CISC 484/684 — Intro to Machine Learning: Final Practice

True/False

For each question, check either True or False. If False, explain why.
6 points for True, 3 points for False, 3 points for an explanation for False.

1) The Kullback-Leibler divergence can be used as a "similarity" measure between two distributions. A 0 KL value indicates that the two distributions are identical.

☐ True

☐ False

Explanation if False:

True.

2) A perceptron can only classify data that is linearly separable.

☐ True

☐ False

Explanation if False:

True. A single-layer perceptron uses a linear decision boundary, so it can only solve linearly separable problems. Non-linear problems require additional layers or non-linear features.

3) Adding more layers to a neural network always reduces training error.

☐ True

☐ False

Explanation if False:

False. While deeper networks have greater capacity, they may suffer from vanishing/exploding gradients, poor initialization, or overfitting, so training error may not always decrease without proper techniques.

4) In a CNN, the convolution operation reduces the spatial dimensions of the input.

☐ True

☐ False

Explanation if False:

False. Convolution itself does not necessarily reduce spatial dimensions — reduction occurs only if stride > 1 or padding is less than needed to maintain size.

5) Dropout works by randomly zeroing out some neurons only during inference to prevent overfitting.

☐ True

☐ False

Explanation if False:

False. Dropout is applied only during training; at inference, all neurons are used but activations are scaled to match expected values.

6) In GAN training, the generator's objective is to minimize the discriminator's ability to correctly distinguish real from fake samples.

☐ True

☐ False

Explanation if False:

True. The generator tries to produce outputs that fool the discriminator, making its classification accuracy close to random guessing.

7) In a standard VAE, the encoder outputs both a mean vector and a variance vector for the latent distribution.

☐ True

☐ False

Explanation if False:

True. The encoder parameterizes a Gaussian latent distribution via its mean and (log-)variance, enabling stochastic sampling via the reparameterization trick.

8) Autoencoders are generative models because they learn a latent representation of the input.

☐ True

☐ False

Explanation if False:

False. Standard autoencoders are not generative in the probabilistic sense; they reconstruct inputs but do not define a generative distribution over new samples. VAEs, however, are generative.

9) A perceptron uses the sigmoid activation function to compute its output.

☐ True

☐ False

Explanation if False:

False. The original perceptron uses a hard threshold activation. Logistic regression uses the sigmoid; modern neural networks may use other differentiable activations.

10) In a CNN, increasing the number of filters in each layer always improves test accuracy.

☐ True

☐ False

Explanation if False:

False. While more filters can capture more features, excessive parameters can lead to overfitting, higher computation cost, and possible performance degradation.

Short Answer

11) **K-means Clustering**

Suppose you have the following 1-D data points: $x_1 = -4, x_2 = -3, x_3 = -2, x_4 = 2$, $x_5 = 3$. You wish to cluster these points using k-means, with $k = 2$ clusters, and you start from initial cluster centers $c_1 = 2$ and $c_2 = 3$.

1. Proceed with an assignment step: for each of the 5 points, give its assignment as either cluster 1 or cluster 2.

   x_1 through x_4 are assigned to cluster 1 , and x_5 is assigned to cluster 2.

2. At this point, what is the value of the k-means objective function? You do not need to worry about performing arithmetic here. In other words, an answer with a form resembling $(7.3 + 2.1)^3 + (8.3 + 1.2)^4$ is fine.

   $(-4 - 2)^2 + (-3 - 2)^2 + (-2 - 2)^2 + (2 - 2)^2 + (3 - 3)^2$

3. Proceed with an update step: compute the new cluster centers c_1 and c_2.

   $c_1 = -1.75, c_2 = 3$

4. Proceed again with an assignment step: for each point, give its new assignment as either cluster 1 or cluster 2.

   x_1 through x_3 are assigned to cluster 1 , and x_4 and x_5 are assigned to cluster 2 .

5. At this point, what is the value of the k-means objective function? You do not need to worry about performing arithmetic here. In other words, an answer with a form resembling $(7.3 + 2.1)^3 + (8.3 + 1.2)^4$ is fine.

   $(-4 - -1.75)^2 + (-3 - -1.75)^2 + (-2 - -1.75)^2 + (2 - 3)^2 + (3 - 3)^2$

12) **Neural Networks**

Consider the neural network architecture shown above for a binary classification problem. The values for the weights are shown in the figure. We define:

$$
\begin{aligned}
a_1 &= w_{11}x_1 + b_{11} \\
a_2 &= w_{12}x_1 + b_{12} \\
a_3 &= w_{21}z_1 + w_{22}z_2 + b_{21} \\
z_1 &= \text{ReLU}\left(a_1\right) \\
z_2 &= \text{ReLU}\left(a_2\right) \\
z_3 &= \sigma\left(a_3\right), \sigma(x) = \frac{1}{1+e^{-x}} \\
&\text{where } \text{ReLU}(x) = \max(0, x).
\end{aligned}
$$

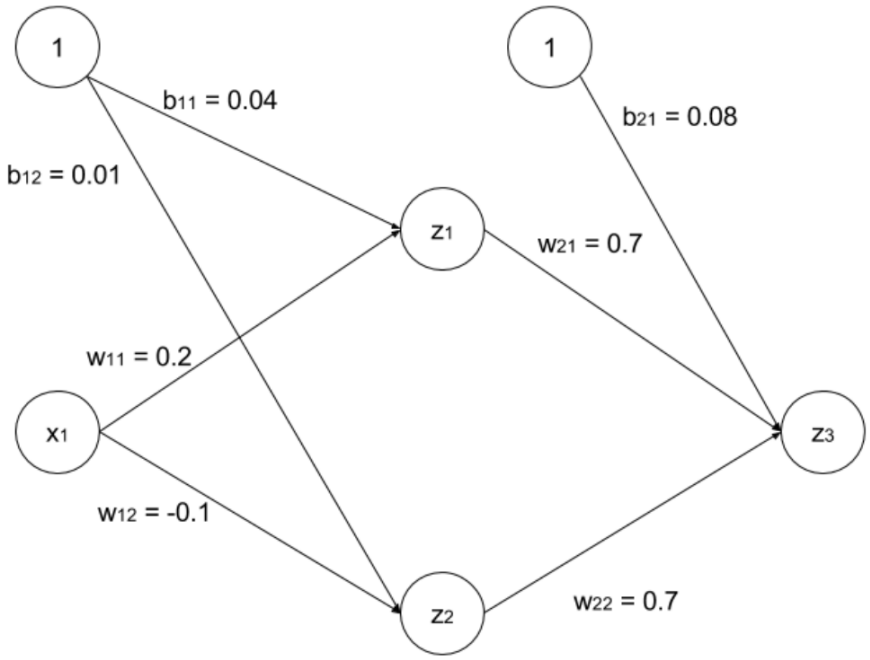

Figure 1: Neural Networks

1. For $x_1 = 0.3$, compute z_3 in terms of e.

   $z_3 = \frac{1}{1+e^{-0.15}}$

2. **Select from 0 or 1**: Which class does the network predict for the data point $x_1 = 0.3$, assuming that $\widehat{y} = 1$ if $z_3 > \frac{1}{2}$ and otherwise, $\widehat{y} = 0$.

   $\widehat{y}\left(x_1 = 0.3\right) = 1$

3. Perform backpropagation on the bias term b_{21} by deriving the expression for the gradient of the loss function $L\left(y, z_3\right)$ with respect to the bias term b_{21}, $\frac{\partial L}{\partial b_{21}}$.

Express your answer in terms of partial derivatives of the form $\frac{\partial \alpha}{\partial \beta}$, where α and β can be any of $L, z_i, a_i, b_{ij}, w_{ij}, x_1$ for all valid values of i, j. Your backpropagation algorithm should be as explicit as possible - that is, make sure each partial derivative $\frac{\partial \alpha}{\partial \beta}$ cannot be decomposed further into simpler partial derivatives. Do not evaluate the partial derivatives.

$\frac{\partial L}{\partial b_{21}} = \frac{\partial L}{\partial z_3} \frac{\partial z_3}{\partial a_3} \frac{\partial a_3}{\partial b_{21}}$

4. Perform backpropagation on the bias term b_{12} by deriving the expression for the gradient of the loss function $L(y, z_3)$ with respect to the bias term b_{12}, $\frac{\partial L}{\partial b_{12}}$. Express your answer in terms of partial derivatives of the form $\frac{\partial \alpha}{\partial \beta}$, where α and β can be any of $L, z_i, a_i, b_{ij}, w_{ij}, x_1$ for all valid values of i, j. Your backpropagation algorithm should be as explicit as possible - that is, make sure each partial derivative $\frac{\partial \alpha}{\partial \beta}$ cannot be decomposed further into simpler partial derivatives. Do not evaluate the partial derivatives.

$\frac{\partial L}{\partial b_{12}} = \frac{\partial L}{\partial z_3} \frac{\partial z_3}{\partial a_3} \frac{\partial a_3}{\partial z_2} \frac{\partial z_2}{\partial a_2} \frac{\partial a_2}{\partial b_{12}}$

13) Convolutional Neural Networks

Let's begin by considering some of the high-level components of a CNN kernel along with the basic motivation.

1. What is a kernel?

   A matrix or tensor of weights that is slid over an image tensor. At each position that is dictated by hyperparameters such as padding/stride/kernel-size, we perform elementwise multiplication of the kernel with the underlying part of the image tensor and sum the resulting entries to obtain an output value. (The kernel is sometimes also referred to as the convolutional weights or the filter.)

2. Why do we need stride, and what benefits/tradeoffs might different values of stride have on the output?

   The stride defines how many pixels the kernel moves with each step as it passes over the rows/columns of the image tensor. Larger values of stride can allow you to reduce the output dimensionality, which could combat overfitting, along with reducing computational power. The downside however, is that you lose more and more information with larger values of stride, which could limit the upside of your model's accuracy. Very small stride (i.e., stride = 1) preserves the size of the input image and can identify more fine-grained features, but comes with a greater computational cost.

3. What functionality does padding add to the kernel? Why might we want to use it?

   Padding surrounds the image tensor with rows/columns of zeros. By adding an appropriate amount, we can ensure that the output shape is the same as the input shape and allow every pixel to be included in the convolution. Furthermore, padding helps filters focus on the corner pixels just as much as middle pixels by making the filter pass over the corner pixels multiple times, as opposed to just once.

4. Consider the following image, filter, and output shape

$X =$

1	0	-2	3	4	1
2	9	5	6	0	-1
0	-3	1	3	4	4
6	5	2	0	6	8
-5	4	-3	1	3	-2
4	1	2	8	9	7

$F =$

-1	-1	-1
-1	8	-1
-1	-1	-1

$Y =$

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

   The shape of this particular Y is that of a kernel using no padding and a stride of 1.

   (a) Suppose we decide that, instead of having our output shape be $(4, 4)$, we want a slightly smaller, $(3, 3)$ image as output for the kernel. In order for this to happen, what is the smallest combination of stride and padding that would work?

   $s = 2, p = 1$. Notice that in this setting the kernel is never actually overlaid over the rightmost column or bottom most row of padding, since doing so would cause the kernel to go out of bounds.

(b) Let's make this a bit more general. Suppose our original image of shape (a, a), and we want the shape of our final image to be of shape (b, b), where $b \leq a$. Furthermore, the shape of the filter is (k, k), the stride length is s, and the padding is p. Express b in terms of all defined variables.
$b = \lfloor \frac{a-k+2p}{s} \rfloor + 1$