated.

Suppose we have the following Bayes Net with factors $P(T)$, $P(C|T)$, $P(S|T)$, and $P(E|C,S)$, and we want to compute $P(T|+e)$.




We proceed by eliminating the variables $C$ and $S$:

1. Join (multiply) all the factors involving $C$, forming $P(C,+e|T,S) = P(C|T) \cdot P(+e|C,S)$.
2. Sum out $C$ from this new factor, leaving us with a new factor $P(+e|T,S)$.
3. Join all factors involving $S$, forming $P(+e,S|T) = P(S|T) \cdot P(+e|T,S)$.
4. Sum out $S$, yielding $P(+e|T)$.
5. Join the remaining two factors $P(T)$ and $P(+e|T)$ and normalize to get $P(T|+e)$.




| | P(A) |
|---|---|
| +a | 0.25 |
| −a | 0.75 |

| | P(B\|A) | +b | −b |
|---|---|---|---|
| +a | | 0.5 | 0.5 |
| −a | | 0.25 | 0.75 |

| | P(C\|A) | +c |
|---|---|---|
| +a | | 0.2 |
| −a | | 0.6 0.4 |

| | P(D\|B) | +d | −d |
|---|---|---|---|
| +b | | 0.6 | 0.4 |
| −b | | 0.8 | 0.2 |

| | P(E\|B) | +e | −e |
|---|---|---|---|
| +b | | 0.25 | 0.75 |
| −b | | 0.1 | 0.9 |

Using the Bayes' Net and conditional probability tables above, calculate the following quantities:

(i) $P(+b|+a) = 0.5$

(ii) $P(+a,+b) =$
$0.25 * 0.5 = 0.125 = \frac{1}{8}$

(iii) $P(+a|+b) =$
$\frac{0.25*0.5}{0.25*0.5+0.25*0.75} = 0.4 = \frac{2}{5}$

Now we are going to consider variable elimination in the Bayes' Net above.

(i) Assume we have the evidence $+c$ and wish to calculate $P(E \mid +c)$. What factors do we have initially?
$P(A)$, $P(B \mid A)$, $P(+c \mid A)$, $P(D \mid B)$, $P(E \mid B)$

(ii) If we eliminate variable B, we create a new factor. What probability does that factor correspond to?
$P(D, E \mid A)$

(iii) What is the equation to calculate the factor we create when eliminating variable B?
$f(A, D, E) = \sum_B P(B \mid A) * P(D \mid B) * P(E \mid B)$

(iv) After eliminating variable B, what are the new set of factors? As in (ii), write the probabilities that the facto represent. For each factor, also provide its size.

| Factor | Size after elimination |
|---|---|
| $P(A)$ | 2 |
| $P(+c \mid A)$ | 2 |
| $P(D, E \mid A)$ | $2^3$ |

(v) Now assume we have the evidence $-c$ and are trying to calculate $P(A|-c)$. What is the most efficient elimination ordering? If more than one ordering is most efficient, provide any one of them. E, D, B or D, E, B

(vi) Once we have run variable elimination and have $f(A,-c)$ how do we calculate $P(+a \mid -c)$? $\frac{f(+a,-c)}{f(+a,-c)+f(-a,-c)}$ or note that elimination is unnecessary - just use Bayes' rule

Pacman is in an unknown MDP where there are three states [A, B, C] and two actions [Stop, Go]. We are gi the following samples generated from taking actions in the unknown MDP. For the following problems, assu $\gamma = 1$ and $\alpha = 0.5$.

(a) We run Q-learning on the following samples:

| s | a | s' | r |
|---|---|---|---|
| A | Go | B | 2 |
| C | Stop | A | 0 |
| B | Stop | A | -2 |
| B | Go | C | -6 |
| C | Go | A | 2 |
| A | Go | A | -2 |

What are the estimates for the following Q-values as obtained by Q-learning? All Q-values are initialized to 0.

(i) $Q(C, Stop) =$ 0.5

(ii) $Q(C, Go) =$ 1.5

For this, we only need to consider the following three samples.

$$Q(A, Go) \leftarrow (1-\alpha)Q(A, Go) + \alpha(r + \gamma \max_a Q(B, a)) = 0.5(0) + 0.5(2) = 1$$
$$Q(C, Stop) \leftarrow (1-\alpha)Q(C, Stop) + \alpha(r + \gamma \max_a Q(A, a)) = 0.5(0) + 0.5(1) = 0.5$$
$$Q(C, Go) \leftarrow (1-\alpha)Q(C, Go) + \alpha(r + \gamma \max_a Q(A, a)) = 0.5(0) + 0.5(3) = 1.5$$

Instead of sampling, we now wish to use **variable elimination** to calculate $P(+a|+d)$. We start with the factorized representation of the joint probability:

$$P(A,B,C,+d) = P(A)P(B|A)P(C|A,B)P(+d|C)$$

(i) We begin by eliminating the variable $B$, which creates a new factor $f_1$. Complete the expression for the factor $f_1$ in terms of other factors.

$f_1($ $A, C$ $) =$ $\sum_b P(b|A)P(C|A,b)$

(ii) After eliminating $B$ to create a factor $f_1$, we next eliminate $C$ to create a factor $f_2$. What are the remaining factors after both $B$ and $C$ are eliminated?

■ $p(A)$ ☐ $p(B|A)$ ☐ $p(C|A,B)$ ☐ $p(+d|C)$ ☐ $f_1$ ■ $f_2$

(iii) After eliminating both $B$ and $C$, we are now ready to calculate $P(+a|+d)$. Write an expression for $P(+a|+d)$ in terms of the remaining factors.

$P(+a|+d) =$ $\frac{P(+a)f_2(+a,+d)}{\sum_a P(a)f_2(a,+d)}$

Suppose $T$ and $R$ are *unknown*. You will develop sample-based methods to estimate $Q^\pi$. You obtain a series of *samples* $(s_1, a_1, r_1), (s_2, a_2, r_2), \ldots (s_T, a_T, r_T)$ from acting according to this policy (where $a_t = \pi(s_t)$, for all $t$).

(i) Recall the update equation for the Temporal Difference algorithm, performed on each sample in sequence:

$$V(s_t) \leftarrow (1-\alpha)V(s_t) + \alpha(r_t + \gamma V(s_{t+1}))$$

which approximates the expected discounted reward $V^\pi(s)$ for following policy $\pi$ from each state $s$, for a learning rate $\alpha$.

Fill in the blank below to create a similar update equation which will approximate $Q^\pi$ using the samples.

You can use any of the terms $Q, s_t, s_{t+1}, a_t, a_{t+1}, r_t, r_{t+1}, \gamma, \alpha, \pi$ in your equation, as well as $\sum$ and max with any index variables (i.e. you could write $\max_a$, or $\sum_a$ and then use $a$ somewhere else), but no other terms.

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha\left[r_t + \gamma Q(s_{t+1}, a_{t+1})\right]$$

(ii) Now, we will approximate $Q^\pi$ using a linear function: $Q(s,a) = \mathbf{w}^T\mathbf{f}(s,a)$ for a weight vector $\mathbf{w}$ and feature function $\mathbf{f}(s,a)$.

To decouple this part from the previous part, use $Q_{samp}$ for the value in the blank in part (i) (i.e. $Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha Q_{samp}$).

Which of the following is the correct sample-based update for $\mathbf{w}$?

○ $\mathbf{w} \leftarrow \mathbf{w} + \alpha[Q(s_t,a_t) - Q_{samp}]$
○ $\mathbf{w} \leftarrow \mathbf{w} - \alpha[Q(s_t,a_t) - Q_{samp}]$
○ $\mathbf{w} \leftarrow \mathbf{w} + \alpha[Q(s_t,a_t) - Q_{samp}]\mathbf{f}(s_t,a_t)$
● $\mathbf{w} \leftarrow \mathbf{w} - \alpha[Q(s_t,a_t) - Q_{samp}]\mathbf{f}(s_t,a_t)$
○ $\mathbf{w} \leftarrow \mathbf{w} + \alpha[Q(s_t,a_t) - Q_{samp}]\mathbf{w}$
○ $\mathbf{w} \leftarrow \mathbf{w} - \alpha[Q(s_t,a_t) - Q_{samp}]\mathbf{w}$

(iii) The algorithms in the previous parts (part i and ii) are:

☐ model-based ■ model-free

- Sampling distribution if z sampled and e fixed evidence

$$S_{WS}(z,e) = \prod_{i=1}^{l} P(z_i|\text{Parents}(Z_i))$$

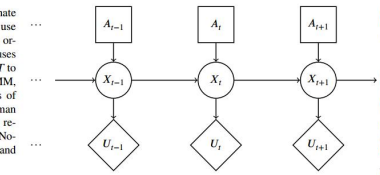
- Now, samples have weights

$$w(z,e) = \prod_{i=1}^{m} P(e_i|\text{Parents}(E_i))$$

- Together, weighted sampling distribution is consistent

$$S_{WS}(z,e) \cdot w(z,e) = \prod_{i=1}^{l} P(z_i|\text{Parents}(z_i)) \prod_{i=1}^{m} P(e_i|\text{Parents}(e_i))$$
$$= P(z,e)$$

Pacman is tired of using HMMs to estimate the location of ghosts. He wants to use HMMs to plan what actions to take in order to maximize his utility. Pacman uses the HMM (drawn to the right) of length $T$ to model the planning problem. In the HMM, $X_{1:T}$ is the sequence of hidden states of Pacman's world, $A_{1:T}$ are actions Pacman can take, and $U_t$ is the utility Pacman receives at the particular hidden state $X_t$. Notice that there are no evidence variables, and utilities are not discounted.




$$B_t(X_t) = p(X_t|a_{1:t})$$
$$= \sum_{x_{t-1}} p(X_t|x_{t-1}, a_t)p(x_{t-1}|a_{1:t-1})$$
$$= \sum_{x_{t-1}} p(X_t|x_{t-1}, a_t)B_{t-1}(x_{t-1})$$

$$\text{MEU}_{1:T} = \max_{a_{1:T}} \sum_{t=1}^{T} \sum_{x_t} B_t(x_t)U_t(x_t)$$

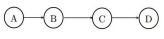
## Sampling

Suppose we want to evaluate $P(Q|E)$ where $Q$ are the query variables and $E$ are the evidence variables.

**Prior Sampling**: Draw samples from the Bayes net by sampling the parents and then sampling the children given the parents. $P(Q|E) \approx \frac{\text{count}(Q \text{and} E)}{\text{count}(E)}$.

**Rejection Sampling**: Like prior sampling, but ignore all samples that are inconsistent with the evidence.

**Likelihood Weighting**: Fix the evidence variables, and weight each sample by the probability of the evidence variables given their parents.




**Gibbs Sampling**:

1. Fix evidence.
2. Initialize other variables randomly
3. Repeat:
   (a) Choose non-evidence variable $X$.
   (b) Resample $X$ from $P(X|markovblanket(X))$

Using Likelihood Weighting Sampling to estimate $P(-a \mid +b, -d)$, the following samples were obtained. Fill in the weight of each sample in the corresponding row.

| Sample | | | | Weight |
|---|---|---|---|---|
| $-a$ | $+b$ | $+c$ | $-d$ | $P(+b \mid -a)P(-d \mid +c) = 1/3 * 5/6 = 5/18 = 0.277$ |
| $+a$ | $+b$ | $+c$ | $-d$ | $P(+b \mid +a)P(-d \mid +c) = 1/5 * 5/6 = 5/30 = 1/6 = 0.17$ |
| $+a$ | $+b$ | $-c$ | $-d$ | $P(+b \mid +a)P(-d \mid -c) = 1/5 * 1/8 = 1/40 = 0.025$ |
| $-a$ | $+b$ | $-c$ | $-d$ | $P(+b \mid -a)P(-d \mid -c) = 1/3 * 1/8 = 1/24 = 0.042$ |

Let's use Particle Filtering to estimate the distribution of $P(W_2|O_1 = a, O_2 = b)$. Here's the HMM again. $O_1$ and $O_2$ are supposed to be shaded.


| $W_1$ | $P(W_1)$ |
|---|---|
| 0 | 0.3 |
| 1 | 0.7 |

| $W_t$ | $W_{t+1}$ | $P(W_{t+1}\|W_t)$ |
|---|---|---|
| 0 | 0 | 0.4 |
| 0 | 1 | 0.6 |
| 1 | 0 | 0.8 |
| 1 | 1 | 0.2 |

| $W_t$ | $O_t$ | $P(O_t\|W_t)$ |
|---|---|---|
| 0 | a | 0.9 |
| 0 | b | 0.1 |
| 1 | a | 0.5 |
| 1 | b | 0.5 |

We start with two particles representing our distribution for $W_1$.
$P_1 : W_1 = 0$
$P_2 : W_1 = 1$
Use the following random numbers to run particle filtering:

$$[0.22,\ 0.05,\ 0.33,\ 0.20,\ 0.84,\ 0.54,\ 0.79,\ 0.66,\ 0.14,\ 0.96]$$

**(a) Observe**: Compute the weight of the two particles after evidence $O_1 = a$.

$w(P_1) = P(O_t = a|W_t = 0) = 0.9$
$w(P_2) = P(O_t = a|W_t = 1) = 0.5$

**(b) Resample**: Using the random numbers, resample $P_1$ and $P_2$ based on the weights.

We now sample from the weighted distribution we found above. Using the first two random samples, we find:
$P_1 = sample(weights, 0.22) = 0$
$P_2 = sample(weights, 0.05) = 0$

**(c) Predict**: Sample $P_1$ and $P_2$ from applying the time update.

$P_1 = sample(P(W_{t+1}|W_t = 0), 0.33) = 0$
$P_2 = sample(P(W_{t+1}|W_t = 0), 0.20) = 0$

**(d) Update**: Compute the weight of the two particles after evidence $O_2 = b$.

$w(P_1) = P(O_t = b|W_t = 0) = 0.1$
$w(P_2) = P(O_t = b|W_t = 0) = 0.1$

**(e) Resample**: Using the random numbers, resample $P_1$ and $P_2$ based on the weights.

Because both of our particles have $X = 0$, resampling will still leave us with two particles with $X = 0$.
$P_1 = 0$
$P_2 = 0$

**(f)** What is our estimated distribution for $P(W_2|O_1 = a, O_2 = b)$?

$P(W_2 = 0|O_1 = a, O_2 = b) = 2/2 = 1$
$P(W_2 = 1|O_1 = a, O_2 = b) = 0/2 = 0$

**拉普拉斯平滑 (Laplace Smoothing)** 是一种在统计学和自然语言处理 (NLP) 中常用的技术，主要用于解决在概率模型中遇到未见过事件时的问题，特别是在词汇稀疏的情况下。它的目标是通过调整频率估计，避免出现零概率的问题。

**1. 问题背景**

在许多统计模型（如朴素贝叶斯分类器）中，我们需要计算词汇或事件的条件概率。假设我们在处理一个词汇表时，会遇到以下问题：

- **未见过的词**：模型在训练时没有看到过某个词，但我们在测试时可能会遇到这个词。在没有平滑处理的情况下，模型可能会为这个词分配零概率，这会影响后续的计算，尤其是在做推理时。

**2. 拉普拉斯平滑的原理**

拉普拉斯平滑（也叫加1平滑）通过给每个词的计数加上一个常数（通常是 1），来解决零概率问题。具体来说，对于每个词汇 $w$，其概率估计为：

$$P(w) = \frac{\text{Count}(w) + 1}{\text{Total count of all words} + \text{Vocabulary size}}$$

- **Count(w)** 是词 $w$ 在训练集中的出现次数。
- **Vocabulary size** 是词汇表的大小，即所有不同词的数量。
- **Total count of all words** 是训练集中所有词汇出现的总次数。

What it the **minimum** number of parameters needed to fully model a joint distribution $P(Y, F_1, F_2, ..., F_n)$ over label $Y$ and $n$ features $F_i$? Assume binary class where each feature can possibly take on $k$ distinct values. $2k^n - 1$

Under the **Naive Bayes assumption**, what is the **minimum** number of parameters needed to model a joint distribution $P(Y, F_1, F_2, ..., F_n)$ over label $Y$ and $n$ features $F_i$? Assume binary class where each feature can take on $k$ distinct values. $2n(k-1)+1$

You suspect that you are overfitting with your Naive Bayes with Laplace Smoothing. How would you adjust the strength $k$ in Laplace Smoothing?

● Increase $k$ ○ Decrease $k$

While using Naive Bayes with Laplace Smoothing, increasing the strength $k$ in Laplace Smoothing can:

■ Increase training error ☐ Decrease training error
■ Increase validation error ■ Decrease validation error

In binary perceptron where the initial weight vector is $\vec{0}$, the final weight vector can be written as a linear combination the training data feature vectors.

● True ○ False

For binary class classification, logistic regression produces a linear decision boundary.

● True ○ False

In the binary classification case, logistic regression is exactly equivalent to a single-layer neural network with a sigmoid activation and the cross-entropy loss function.

● True ○ False

You train a linear classifier on 1,000 training points and discover that the training accuracy is only 50%. Which of the following, if done in isolation, has a good chance of improving your training accuracy?

■ Add novel features ☐ Train on more data

Consider the following Hidden Markov Model. $O_1$ and $O_2$ are supposed to be shaded.


| $W_1$ | $P(W_1)$ |
|---|---|
| 0 | 0.3 |
| 1 | 0.7 |

| $W_t$ | $W_{t+1}$ | $P(W_{t+1}\|W_t)$ |
|---|---|---|
| 0 | 0 | 0.4 |
| 0 | 1 | 0.6 |
| 1 | 0 | 0.8 |
| 1 | 1 | 0.2 |

| $W_t$ | $O_t$ | $P(O_t\|W_t)$ |
|---|---|---|
| 0 | a | 0.9 |
| 0 | b | 0.1 |
| 1 | a | 0.5 |
| 1 | b | 0.5 |

Suppose that we observe $O_1 = a$ and $O_2 = b$.
Using the forward algorithm, compute the probability distribution $P(W_2|O_1 = a, O_2 = b)$ one step at a time.

(a) Compute $P(W_1, O_1 = a)$.

$P(W_1, O_1 = a) = P(W_1)P(O_1 = a|W_1)$
$P(W_1 = 0, O_1 = a) = (0.3)(0.9) = 0.27$
$P(W_1 = 1, O_1 = a) = (0.7)(0.5) = 0.35$

(b) Using the previous calculation, compute $P(W_2, O_1 = a)$.

$P(W_2, O_1 = a) = \sum_{w_1} P(w_1, O_1 = a)P(W_2|w_1)$
$P(W_2 = 0, O_1 = a) = (0.27)(0.4) + (0.35)(0.8) = 0.388$
$P(W_2 = 1, O_1 = a) = (0.27)(0.6) + (0.35)(0.2) = 0.232$

(c) Using the previous calculation, compute $P(W_2, O_1 = a, O_2 = b)$.

$P(W_2, O_1 = a, O_2 = b) = P(W_2, O_1 = a)P(O_2 = b|W_2)$
$P(W_2 = 0, O_1 = a, O_2 = b) = (0.388)(0.1) = 0.0388$
$P(W_2 = 1, O_1 = a, O_2 = b) = (0.232)(0.5) = 0.116$

(d) Finally, compute $P(W_2|O_1 = a, O_2 = b)$.

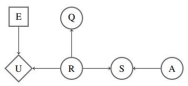
Renormalizing the distribution above, we have
$P(W_2 = 0|O_1 = a, O_2 = b) = 0.0388/(0.0388 + 0.116) \approx 0.25$
$P(W_2 = 1|O_1 = a, O_2 = b) = 0.116/(0.0388 + 0.116) \approx 0.75$

Pacman notices that calculating the beliefs under this model is very slow using exact inference. He therefore decides to try out various particle filter methods to speed up inference. Order the following methods by how accurate their estimate of $B_T(X_T)$ is? If different methods give an equivalently accurate estimate, mark them as the same number.

| | Most accurate | | | | Least accurate |
|---|---|---|---|---|---|
| Exact inference | ● 1 | ○ 2 | ○ 3 | ○ 4 | |
| Particle filtering with no resampling | ○ 1 | ● 2 | ○ 3 | ○ 4 | |
| Particle filtering with resampling before every time elapse | ○ 1 | ○ 2 | ○ 3 | ● 4 | |
| Particle filtering with resampling before every other time elapse | ○ 1 | ○ 2 | ● 3 | ○ 4 | |

Exact inference will always be more accurate than using a particle filter. When comparing the particle filter resampling approaches, notice that because there are no observations, each particle will have weight 1. Therefore resampling when particle weights are 1 could lead to particles being lost and hence prove bad.

Valerie is unsatisfied with the previous model and wants to incorporate more variables into her decision network. First, she realizes that the air quality (A) can affect her smelling accuracy. Second, she realizes that she can question (Q) the people around to see if they know where the cookie came from. These additions are reflected in the decision network below.




Choose one for each equation:

| | Could Be True | Must Be True | Must Be False |
|---|---|---|---|
| $VPI(A,S) > VPI(A) + VPI(S)$ | ● | ○ | ○ |
| $VPI(A) = 0$ | ○ | ● | ○ |
| $VPI(Q,R) \le VPI(Q) + VPI(R)$ | ○ | ● | ○ |
| $VPI(S,R) > VPI(R)$ | ○ | ○ | ● |
| $VPI(Q) \ge 0$ | ○ | ● | ○ |
| $VPI(Q,A) > VPI(Q)$ | ○ | ○ | ● |
| $VPI(S\|A) < VPI(S)$ | ○ | ○ | ● |
| $VPI(A\|S) > VPI(A)$ | ● | ○ | ○ |

We would like to classify some data. We have $N$ samples, where each sample consists of a feature vector $\mathbf{x} = [x_1, \cdots, x_k]^T$ and a label $y \in \{0, 1\}$.

Logistic regression produces predictions as follows:

$$P(Y = 1 \mid X) = h(\mathbf{x}) = s\left(\sum_i w_i x_i\right) = \frac{1}{1 + \exp(-(\sum_i w_i x_i))}$$

$$s(\gamma) = \frac{1}{1 + \exp(-\gamma)}$$

where $s(\gamma)$ is the logistic function, $\exp x = e^x$, and $\mathbf{w} = [w_1, \cdots, w_k]^T$ are the learned weights.

Let's find the weights $w_j$ for logistic regression using stochastic gradient descent. We would like to minimize the following loss function (called the cross-entropy loss) for each sample:

$$L = -[y \ln h(\mathbf{x}) + (1-y) \ln(1 - h(\mathbf{x}))]$$

(a) Show that $s'(\gamma) = s(\gamma)(1 - s(\gamma))$

$s(\gamma) = (1 + \exp(-\gamma))^{-1}$
$s'(\gamma) = -(1 + \exp(-\gamma))^{-2}(-\exp(-\gamma))$
$s'(\gamma) = \frac{1}{1+\exp(-\gamma)} \cdot \frac{\exp(-\gamma)}{1+\exp(-\gamma)}$
$s'(\gamma) = s(\gamma)(1 - s(\gamma))$

(b) Find $\frac{dL}{dw_j}$. Use the fact from the previous part.

Use chain rule:

$$\frac{dL}{dw_j} = -\left[\frac{y}{h(\mathbf{x})} s'(\sum_i w_i x_i) x_j - \frac{1-y}{1-h(\mathbf{x})} s'(\sum_i w_i x_i) x_j\right]$$

Use fact from previous part:

$$\frac{dL}{dw_j} = -\left[\frac{y}{h(\mathbf{x})} h(\mathbf{x})(1-h(\mathbf{x}))x_j - \frac{1-y}{1-h(\mathbf{x})} h(\mathbf{x})(1-h(\mathbf{x}))x_j\right]$$

Simplify:

$$\frac{dL}{dw_j} = -[y(1-h(\mathbf{x}))x_j - (1-y)h(\mathbf{x})x_j]$$
$$= -x_j[y - yh(\mathbf{x}) - h(\mathbf{x}) + yh(\mathbf{x})]$$
$$= -x_j(y - h(\mathbf{x}))$$

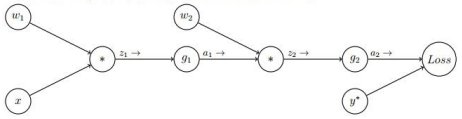
(c) Now, find a simple expression for $\nabla_{\mathbf{w}} L = [\frac{dL}{dw_1}, \frac{dL}{dw_2}, ..., \frac{dL}{dw_k}]^T$

$$\nabla_{\mathbf{w}} L = [-x_1(y - h(\mathbf{x})), -x_2(y - h(\mathbf{x})), ..., -x_k(y - h(\mathbf{x}))]^T$$
$$= -[x_1, x_2, ...x_k]^T(y - h(\mathbf{x}))$$
$$= -\mathbf{x}(y - h(\mathbf{x}))$$

(d) Write the stochastic gradient descent update for $\mathbf{w}$. Our step size is $\eta$.

$$\mathbf{w} \leftarrow \mathbf{w} + \eta\mathbf{x}(y - h(\mathbf{x}))$$

Consider the following computation graph for a simple neural network for binary classification. Here $x$ is a single real-valued input feature with an associated class $y^*$ (0 or 1). There are two weight parameters $w_1$ and $w_2$, and non-linearity functions $g_1$ and $g_2$ (to be defined later, below). The network will output a value $a_2$ between 0 and 1, representing the probability of being in class 1. We will be using a loss function $Loss$ (to be defined later, below), to compare the prediction $a_2$ with the true class $y^*$.




1. Perform the forward pass on this network, writing the output values for each node $z_1, a_1, z_2$ and $a_2$ in terms of the node's input values:

$z_1 = x * w_1$
$a_1 = g_1(z_1)$
$z_2 = a_1 * w_2$
$a_2 = g_2(z_2)$

2. Compute the loss $Loss(a_2, y^*)$ in terms of the input $x$, weights $w_i$, and activation functions $g_i$:
Recursively substituting the values computed above, we have:

$Loss(a_2, y^*) = Loss(g_2(w_2 * g_1(w_1 * x)), y^*)$

3. Now we will work through parts of the backward pass, incrementally. Use the chain rule to derive $\frac{\partial Loss}{\partial w_2}$. Write your expression as a product of partial derivatives at each node: i.e. the partial derivative of the node's output with respect to its inputs. (Hint: the series of expressions you wrote in part 1 will be helpful; you may use any of those variables.)

$$\frac{\partial Loss}{\partial w_2} = \frac{\partial Loss}{\partial a_2}\frac{\partial a_2}{\partial z_2}\frac{\partial z_2}{\partial w_2}$$

4. Suppose the loss function is quadratic, $Loss(a_2, y^*) = \frac{1}{2}(a_2 - y^*)^2$, and $g_1$ and $g_2$ are both sigmoid functions $g(z) = \frac{1}{1+e^{-z}}$ (note: it's typically better to use a different type of loss, *cross-entropy*, for classification problems, but we'll use this to make the math easier).

Using the chain rule from Part 3, and the fact that $\frac{\partial g(z)}{\partial z} = g(z)(1 - g(z))$ for the sigmoid function, write $\frac{\partial Loss}{\partial w_2}$ in terms of the values from the forward pass, $y^*$, $a_1$, and $a_2$:

First we'll compute the partial derivatives at each node:

$$\frac{\partial Loss}{\partial a_2} = (a_2 - y^*)$$
$$\frac{\partial a_2}{\partial z_2} = \frac{\partial g_2(z_2)}{\partial z_2} = g_2(z_2)(1 - g_2(z_2)) = a_2(1 - a_2)$$
$$\frac{\partial z_2}{\partial w_2} = a_1$$

Finally, write $\frac{\partial Loss}{\partial w_2}$ in terms of $x, y^*, w_i, a_i, z_i$: The partial derivatives at each ones we computed in Part 4) are:

$$\frac{\partial z_2}{\partial a_1} = w_2$$
$$\frac{\partial a_1}{\partial z_1} = \frac{\partial g_1(z_1)}{\partial z_1} = g_1(z_1)(1 - g_1(z_1)) = a_1(1 - a_1)$$
$$\frac{\partial z_1}{\partial a_1} = x$$

o the chain rule from part 3:

$$\frac{\partial Loss}{\partial w_2} = \frac{\partial Loss}{\partial a_2}\frac{\partial a_2}{\partial z_2}\frac{\partial z_2}{\partial w_2}$$
$$= (a_2 - y^*) * a_2(1 - a_2) * a_1$$

ule to derive $\frac{\partial Loss}{\partial w_1}$ as a product of partial derivatives at each node used in the chain

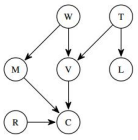
$$\frac{\partial Loss}{\partial w_1} = \frac{\partial Loss}{\partial a_2}\frac{\partial a_2}{\partial z_2}\frac{\partial z_2}{\partial a_1}\frac{\partial a_1}{\partial z_1}\frac{\partial z_1}{\partial w_1}$$

Plugging into the chain rule from Part 5 gives:

$$\frac{\partial Loss}{\partial w_1} = \frac{\partial Loss}{\partial a_2}\frac{\partial a_2}{\partial z_2}\frac{\partial z_2}{\partial a_1}\frac{\partial a_1}{\partial z_1}\frac{\partial z_1}{\partial w_1}$$
$$= (a_2 - y^*) * a_2(1 - a_2) * w_2 * a_1(1 - a_1) * x$$

What is the gradient descent update for $w_1$ with step-size $\alpha$ in terms of the values computed above?

$w_1 \leftarrow w_1 - \alpha(a_2 - y^*) * a_2(1 - a_2) * w_2 * a_1(1 - a_1) * x$




Pacman is trying to compute $P(C|+w,-t)$.

First, Pacman uses the standard variable elimination algorithm (as seen in lecture) to eliminate L and M.

**(a)** [2 pts] Select the factor(s) needed to eliminate L and M.

- ☐ $P(+w)$
- ■ $P(M|+w)$
- ☐ $P(C|R,+w,V)$
- ■ $P(C|R,M,V)$
- ☐ $P(-t)$
- ☐ $P(C|+w)$
- ■ $P(L|-t)$
- ☐ $P(R)$

These are all the factors that mention L or M.

**(b)** [2 pts] Select the factor(s) that remain after eliminating L and M.

Initial list of factors:

$P(+w)$
$P(-t)$
$P(M|+w)$
$P(V|+w,+t)$
$P(L|-t)$
$P(R)$
$P(C|R,M,V)$

When we join on $L$, we combine all factors involving $L$. There's only one factor involving $L$:

$P(L|-t)$

When we sum out $L$, this factor sums to 1, so we can safely drop it the list of factors.

$\sum_l P(l|-t) = 1$

When we join on $M$, we combine all factors involving $M$:

$P(M|+w) \cdot P(C|R,M,V) = f(M,+w,C,R,V)$

When we sum out $M$, we get:

$\sum_m f(m,+w,C,R,V) = f(+w,C,R,V)$

The remaining list of factors is:

$P(+w)$
$P(-t)$
$P(V|+w,-t)$
$P(R)$
$f(+w,C,R,V)$

Note: An earlier version of the solutions did not mark these three answer choices: $f(-t)$, $f(V,+w)$. The question was not clear on whether we wanted you to select the factors generated from elimin entire list of factors after eliminating L and M. If you use the former interpretation, then those three answer choices would not be marked. If you use the latter interpretation, then those three answer choices would be marked.

During grading, we gave everybody points for these three choices, regardless of whether you selected them or did not select them. (The other three answer choices are unaffected by the alternate interpretation.)

[2 pts] Which statement best explains why variable elimination (VE) is better than inference by enumeration (IBE)?

- ○ VE can use larger factors, so we can read off more values and use less computation.
- ● VE can use smaller factors, so we can use less computation.
- ○ While VE is usually better, sometimes IBE creates smaller factors.
- ○ VE is always strictly better, because it always uses strictly smaller factors, using less computation.

[1 pt]

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sample 1: | +r | +w | +t | −v | +m | +l | +c |
| Sample 2: | −r | −w | −t | +v | −m | +l | +c |

■ Prior Sampling ☐ Likelihood Weighting ○ None of the above
☐ Rejection Sampling ☐ Gibbs Sampling

[1 pt]

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sample 1: | −r | +w | −t | −v | +m | +l | −c |
| Sample 2: | −r | +w | −t | +v | +m | +l | −c |

■ Prior Sampling ■ Likelihood Weighting ○ None of the above
■ Rejection Sampling ■ Gibbs Sampling

[1 pt]

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sample 1: | −r | +w | +t | rejected | rejected | rejected | rejected |
| Sample 2: | −r | +w | −t | rejected | rejected | rejected | rejected |

☐ Prior Sampling ☐ Likelihood Weighting ● None of the above
☐ Rejection Sampling ☐ Gibbs Sampling

[1 pt]

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sample 1: | +r | +w | −t | +v | −m | +l | −c |
| Sample 2: | −r | +w | −t | −v | −m | +l | −c |

■ Prior Sampling ■ Likelihood Weighting ○ None of the above
■ Rejection Sampling ☐ Gibbs Sampling

[4 pts] Consider a standard HMM (from lecture) with states $X_1, \ldots, X_N$ and evidence $E_1, \ldots, E_N$.

Suppose we're running particle filtering with a large number of particles. At time step $t$, we have a particle at state $X_t = x_t$, with weight 0.7. Select all true statements about this particle.

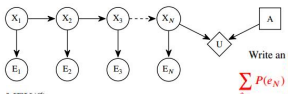
- ☐ We've just finished a time elapse update, and have not performed the observation update for time $t$ yet.
- ☐ 0.7 is the probability of the particle visiting all the states it's visited so far: $P(x_1, \ldots, x_t)$.
- ■ If this particle is drawn during resampling, the resulting new particle will still be at state $x_t$.
- ☐ 0.3 is the probability that this particle disappears during resampling.
- ○ None of the above

Option 1: False. The particles take on weights after the observation update, not after the time elapse update.

Option 2: False. 0.7 is the probability of the evidence, given the particle's current state. $P(e_t|x_t)$ is not equal to $P(x_1, \ldots, x_t)$.

Option 3: True. The resampling process does not change the particles' states.

Option 4: False. The weight of a particle is not the probability that it is resampled, because the weights of the particles are not normalized (don't sum to 1), and because we are drawing many particles from the weighted particle distribution, with replacement (so there are many chances for the particle to be resampled).




Write an expression that can be used to compute MEU($E_N$).

$$\sum_{e_N} P(e_N) \left[ \max_a \sum_{x_N} P(x_N|e_N) U(x_N, a) \right]$$

MEU(∅)

- ● Run the forward algorithm, skipping all observation updates.
- ○ Read the conditional probability table under $X_N$.

We need $P(X_N)$. This is not in the Bayes' net; the CPT under $X_N$ is $P(X_N|X_{N-1})$.

To obtain $P(X_N)$, we need to run the forward algorithm, skipping all observation updates, because we have no evidence.

- ● Apply an additional observation update.
- ○ Apply an additional time elapse update.
- ○ Apply an additional time elapse and observation update.
- ○ Re-run the forward algorithm from the beginning, with no modifications.

From the previous subpart, we have $P(X_N)$, but we need $P(X_N|E_N)$. This requires one extra evidence update in the forward algorithm: we need to weight every value in the table $P(X_N)$ by $P(E_N|X_N)$, and then normalize.

In equations: We have $P(X_N)$ (from the earlier subparts) and $P(E_N|X_N)$ (from the Bayes' net). We want $P(X_N|E_N)$, so we can apply Bayes' rule:

$$P(X_N|E_N) = \frac{P(X_N)P(E_N|X_N)}{P(E_N)}$$

In other words, in the numerator, we're multiplying every value in the $P(X_N)$ table by $P(E_N|X_N)$, and then we normalize (since the denominator is a constant).

[2 pts] Which of the following computations can be used to derive the distribution in blank (i)?

- ○ $P(x_N)P(E_N|x_N)$
- ○ $\sum_{x_1} P(x_N)P(e_N|x_N)$
- ● $\sum_{x_N} P(x_N)P(E_N|x_N)$
- ○ $\sum_{x_N} P(E_N)P(x_N|E_N)$

We're given $P(X_N)$ (from earlier subparts), and $P(E_N|X_N)$ (from the Bayes' net). Using the chain rule, we can get:

$$P(X_N)P(E_N|X_N) = P(X_N, E_N)$$

Then, you can sum out $X_N$ to get:

$$\sum_{x_N} P(x_N)P(E_N|x_N) = P(E_N)$$

For the rest of the question, suppose we would like to compute $\text{VPI}(E_1, \ldots, E_N) = \text{MEU}(E_1, \ldots, E_N) - \text{MEU}(\emptyset)$.

**(f)** [2 pts] To compute MEU($E_1, \ldots, E_N$), we'll need one or more distribution(s) over $X_N$.
Which of these computations will generate the necessary distribution(s) over $X_N$?

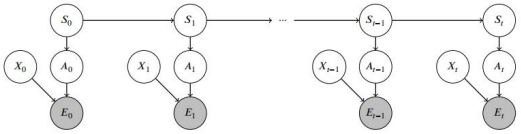
- ○ Run the forward algorithm once, with no modifications.
- ● Run the forward algorithm $2^N$ times, with no modifications.
- ○ Run the forward algorithm once, skipping all observation updates.
- ○ Run the forward algorithm $2^N$ times, skipping all observation updates.

We don't know what the evidence is, so we have to run the forward algorithm once for every possible setting of the evidence $P(X_N|e_1, \ldots, e_N)$.

**(g)** [1 pt] To compute MEU($E_1, \ldots, E_N$), which other distribution do we need?

- ○ $P(E_1)$
- ○ $P(E_N)$
- ● $P(E_1, \ldots, E_N)$
- ○ $P(E_1, \ldots, E_N|X_N)$

We need to weight each MEU($e_1, \ldots, e_N$) by the corresponding probability of evidence, $P(E_1, \ldots, E_N)$.




**(a)** Consider the above dynamic bayes net which ends at some finite timestep $t$. In this problem, we are trying to approximate the most likely value of $S_t$ given all the evidence variables up to and including $t$. For each of the following subparts, first decide whether the given method can be used to solve this problem. Then, if yes, select all CPTs which must be known to run the algorithm.

**(i)** [1 pt] Variable elimination
○ No ● Yes: ■ $P(S_0)$, ■ $P(S_t|S_{t-1}), t > 0$ ■ $P(E_t|X_t, A_t) \forall t$ ■ $P(A_t|S_t) \forall t$ ■ $P(X_t) \forall t$

**(ii)** [1 pt] Value iteration
● No ○ Yes: ☐ $P(S_0)$, ☐ $P(S_t|S_{t-1}), t > 0$ ☐ $P(E_t|X_t, A_t) \forall t$ ☐ $P(A_t|S_t) \forall t$ ☐ $P(X_t) \forall t$

**(iii)** [1 pt] Gibbs sampling
○ No ● Yes: ■ $P(S_0)$, ■ $P(S_t|S_{t-1}), t > 0$ ■ $P(E_t|X_t, A_t) \forall t$ ■ $P(A_t|S_t) \forall t$ ■ $P(X_t) \forall t$

**(iv)** [1 pt] Prior sampling
○ No ● Yes: ■ $P(S_0)$, ■ $P(S_t|S_{t-1}), t > 0$ ■ $P(E_t|X_t, A_t) \forall t$ ■ $P(A_t|S_t) \forall t$ ■ $P(X_t) \forall t$

**(v)** [1 pt] Particle Filtering
○ No ● Yes: ■ $P(S_0)$, ■ $P(S_t|S_{t-1}), t > 0$ ■ $P(E_t|X_t, A_t) \forall t$ ■ $P(A_t|S_t) \forall t$ ■ $P(X_t) \forall t$

) For the HMM shown above, determine the correct recursive formula for the belief distribution update from $B(S_{t-1})$ to $B(S_t)$. Recall that the belief distribution $B(S_t)$ represents the probability $P(S_t|E_{0:t})$ and involves two steps: **(i)** Time elapse and **(ii)** Observation update.

$$B(S_t) \propto \underline{\quad\text{(ii)}\quad} \cdot \underline{\quad\text{(i)}\quad}$$

**(i)** [1 pt] Time elapse

- ● $\sum_{S_{t-1}} P(S_t|S_{t-1})B(S_{t-1})$
- ○ $\sum_{S_{t-1}} \sum_{A_{t-1}} P(S_t|S_{t-1})P(A_{t-1}|S_{t-1})B(S_{t-1})$
- ○ $\sum_{S_{t-1}} P(S_t|S_{t-1})P(A_t|S_t)B(S_{t-1})$
- ○ $\sum_{S_{t-1}} \sum_{A_{t-1}} P(S_t|S_{t-1})P(A_{t-1}|S_{t-1})P(A_t|S_t)B(S_{t-1})$

This is the standard time elapse update (see 8.2 of the textbook). Note that the extra variables do not affect that transition from $S_t$ to its next state $S_{t+1}$ so the time elapse expression is the same.

**(ii)** [1 pt] Observation update

- ○ $P(E_t|X_t, A_t)$
- ○ $P(E_t|X_t, A_t)P(X_t)P(A_t|S_t)$
- ○ $\sum_{x \in X_t} \sum_{a \in A_t} P(E_t|x,a)$
- ● $\sum_{x \in X_t} \sum_{a \in A_t} P(E_t|x,a)P(x)P(a|S_t)$
- ○ $\prod_{x \in X_t} \prod_{a \in A_t} P(E_t|x,a)$
- ○ $\prod_{x \in X_t} \prod_{a \in A_t} P(E_t|x,a)P(x)P(a|S_t)$

The observation update is the probability of evidence ($E_t$) given the state ($S_t$). Since there are extra hidden variables $X_t$ and $A_t$ at each timestep, these need to be summed out and eliminated to get the correct expression for $P(E_t|S_t)$ at each timestep.