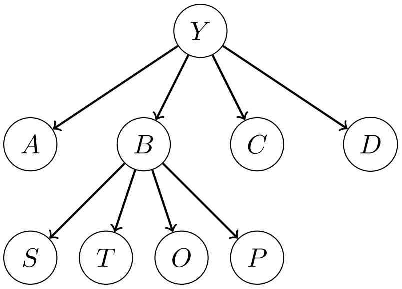
CS 188 Summer 2022 Final Review Machine Learning Solutions

Q1. A Nonconvolutional Nontrivial Network

You have a robotic friend MesutBot who has trouble passing Recaptchas (and Turing tests in general). MesutBot got a 99.99% on the last midterm because he could not determine which squares in the image contained stop signs. To help him ace the final, you decide to design a few classifiers using the below features.

- $A = 1$ if the image contains an octagon, else 0.
- $B = 1$ if the image contains the word STOP, else 0.
  - $S = 1$ if the image contains the letter S, else 0.
  - $T = 1$ if the image contains the letter T, else 0.
  - $O = 1$ if the image contains the letter O, else 0.
  - $P = 1$ if the image contains the letter P, else 0.
- $C = 1$ if the image is more than 50% red in color, else 0.
- $D = 1$ if the image contains a post, else 0.



(a) First, we use a Naive Bayes-inspired approach to determine which images have stop signs based on the features and Bayes Net above. We use the following features to predict $Y = 1$ if the image has a stop sign anywhere, or $Y = 0$ if it doesn't.

(i) Which expressions would a Naive Bayes model use to predict the label for B if given the values for features $S = s, T = t, O = o, P = p$? Choose all valid expressions.

■ $b = \arg\max_b P(b)P(s|b)P(t|b)P(o|b)P(p|b)$

□ $b = \arg\max_b P(s|b)P(t|b)P(o|b)P(p|b)$

■ $b = \arg\max_b P(b|s,t,o,p)$

■ $b = \arg\max_b P(b,s,t,o,p)$

□ $b = \arg\max_b P(s,t,o,p|b)$

○ None

Note $\arg\max_b P(b)P(s|b)P(t|b)P(o|b)P(p|b) = \arg\max_b P(b,s,t,o,p)$, which are both correct. The conditional probability assumptions from the Bayes Net enable us to write this equality.

Note $P(s|b)P(t|b)P(o|b)P(p|b) = P(s,t,o,p|b)$. This can be read off of the Bayes Net as well, because all the features are independent given the label $B = b$.

Finally note $\arg\max_b P(b|s,t,o,p) = \arg\max_b \frac{P(b,s,t,o,p)}{P(s,t,o,p)} = \arg\max_b P(b,s,t,o,p)$ because $P(s,t,o,p)$ has all four of its values already given, and does not depend on our optimization variable b in any way.

(ii) **[Optional]** Which expressions would we use to predict the label for Y with our Bayes Net above? Assume we are given all features except B. So $A = a, S = s, T = t$, etc. For the below choices, the underscore means we are dropping the value of that variable. So $y, \_ = (0, 1)$ would mean $y = 0$.

□ $y, \_\_ = \arg\max_{y,b} P(y)P(a|y)P(b|y)P(c|y)P(d|y)P(s|b)P(t|b)P(o|b)P(p|b)$

□ $y, \_\_ = \arg\max_{y,b} P(s)P(t)P(o)P(p)P(a)P(b|s,t,o,p)P(c)P(d)P(y|a,b,c,d)$

□ First compute $b' = \arg\max_b$ of the formula chosen in part (ii).
Then compute $y = \arg\max_y P(y)P(a|y)P(b'|y)P(c|y)P(d|y)$

□ First compute $b' = \arg\max_b$ of the formula chosen in part (ii).
Then compute $y = \arg\max_y P(y|a,b',c,d)$

■ $y = \arg\max_y \sum_{b'} P(y)P(a|y)P(b'|y)P(c|y)P(d|y)P(s|b')P(t|b')P(o|b')P(p|b')$

○ None

Sum out possibilities for b given the features S, T, O, P

(iii) **[Optional]** One day MesutBot got allergic from eating too many cashews. The incident broke his letter S detector, so that he no longer gets reliable S features. Now what expressions would we use to predict the label for Y? Assume all features except B, S are given. So $A = a, T = t, O = o$, etc.

□ $y = \arg\max_y P(y)P(a|y)P(c|y)P(d|y)$

□ $y, \_\_, \_\_ = \arg\max_{y,b,s} P(y)P(a|y)P(b|y)P(c|y)P(d|y)P(s|b)P(t|b)P(o|b)P(p|b)$

□ $y, \_\_ = \arg\max_{y,s} P(y)P(a|y)P(b|y)P(c|y)P(d|y)P(s|b)P(t|b)P(o|b)P(p|b)$

□ $y, \_\_ = \arg\max_{y,b} P(y)P(a|y)P(b|y)P(c|y)P(d|y)P(t|b)P(o|b)P(p|b)$

□ $y, \_\_ = \arg\max_{y,b} P(y)P(a|y)P(b|y)P(c|y)P(d|y)P(s|b)P(t|b)P(o|b)P(p|b)$

□ $y, \_\_ = \arg\max_{y,b} P(y|a,b,c,d)$

■ $y = \arg\max_y P(y)P(a|y)P(c|y)P(d|y) \sum_{b',s'} P(b'|y)P(s'|b')P(t|b')P(o|b')P(p|b')$

○ None

Use variable elimination on s and b (because b cannot be accurately calculated without s).

(b) You decide to try to output a probability $P(Y|features)$ of a stop sign being in the picture instead of a discrete ±1 prediction. We denote this probability as $P(Y|\vec{f}(x))$. Which of the following functions return a **valid** probability distribution for $P(Y = y|\vec{f}(x))$? Recall that $y \in \{-1, 1\}$.

■ $\frac{e^{y\cdot\vec{w}^T\vec{f}(x)}}{e^{-y\cdot\vec{w}^T\vec{f}(x)}+e^{y\cdot\vec{w}^T\vec{f}(x)}}$

■ $\frac{1}{2}$

□ $\frac{0.5}{1+e^{-\vec{w}^T\vec{f}(x)}}$

□ $\frac{-1}{1+e^{\vec{w}^T\vec{f}(x)}}+1$

○ None

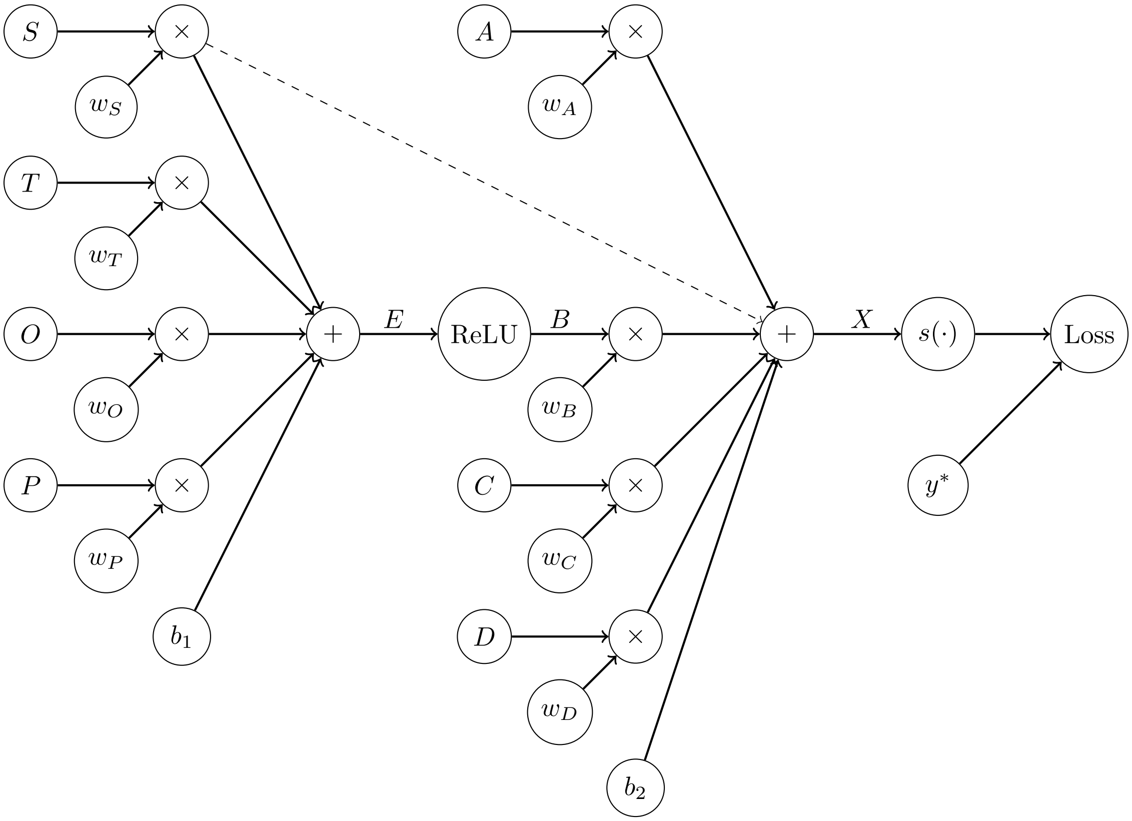
Valid probability distribution means that the probabilities over all possible values of y must sum to 1.

$P(Y = y|\vec{f}(x)) = \frac{e^{y\cdot\vec{w}^T\vec{f}(x)}}{e^{-y\cdot\vec{w}^T\vec{f}(x)}+e^{y\cdot\vec{w}^T\vec{f}(x)}}$ works because $P(Y = 1|\vec{f}(x)) + P(Y = -1|\vec{f}(x)) = 1$ (it is the softmax function).

$\frac{1}{2}$ works because we just need $P(Y = 1|\vec{f}(x)) + P(Y = -1|\vec{f}(x)) = \frac{1}{2} + \frac{1}{2} = 1$, so it is valid.

$\frac{0.5}{1+e^{-\vec{w}^T\vec{f}(x)}}$ and $\frac{-1}{1+e^{\vec{w}^T\vec{f}(x)}} + 1$ don't depend on y so we can't guarantee the sum of the two probabilities adds to 1, and thus cannot guarantee that those two expressions are a valid probability distribution.

You note that features are inputs into a neural network and the output is a label, so you modify the Bayes Net from above into a Neural Network computation graph. Recall the logistic function $s(x) = \frac{1}{1+e^{-x}}$ has derivative $\frac{\partial s(x)}{\partial x} = s(x)[1 - s(x)]$



(c) For this part, ignore the dashed edge when calculating the below.

(i) What is $\frac{\partial Loss}{\partial w_A}$?

● $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot A$

○ $2(s(X) - y^*) \cdot [s(X) \cdot (1 - s(X))] \cdot A$

○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot 2A + 1$

○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot 2A$

○ $2(s(X) - y^*) \cdot [s(X) \cdot (1 - s(X))] \cdot A + 1$

○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot A + 1$

○ None

$$\begin{aligned} \frac{\partial Loss}{\partial w_A} &= \frac{\partial Loss}{\partial s(X)} \cdot \frac{\partial s(X)}{\partial X} \cdot \frac{\partial X}{\partial A w_A} \cdot \frac{\partial A w_A}{\partial w_A} \\ &= \frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot 1 \cdot A \end{aligned}$$

(ii) What is $\frac{\partial Loss}{\partial w_S}$? Keep in mind we are still ignoring the dotted edge in this subpart.

● $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases}\right) \cdot S$

○ $2(s(X) - y^*) \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases}\right) \cdot S$

○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases}\right) \cdot 2S + S$

○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases}\right) \cdot 2S$

○ $2(s(X) - y^*) \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases}\right) \cdot S + S$

○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases}\right) \cdot S + S$

○ None

$$\frac{\partial Loss}{\partial w_S} = \frac{\partial Loss}{\partial s(X)} \cdot \frac{\partial s(X)}{\partial X} \cdot \frac{\partial X}{\partial B w_B} \cdot \frac{\partial B w_B}{\partial ReLU(E)} \cdot \frac{\partial ReLU(E)}{\partial E} \cdot \frac{\partial E}{\partial S w_S} \cdot \frac{\partial S w_S}{\partial w_S}$$
$$= \frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot 1 \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases}\right) \cdot 1 \cdot S$$

(d) **[Optional]** MesutBot is having trouble paying attention to the S feature because sometimes it gets zeroed out by the ReLU, so we connect it directly to the input of $s(\cdot)$ via the dotted edge. For the below, treat the dotted edge as a regular edge in the neural net.

(i) Which of the following is equivalent to $\frac{\partial Loss}{\partial w_A}$?

● $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot A$

○ $2(s(X) - y^*) \cdot [s(X) \cdot (1 - s(X))] \cdot A$

○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot 2A + A$

○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot 2A$

○ $2(s(X) - y^*) \cdot [s(X) \cdot (1 - s(X))] \cdot A + A$

○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot A + A$

○ None

This doesn't change because the added edge is further upstream from w_A and doesn't affect gradient flows between w_A and $Loss$. From above, we copy:

$$\frac{\partial Loss}{\partial w_A} = \frac{\partial Loss}{\partial s(X)} \cdot \frac{\partial s(X)}{\partial X} \cdot \frac{\partial X}{\partial A w_A} \cdot \frac{\partial A w_A}{\partial A}$$
$$= \frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot 1 \cdot w_A$$

(ii) Which of the following is equivalent to $\frac{\partial Loss}{\partial w_S}$? Keep in mind we are still treating the dotted edge as a regular edge.

○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot S$

○ $2(s(X) - y^*) \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot S$

○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot 2S + S$

○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot 2S$

○ $2(s(X) - y^*) \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot S + S$

○ $\frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot S + S$

● None

Due to the new dotted edge, there are now two paths along the neural network that lead from output to w_S.

$$\begin{aligned}
\frac{\partial Loss}{\partial w_S} &= \frac{\partial Loss}{\partial s(X)} \cdot \frac{\partial s(X)}{\partial X} \cdot \left(\frac{\partial X}{\partial B w_B} \cdot \frac{\partial B w_B}{\partial ReLU(E)} \cdot \frac{\partial ReLU(E)}{\partial E} \cdot \frac{\partial E}{\partial S w_S} + \frac{\partial X}{\partial S w_S} \right) \cdot \frac{\partial S w_S}{\partial w_S} \\
&= \frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot \left(1 \cdot w_B \cdot \left(\begin{cases} 1 & E \geq 0 \\ 0 & E < 0 \end{cases} \right) \cdot 1 + 1 \right) \cdot S \\
&= \frac{\partial Loss}{\partial s(X)} \cdot [s(X) \cdot (1 - s(X))] \cdot \left(\begin{cases} w_B + 1 & E \geq 0 \\ 1 & E < 0 \end{cases} \right) \cdot S
\end{aligned}$$

Q2. Kernel Perceptron

Remember that the perceptron update rule looks like:

$$w \leftarrow w + yx$$

for input feature vectors x and class labels $y \in \{-1, 1\}$. If $wx = 0$, we predict positive label.

(a) Suppose $w = [1, 1]$ initially, and we observe the following training examples:

x_0	x_1	y
1	2	-1
3	1	1
1	1	-1
1	0	1

What is the final value of w? [1, -3]

(b) Notice that because of the update rule, the final weight vector w^* is just a linear combination of training examples and the initializer. Suppose we iterate over the training set following the order in the table until all the training samples are classified correctly, fill in the coefficients below:

$$w = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \underline{-2} \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \underline{1} \cdot \begin{bmatrix} 3 \\ 1 \end{bmatrix} + \underline{-1} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \underline{0} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

This means that instead of *explicitly* representing w as a vector of feature weights, we can *implicitly* represent the decision rule with a vector v with one weight per example.

(c) Now suppose x and w are D-dimensional, and we have N training examples. How many numbers do I need to represent w *explicitly*?

D because w is D-dimensional.

(d) How many numbers do I need to represent w *implicitly*? (Assume that the initial value for w is public information so you do not need to consume any memory to store it.)

N because we need one coefficient for each training sample.

(e) Write the update rule for the implicit representation if you pick the i^{th} training sample (use e_i to represent a vector whose i^{th} position is 1 and all the other positions are 0s):

$$v \leftarrow v + ye_i$$

(f) Write the prediction rule for the implicit representation if the initial w is a zero vector. You can use v_i to represent the i^{th} value from v and x_i to represent the i^{th} training sample:

$$\text{pred}(x) = \sum_i v_i \langle x_i, x \rangle$$

(g) When is it more space-efficient to use the implicit representation? (Your answer should be at most three words/symbols, and be expressed in terms of D and N.)

$N < D$

(h) Now suppose x is two-dimensional, and we introduce a feature transformation

$$f(x) = [x_0^2, x_1^2, \sqrt{2}x_0, \sqrt{2}x_1, \sqrt{2}x_0x_1, 1]$$

It is not too hard to show that for any vectors a and b,

$$f(a) \cdot f(b) = (a \cdot b + 1)^2$$

How can we take advantage of this fact when working with the implicit representation? (Your answer should be 1 sentence.)

We can replace the old features with the transformed features to classify non-linearly separable training set.