1. Remember the problem set we solved in the last class. (refer the questions and your answers if needed).
We solved the Separable SVM problem in 1D for

$$(-1, +1), (0, -1), (+1, -1)$$

We knew the solution: w as -2 and b as -1
Assume x was k times (say was measured in a different unit; remember the normalization of the data). For example, when $k = 2$, the data will look like:

$$(-2, +1), (0, -1), (+2, -1)$$

1.1 w and b will also become k times.
1.2 w will remain the same, while b will become $\frac{b}{k}$.
1.3 b will remain the same, while w will become $\frac{w}{k}$
1.4 w and b will remain the same.
1.5 No such systematic change is possible for w and b. The problem will have to be solved afreash.

Ans: C

2. Remember the problem set we solved in the last class. (refer the questions and your answers if needed).
   Consider the ExOR problem. There are four samples and four αs and four support vectors.
   We can generalize this observation as:
   "**For any linearly non-separable problem**, *number of support vectors is same as number of samples.*"
   Make the necessary minimal changes (if any required) and rewrite as true sentences in the space provided. Avoid changing the words in bold.
   FIB. "not same"

3. There is a popular problem called "parity". Consider **x** be d-dimensional, $d > 1$ which each $x_i \in \{-1, +1\}$ and y be $+1$ if the number of $+1$ in **x** is odd and else -1.0

   3.1 This problem is linearly separable.

   3.2 When $d = 2$, this problem reduced to *ExoR*.

   3.3 This is an example of a linearly non-separable problem.

   Ans: BC

4. Remember the problem set we solved in the last class. (refer the questions and your answers if needed).
   We solved the Separable SVM problem in 1D for

$$(-1, +1), (0, -1), (+1, -1)$$

   We knew the solution: w as -2 and b as -1
   Consider an extension, we add some small zero mean Gaussian noise $\mathcal{N}(0, \sigma^2)$ to each of the samples and create 10 variations each. (total of 30 samples). (Assume $\sigma = 0.5$.)
   We solve the SVM problem.
   - 4.1 We expect around 20 Support Vectors. (non zero αs)
   - 4.2 We will have only two Support Vectors.
   - 4.3 The optimal values of w and b will become 10 times.
   - 4.4 The optimal values of w and b will remain almost the same.
   - 4.5 Margin remains the same.
   - 4.6 None of the above.

   Ans: D

5. Remember the problem set we solved in the last class. (refer the questions and your answers if needed).

We solved SVM for a linearly in-separable data:

$$(-1, +1), (0, -1), (+1, +1)$$

and obtained: $\alpha_1 = \alpha_3 = 1$ and $\alpha_2 = 2$
"**Assume we had an additional (4th) sample** $(+2, +1)$ **in our data**, *the* α*s for the first three samples, will remain the same.*"
Make the necessary minimal changes (if any required) and rewrite as true sentences in the space provided. Avoid changing the words in bold.
FIB. Ans: no change

1 New Topics

1: K-Means Conceptual

1. **Write True or False** Objective function that K-Means optimizes is concave. False
2. **Write True or False** K-Means can never find the global optima. False
3. **Write True or False** Even though it is a non-convex optimization, K-Means can produce the global optima. True
4. **Write True or False** If K-Means is run twice with the same initialization, the final clusters in both cases will always be the same. Assume that distances of a point from any two centroids are never same. True
5. **Write True or False**If K-Means is run twice with random initialization, the final clusters in both cases will always be the same. False

2: K-Means Numerical

1. Consider the following 6 points in 1D.

   $$\{-5, 0, 1, 2, 3, 4\}$$

   We want to cluster these points with K-means. We initialize it as 1,2,2,2,2,2 as the initial labels for these five points respectively. What will be the final clusters? 1,2,2,2,2,2
2. Consider the following 5 points in 1D.

   $$\{-5, 0, 1, 2, 3, 4\}$$

   We want to cluster these points with K-means. We initialize it as 1,1,2,2,2,2 as the initial labels for these five points respectively. What will be the final clusters? 1,2,2,2,2,2 or 1,1,2,2,2,2
3. Consider the following 5 points in 1D.

   $$\{-5, 0, 1, 2, 3, 4\}$$

   We want to cluster these points with K-means. We initialize it as 1,1,1,2,2,2 as the initial labels for these five points respectively. What will be the final clusters? 1,2,2,2,2,2 or 1,1,2,2,2,2
4. Consider the following 5 points in 1D.

   $$\{-5, 0, 1, 2, 3, 4\}$$

   We want to cluster these points with K-means. We initialize it as 1,1,1,1,2,2 as the initial labels for these five points respectively. What will be the final clusters? 1,2,2,2,2,2 or 1,1,2,2,2,2
5. Consider the following 5 points in 1D.

   $$\{-5, 0, 1, 2, 3, 4\}$$

   We want to cluster these points with K-means. We initialize it as 1,2,1,2,1,2 as the initial labels for these five points respectively. What will be the final clusters? 1,2,2,2,2,2 or 1,1,2,2,2,2

3: Convolution Layer

1. A convolutional layer has 100 inputs and 5 channels of 100 outputs there with sufficient zero padding. The number of learnable parameters is: Each output channel is computed with 7 learnable weights.

   Total number of learnable parameters is:

   (a) 7
   (b) 5
   (c) 12
   (d) 35
   (e) None of the above

   D/E

2. A convolutional layer has 100 inputs and 5 channels of 100 outputs there with sufficient zero padding. The number of learnable parameters is: Each output channel is computed with 7 learnable weights.

   Total number of learnable parameters is:

   (a) 7
   (b) 5
   (c) 12
   (d) 35
   (e) None of the above

   D/E

3. A convolution layer has 2 input channel of size 1000 each. Output is computed over a 1-D window of length 11. There are 4 output channels. Stride is 2.

   How many learnable parameters exists in this layer?

   FIB 11*2*4/ 92

4. A convolution layer has 3 input channel of size 1000 each. Output is computed over a 1-D window of length 7. There are 5 output channels. Stride is 3.

   How many learnable parameters exists in this layer?

   FIB 7*3*5/ 110

5. A convolution layer has 3 input channel of size 100 each. Output is computed over a 1-D window of length 5. There are 7 output channels. Stride is 3.

   How many learnable parameters exists in this layer?

   FIB 5*3*7 /112

6. A convolution layer has 4 input channel of size 100 each. Output is computed over a 1-D window of length 5. There are 7 output channels. Stride is 2.

   How many learnable parameters exists in this layer?

   FIB 5*4*7 /147

7. A convolution layer has 3 input channel of size 100 each. Output is computed over a 1-D window of length 7. There are 7 output channels. Stride is 2.

   How many learnable parameters exists in this layer?

   FIB 7*3*7 /154

4: Recurrent Neuron

1. Consider a recurrent/feedback model of neural network given by

$$s_t = f(Ux_t + Ws_{t-1})$$

$$o_t = g(Vs_t)$$

   (a) U is a set of learnable parameters.
   (b) W is a set of learnable parameters
   (c) s is a set of learnable parameters
   (d) BP identifies optimal (may be local) parameters for each t
   (e) None of the above.

   AB

2. Consider a recurrent/feedback model of neural network given by

$$s_t = f(Ux_t + Ws_{t-1})$$

$$o_t = g(Vs_t)$$

   (a) V is a set of learnable parameters.
   (b) W is a set of learnable parameters
   (c) s is a set of learnable parameters
   (d) BP identifies optimal (may be local) parameters for each t
   (e) None of the above.

   AB

3. Consider a recurrent/feedback model of neural network given by

$$s_t = f(Ux_t + Ws_{t-1})$$

$$o_t = g(Vs_t)$$

   (a) U is a set of learnable parameters.
   (b) V is a set of learnable parameters
   (c) s is a set of learnable parameters
   (d) BP identifies optimal (may be local) parameters for each t
   (e) None of the above.

   AB

4. Consider a recurrent/feedback model of neural network given by

$$s_t = f(Ux_t + Ws_{t-1})$$

$$o_t = g(Vs_t)$$

   (a) U is a set of learnable parameters.
   (b) V is a set of learnable parameters
   (c) W is a set of learnable parameters
   (d) BP can be adapted appropriately to learn the optimal weights.
   (e) None of the above.

ABCD

5. Consider a recurrent/feedback model of neural network given by

$$s_t = f(Ux_t + Ws_{t-1})$$

$$o_t = g(Vs_t)$$

   (a) The suffix t is used to denote that such models are appropriate for processing sequences.

   (b) $f()$ and $g()$ are activation functions. (such as tanh or relu)

   (c) BP can be adapted appropriately to learn the optimal weights.

   (d) All of the above.

   ABCD/ABC/D

5: Momentum

1. Make the necessary minimal changes (if any required) and rewrite as true sentences in the space provided. Avoid changing the words in bold.

   Adding a momentum term *removes the problem of local minima* **in Neural network training.** does not remove

2. Make the necessary minimal changes (if any required) and rewrite as true sentences in the space provided. Avoid changing the words in bold.

   Adding a momentum term *guarantees global optimal solution* **in Neural network training.** does not guarantee

3. Make the necessary minimal changes (if any required) and rewrite as true sentences in the space provided. Avoid changing the words in bold.

   Adding a momentum term *slows down the* **Neural network training.** speeds up

4. Make the necessary minimal changes (if any required) and rewrite as true sentences in the space provided. Avoid changing the words in bold.

   Adding a momentum term *speeds up the* **Neural network training.** no change

5. Make the necessary minimal changes (if any required) and rewrite as true sentences in the space provided. Avoid changing the words in bold.

   Adding a momentum term *is very effective in when the slope of error surface is very small* **for Neural network training.** no change

2 New Questions

1: NN Learning

1. Consider the following question with multiple answers. Answers are mostly incorrect. Correctly answer in the space given with an answer approximately of equal length/detail (not an essay!).

   Q: Given the objective/loss of a neural network, why don't we differentiate with respect to the individual weights, equate to zero and find the optimal (may be local) solution in one step?

   Ans1: Yes; it is feasible. We do not do this because back-propagation is simpler to implement in python.

   Ans2: No; this is not possible. Because we do not know how to differentiate the loss with respect to individual weights. Possible but not feasible in reality because solving a non linear equation with a large number of variables (number of weights) is almost impossible

2: K-Means

1. Consider the following question with multiple answers. Answer is mostly incorrect. Correctly answer in the space given with an answer approximately of equal length/detail (not an essay!).

   Q: Consider the problem of running K Means with K=2 on the four points $(\pm 1, \pm 1)$. Is it true to say that this problem can have two optimal solutions as $C_1 = \{(+1,+1),(+1,-1)\}, C_2 = \{(-1,-1),(-1,+1)\}$ And $C_1 = \{(+1,+1),(-1,+1)\}, C_2 = \{(-1,-1),(+1,-1)\}$

   Ans1: Yes; it is true since both the clustering are identical.

   Ans2: Yes; both have the same objective, and identical clusters.

   Ans3: No; there is actually a third equal objective clustering $C_1 = \{(+1,+1),(-1,-1)\} C_2 = \{(-1,+1),(+1,-1)\}$

   Ans4: No; out of the two possibilities, one is a global optima and the other one is only a local one. Yes, both are optimal solutions and have the same objective even though the clustering is different

3: Compact Neural Networks

1. Consider the following question with multiple answers. Answer is mostly incorrect. Correctly answer in the space given with an answer approximately of equal length/detail (not an essay!).

   Q: We wish our neural network is small so that the number of parameters/memory is small. How do we achieve that, say for an MLP?

   Ans1: We start with a single layer perceptron and keep on adding new layers/weights one at a time and see when do we get the acceptable performance

   Ans2: We L2 regularize the MLP.

   Ans3: We make small/tiny weights to zero and retrain L1 regularization (Ans 3) or Dropout will achieve sparsity

4: Initialization

1. Consider the following question with multiple answers. Answer is mostly incorrect. Correctly answer in the space given with an answer approximately of equal length/detail (not an essay!).

   Q: We can initialize all the weights of an MLP as equal (say 1.0). This is considered to be bad. Why?

   Ans1: No; This is a perfectly fine initialization

   Ans2: Yes; this is not a good initialization. Since weights are equal, we can not find the derivatives

   Ans3: Yes; this is not a good initialization. Since weights are equal, all weights learn at constant speed and we will not get a sparse solution as network. Not a good initialization, all nodes of a layer for all layers except first layer will have same derivative and thus weights will remain always same across a layer. "Symmetry" needs to be broken.

5: Auto Encoder

1. Consider the following question with multiple answers. Answer is mostly incorrect. Correctly answer in the space given with an answer approximately of equal length/detail (not an essay!).

   Q: Consider an autoencoder where input and output are x. We train this network by minimizing the MSE loss of predicted and actual, using BP. Why does not loss become zero?

   Ans1: Neural networks can not learn identity function

   Ans2: Training by backpropagation will never allow to make loss zero or near zero. Loss does not become zero because we only want to minimize the loss on a non-convex surface locally in a neural network[OR] bottleneck layer inability to capture data perfectly causes non-zero loss"

3 Problems from Quiz 2

1: : SVM

1. Remember the SVM problem from the problems we solved in the class. (1D samples)

   $$(-1,+1),(0,-1),(+1,-1)$$

   we geometrically solved the problem and saw the optimal primal solution as $w=-2$ and $b=-1$
   Assume the samples were

   $$(-20,+1),(0,-1),(+20,-1)$$

   geometrically solve and give the answer as w=——,b=——
   FIB w=-0.1, b=-1

2. Remember the SVM problem from the problems we solved in the class. (1D samples)

   $$(-1,+1),(0,-1),(+1,-1)$$

   we geometrically solved the problem and saw the optimal primal solution as $w=-2$ and $b=-1$
   Assume the samples were

   $$(-10,-1),(0,+1),(+10,+1)$$

   geometrically solve and give the answer as w=——,b=——
   FIB w=0.2, b=1

3. Remember the SVM problem from the problems we solved in the class. (1D samples)

   $$(-1,+1),(0,-1),(+1,-1)$$

   we geometrically solved the problem and saw the optimal primal solution as $w=-2$ and $b=-1$
   Assume the samples were

   $$(-20,+1),(0,+1),(+20,-1)$$

   geometrically solve and give the answer as w=——,b=——
   FIB w=-0.1,b=1

4. Remember the SVM problem from the problems we solved in the class. (1D samples)

   $$(-10,+1),(0,-1),(+10,-1)$$

   we geometrically solved the problem and saw the optimal primal solution as $w=-2$ and $b=-1$
   Assume the samples were

   $$(-20,-1),(0,+1),(+20,+1)$$

   geometrically solve and give the answer as w=——,b=——
   FIB w=0.1, b=1

5. Remember the SVM problem from the problems we solved in the class. (1D samples)

   $$(-1,+1),(0,-1),(+1,-1)$$

   we geometrically solved the problem and saw the optimal primal solution as $w=-2$ and $b=-1$
   Assume the samples were

   $$(0,+1),(+10,-1),(+20,-1)$$

   geometrically solve and give the answer as w=——,b=——
   FIB w=-0.2, b=1

2: Backpropagation

1. Consider an MLP which is getting trained with Back Propagation for a multiclass classification problem.

   (a) The performance of the final model will depend on the initialization.

   (b) The performance of the final model will depend on the learning rate we use.

   (c) The performance of the final model will depend on the termination criteria we use.

   (d) The performance of the final model will depend on the loss function we use.

   (e) Exactly three of the above four are correct.

   ABCD

2. Consider an MLP which is getting trained with Back Propagation for a multiclass classification problem.

   (a) The optimization problem we solve is convex if the number of classes is two.

   (b) The optimization problem we solve is non-convex independent of the number of classes.

   (c) We typically terminate the training when we reach a local minima (ie., GD can not change the solution)

   (d) When we stop the training with an "early stopping criteria", the solution is often not a local minima.

   (e) None of the above.

   BC/BCD

3: Activation Functions

1. Consider the popular activation function Leaky-ReLu.

   (a) its gradient can be either positive or negative.

   (b) its value is always non-negative

   (c) it is an increasing function.

   (d) it is a non-decreasing function

   (e) all the above

   CD

2. Consider the popular activation function ReLu.

   (a) its gradient can be either positive or negative.

   (b) its value can be either positive or negative

   (c) it is an increasing function.

   (d) it is a non-decreasing function

   (e) all the above

   CD/D

4: DDAG

1 For N classes DDAG, we require (N choose 2) binary classifiers and need to evaluate N-1 binary classifiers during inferenceIf there are 6 classes, a DDAG based multi class classifier will require —— binary classifiers to build the. DDAG. FIB 15

2If there are 6 classes, a DDAG based multi class classifier will require evaluation of —— binary classifiers to make a decision. FIB 5

3If there are 11 classes, a DDAG based multi class classifier will require —— binary classifiers to build the. DDAG. FIB 55

4If there are 11 classes, a DDAG based multi class classifier will require evaluation of —— binary classifiers to make a decision. FIB 10

5"Since for a K class problem, DDAG uses ${}^{K}C_2$ classifiers, the final decision can be ambigous". (Write Tue or False) FIB False

5: MLP gradient

1. Consider an MLP with two inputs, three hidden neurons and one output neurons. Hidden neurons and output neurons have sigmoid activation. There is no bias. Output neuron has a MSE loss.

   Consider a sample $([5,5]^T, 0.7)$ i.e., $x = [5,5]^T$ and $y = 0.7$. We would like to update all the weights based on the gradient of the loss ($\mathcal{L}$). Assume that $w_{ij}^{[k]}$ connects ith neuron of layer k with jth neuron of layer k+1. Thus weights between input and hidden layer are $w_{11}^{[1]}, w_{21}^{[1]}, w_{12}^{[1]}, w_{22}^{[1]}, w_{13}^{[1]}, w_{23}^{[1]}$ and those between hidden layer and output layer are $w_{11}^{[2]}, w_{21}^{[2]}, w_{31}^{[2]}$ All weights are initialized with unity. Find the numerical value of $\frac{\partial \mathcal{L}}{\partial w_{11}^{[1]}}$. Answer upto 4 decimal places.

   FIB 0

2. Consider an MLP with two inputs, three hidden neurons and one output neurons. Hidden neurons and output neurons have sigmoid activation. There is no bias. Output neuron has a MSE loss.

   Consider a sample $([5,5]^T, 0.7)$ i.e., $x = [5,5]^T$ and $y = 0.7$. We would like to update all the weights based on the gradient of the loss ($\mathcal{L}$). Assume that $w_{ij}^{[k]}$ connects ith neuron of layer k with jth neuron of layer k+1. Thus weights between input and hidden layer are $w_{11}^{[1]}, w_{21}^{[1]}, w_{12}^{[1]}, w_{22}^{[1]}, w_{13}^{[1]}, w_{23}^{[1]}$ and those between hidden layer and output layer are $w_{11}^{[2]}, w_{21}^{[2]}, w_{31}^{[2]}$ All weights are initialized with unity.

   Find the numerical value of $\frac{\partial \mathcal{L}}{\partial w_{12}^{[1]}}$. Answer upto 4 decimal places.

   FIB 0

3. Consider an MLP with two inputs, three hidden neurons and one output neurons. Hidden neurons and output neurons have sigmoid activation. There is no bias. Output neuron has a MSE loss.

   Consider a sample $([5,5]^T, 0.7)$ i.e., $x = [5,5]^T$ and $y = 0.7$. We would like to update all the weights based on the gradient of the loss ($\mathcal{L}$). Assume that $w_{ij}^{[k]}$ connects ith neuron of layer k with jth neuron of layer k+1. Thus weights between input and hidden layer are $w_{11}^{[1]}, w_{21}^{[1]}, w_{12}^{[1]}, w_{22}^{[1]}, w_{13}^{[1]}, w_{23}^{[1]}$ and those between hidden layer and output layer are $w_{11}^{[2]}, w_{21}^{[2]}, w_{31}^{[2]}$ All weights are initialized with unity.

   Find the numerical value of $\frac{\partial \mathcal{L}}{\partial w_{13}^{[1]}}$. Answer upto 4 decimal places.

   FIB 0

4. Consider an MLP with two inputs, three hidden neurons and one output neurons. Hidden neurons and output neurons have sigmoid activation. There is no bias. Output neuron has a MSE loss.

   Consider a sample $([5,5]^T, 0.7)$ i.e., $x = [5,5]^T$ and $y = 0.7$. We would like to update all the weights based on the gradient of the loss ($\mathcal{L}$). Assume that $w_{ij}^{[k]}$ connects ith neuron of layer k with jth neuron of layer k+1. Thus weights between input and hidden layer are $w_{11}^{[1]}, w_{21}^{[1]}, w_{12}^{[1]}, w_{22}^{[1]}, w_{13}^{[1]}, w_{23}^{[1]}$ and those between hidden layer and output layer are $w_{11}^{[2]}, w_{21}^{[2]}, w_{31}^{[2]}$ All weights are initialized with unity.

   Find the numerical value of $\frac{\partial \mathcal{L}}{\partial w_{23}^{[1]}}$. Answer upto 4 decimal places.

   FIB 0

5. Consider an MLP with two inputs, three hidden neurons and one output neurons. Hidden neurons and output neurons have sigmoid activation. There is no bias. Output neuron has a MSE loss.

   Consider a sample $([5,5]^T, 0.7)$ i.e., $x = [5,5]^T$ and $y = 0.7$. We would like to update all the weights based on the gradient of the loss ($\mathcal{L}$). Assume that $w_{ij}^{[k]}$ connects ith neuron of layer k with jth neuron of layer k+1. Thus weights between input and hidden layer are $w_{11}^{[1]}, w_{21}^{[1]}, w_{12}^{[1]}, w_{22}^{[1]}, w_{13}^{[1]}, w_{23}^{[1]}$ and those between hidden layer and output layer are $w_{11}^{[2]}, w_{21}^{[2]}, w_{31}^{[2]}$ All weights are initialized with unity.

   Find the numerical value of $\frac{\partial \mathcal{L}}{\partial w_{11}^{[2]}}$. Answer upto 4 decimal places.

   FIB 0.0228

6. Consider an MLP with two inputs, three hidden neurons and one output neurons. Hidden neurons and output neurons have sigmoid activation. There is no bias. Output neuron has a MSE loss.

   Consider a sample $([5,5]^T, 0.7)$ i.e., $x = [5,5]^T$ and $y = 0.7$. We would like to update all the weights based on the gradient of the loss ($\mathcal{L}$). Assume that $w_{ij}^{[k]}$ connects ith neuron of layer k with jth neuron of layer k+1. Thus weights between input and hidden layer are $w_{11}^{[1]}, w_{21}^{[1]}, w_{12}^{[1]}, w_{22}^{[1]}, w_{13}^{[1]}, w_{23}^{[1]}$ and those between hidden layer and output layer are $w_{11}^{[2]}, w_{21}^{[2]}, w_{31}^{[2]}$ All weights are initialized with unity.

   Find the numerical value of $\frac{\partial \mathcal{L}}{\partial w_{31}^{[2]}}$. Answer upto 4 decimal places.

   FIB 0.0228

4 Programming Problems

1: Initialization

1. Consider the following implementation of Xavier initialization where weights are initialized as $W_l = \mathcal{N}(\mu = 0, \sigma^2 = \frac{1}{n_{l-1}})$ where W_l denotes weights of layer l and n_{l-1} denote number of nodes in layer $l-1$

```
def initialize_parameters_xavier(layers_dims):
    """
    Arguments:
    layer_dims — python array (list) containing the size of each layer.

    Returns:
    A dictionary containing your parameters "W1", ..., "WL":
        W1 — weight matrix of shape (layers_dims[1], layers_dims[0])
        ...
        WL — weight matrix of shape (layers_dims[L], layers_dims[L-1])
    """
```

```
    parameters = {}
    L = len(layers_dims) - 1 # integer representing the number of layers

    for l in range(1, L + 1):
        # Modify the RHS in the line below
        parameters['W' + str(l)] = None
    return parameters
```

You can use either numpy or PyTorch. Please write the line of code below the comment "Modify the RHS.." FIB: parameters['W' + str(l)] = np.sqrt(1/layers$_d ims[l-1])$*$np.random.randn(layers_dims[l], layers_dims[l-1])$

2. Consider the following implementation of Kalming He initialization where weights are initialized as $W_l = \mathcal{N}(\mu = 0, \sigma^2 = \frac{2}{n_{l-1}})$ where W_l denotes weights of layer l and n_{l-1} denote number of nodes in layer $l-1$

```
def initialize_parameters_kalminghe(layers_dims):
    """
    Arguments:
    layer_dims -- python array (list) containing the size of each layer.

    Returns:
    A dictionary containing your parameters "W1", ..., "WL":
        W1 -- weight matrix of shape (layers_dims[1], layers_dims[0])
        ...
        WL -- weight matrix of shape (layers_dims[L], layers_dims[L-1])
    """
    parameters = {}
    L = len(layers_dims) - 1 # integer representing the number of layers

    for l in range(1, L + 1):
        # Modify the RHS in the line below
        parameters['W' + str(l)] = None
    return parameters
```

You can use either numpy or PyTorch. Please write the line of code below the comment "Modify the RHS.." FIB

parameters['W' + str(l)] = np.sqrt(2/layers$_d ims[l-1])*np.random.randn(layers_dims[l], layers_dims[l-1])$

2: MLP parameters

1. Consider this model:

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(10,5, bias=True)
        self.fc2 = nn.Linear(5, 2, bias=True)
    def forward(self, x):
        x = self.fc1(x)
        x = nn.ReLU()(x)
        x = self.fc2(x)
        return x
```

What is the number of trainable parameters in this model

(a) Depends on the input

(b) 60

(c) 62

(d) 67

(e) None of these

D

2. Consider this model:

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(10,5, bias=False)
        self.fc2 = nn.Linear(5, 2, bias=True)
    def forward(self, x):
        x = self.fc1(x)
        x = nn.ReLU()(x)
        x = self.fc2(x)
        return x
```

What is the number of trainable parameters in this model

(a) Depends on the input

(b) 60

(c) 62

(d) 67

(e) None of these

C

3. Consider this model:

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(10,5, bias=False)
        self.fc2 = nn.Linear(5, 2, bias=False)
    def forward(self, x):
        x = self.fc1(x)
        x = nn.ReLU()(x)
        x = self.fc2(x)
        return x
```

What is the number of trainable parameters in this model

(a) Depends on the input

(b) 60

(c) 62

(d) 67

(e) None of these

B

4. Consider this model:

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(10,4, bias=True)
        self.fc2 = nn.Linear(4, 2, bias=True)
    def forward(self, x):
        x = self.fc1(x)
        x = nn.ReLU()(x)
        x = self.fc2(x)
        return x
```

What is the number of trainable parameters in this model

(a) Depends on the input

(b) 48

(c) 50

(d) 54

(e) None of these

D

5. Consider this model:

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(10,4, bias=False)
        self.fc2 = nn.Linear(4, 2, bias=True)
    def forward(self, x):
        x = self.fc1(x)
        x = nn.ReLU()(x)
        x = self.fc2(x)
        return x
```

What is the number of trainable parameters in this model

(a) Depends on the input

(b) 48

(c) 50

(d) 54

(e) None of these

C

6. Consider this model:

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(10,4, bias=False)
        self.fc2 = nn.Linear(4, 2, bias=False)
    def forward(self, x):
        x = self.fc1(x)
        x = nn.ReLU()(x)
        x = self.fc2(x)
        return x
```

What is the number of trainable parameters in this model

(a) Depends on the input

(b) 48

(c) 50

(d) 54

(e) None of these

B

3: Dropout

1. Consider the following implementation of a MLP layer with dropout in numpy:

```
def mlp_layer_with_dropout(X, W, b, keep_prob):
    """
    Arguments:
        X — input tensor
        W — weight of mlp layer
        b — bias of mlp layer
        keep_prob — Probability of each output not being zeroed out
    Returns:
        Output of MLP layer with dropout
    """
    Y = X.dot(W) + b
    # Modify the RHS of line below. You may find np.random.binomial API useful
    m = None
    return m * Y
```

Please write the line of code below the comment "Modify the RHS.."
FIB

You can check that your function is correct using the following test code:

```
X = np.random.randn(5,10)
W, b = np.random.randn(10,4), np.random.randn(1,4)
out = mlp_layer_with_dropout(X, W, b, 0.25)
assert out.shape==(5,4)
assert np.isclose(np.sum(out!=0)/out.size, 0.25, atol=0.1)
```

Reference: `https://numpy.org/doc/stable/reference/random/generated/numpy.random.binomial.html` np.random.binomial(1, p=keep$_p rob, size = Y.shape$)

2. Consider the following implementation of a MLP layer with dropout in numpy:

```
def mlp_layer_with_dropout(X, W, b, keep_prob):
    """
    Arguments:
        X — input tensor
        W — weight of mlp layer
        b — bias of mlp layer
        keep_prob — Probability of each output not being zeroed out
    Returns:
        Output of MLP layer with dropout
    """
    Y = X.dot(W) + b
    # Modify the RHS of line below. You may find np.random.binomial API useful
```

```
    m = None
    return m * Y
```

Please write the line of code below the comment "Modify the RHS.."
FIB

You can check that your code is correct using the following test code:

```
X = np.random.randn(5,10)
W, b = np.random.randn(10,4), np.random.randn(1,4)
out = mlp_layer_with_dropout(X, W, b, 0.45)
assert out.shape==(5,4)
assert np.isclose(np.sum(out!=0)/out.size, 0.45, atol=0.1)
```

Reference: `https://numpy.org/doc/stable/reference/random/generated/numpy.random.binomial.html` np.random.binomial(1, p=keep$_p$*rob*, *size* = *Y.shape*)

3. Consider the following implementation of a MLP layer with dropout in numpy:

```
def mlp_layer_with_dropout(X, W, b, keep_prob):
    """
    Arguments:
        X -- input tensor
        W -- weight of mlp layer
        b -- bias of mlp layer
        keep_prob -- Probability of each output not being zeroed out
    Returns:
        Output of MLP layer with dropout
    """
    Y = X.dot(W) + b
    # Modify the RHS of line below. You may find np.random.binomial API useful
    m = None
    return m * Y
```

Please write the line of code below the comment "Modify the RHS.."
FIB

You can check that your code is correct using the following test code:

```
X = np.random.randn(5,8)
W, b = np.random.randn(8,4), np.random.randn(1,4)
out = mlp_layer_with_dropout(X, W, b, 0.25)
assert out.shape==(5,4)
assert np.isclose(np.sum(out!=0)/out.size, 0.25, atol=0.1)
```

Reference: `https://numpy.org/doc/stable/reference/random/generated/numpy.random.binomial.html` np.random.binomial(1, p=keep$_p$*rob*, *size* = *Y.shape*)

4. Consider the following implementation of a MLP layer with dropout in numpy:

```
def mlp_layer_with_dropout(X, W, b, keep_prob):
    """
    Arguments:
        X -- input tensor
        W -- weight of mlp layer
        b -- bias of mlp layer
        keep_prob -- Probability of each output not being zeroed out
    Returns:
        Output of MLP layer with dropout
```

```
    """
    Y = X.dot(W) + b
    # Modify the RHS of line below. You may find np.random.binomial API useful
    m = None
    return m * Y
```

Please write the line of code below the comment "Modify the RHS.."
FIB

You can check that your code is correct using the following test code:

```
X = np.random.randn(5,8)
W, b = np.random.randn(8,4), np.random.randn(1,4)
out = mlp_layer_with_dropout(X, W, b, 0.45)
assert out.shape==(5,4)
assert np.isclose(np.sum(out!=0)/out.size, 0.45, atol=0.1)
```

Reference: `https://numpy.org/doc/stable/reference/random/generated/numpy.random.binomial.html` np.random.binomial(1, p=keep$_p$*rob*, *size* = *Y.shape*)

4: Convolution-I

1. Consider the following code snippet for convolution

```
conv = nn.Conv1d(in_channels=1,out_channels=5,
                 kernel_size=7,padding=3,bias=True)
input = torch.randn(12,1,100)
output = conv(input)
```

   What is the number of trainable parameters in conv?
   FIB 40

2. Consider the following code snippet for convolution

```
conv = nn.Conv1d(in_channels=1,out_channels=5,
                 kernel_size=7,padding=3,bias=False)
input = torch.randn(12,1,100)
output = conv(input)
```

   What is the number of trainable parameters in conv?
   FIB 35

3. Consider the following code snippet for convolution

```
conv = nn.Conv1d(in_channels=1,out_channels=5,
                 kernel_size=7,padding=3,bias=True)
input = torch.randn(12,1,50)
output = conv(input)
```

   What is the number of trainable parameters in conv?
   FIB 40

4. Consider the following code snippet for convolution

```
conv = nn.Conv1d(in_channels=1,out_channels=5,
                 kernel_size=7,padding=3,bias=False)
input = torch.randn(12,1,50)
output = conv(input)
```

What is the number of trainable parameters in conv?
FIB 35

5. Consider the following code snippet for convolution

```
conv = nn.Conv1d(in_channels=1,out_channels=5,stride=2,
                 kernel_size=7,padding=3,bias=True)
input = torch.randn(100,1,50)
output = conv(input)
```

What is the number of trainable parameters in conv?
FIB 40

6. Consider the following code snippet for convolution

```
conv = nn.Conv1d(in_channels=1,out_channels=5,stride=2,
                 kernel_size=7,padding=3,bias=False)
input = torch.randn(100,1,50)
output = conv(input)
```

What is the number of trainable parameters in conv?
FIB 35

5: Convolution-II

1. Consider the following code snippet for convolution

```
conv = nn.Conv1d(in_channels=1,out_channels=5,
                 kernel_size=7,padding=3,bias=True)
input = torch.randn(12,1,100)
output = conv(input)
```

What is the shape of output?
FIB (12,5,100)

2. Consider the following code snippet for convolution

```
conv = nn.Conv1d(in_channels=1,out_channels=5,
                 kernel_size=7,padding=3,bias=False)
input = torch.randn(12,1,100)
output = conv(input)
```

What is the shape of output?
FIB (12,5,100)

3. Consider the following code snippet for convolution

```
conv = nn.Conv1d(in_channels=1,out_channels=5,
                 kernel_size=7,padding=3,bias=True)
input = torch.randn(12,1,50)
output = conv(input)
```

What is the shape of output?
FIB (12,5,50)

4. Consider the following code snippet for convolution

```
conv = nn.Conv1d(in_channels=1,out_channels=5,
                 kernel_size=7,padding=3,bias=False)
input = torch.randn(12,1,50)
output = conv(input)
```

What is the shape of output?
FIB (12,5,50)

5. Consider the following code snippet for convolution

```
conv = nn.Conv1d(in_channels=1,out_channels=5,stride=2,
                 kernel_size=7,padding=3,bias=True)
input = torch.randn(100,1,50)
output = conv(input)
```

What is the shape of output?
FIB (100,5,25)

6. Consider the following code snippet for convolution

```
conv = nn.Conv1d(in_channels=1,out_channels=5,stride=2,
                 kernel_size=7,padding=3,bias=False)
input = torch.randn(100,1,50)
output = conv(input)
```

What is the shape of output?
FIB (100,5,25)

1. Consider the ReLU activation function for a neuron.
   1.1 Output is always non-negative.
   1.2 Output is always positive
   1.3 output is either one or zero.
   1.4 Output is same as input.
   1.5 None of the above.

   A
2. Consider the ReLU activation function for a neuron.
   Derivative of the ReLu function:
   2.1 continuous
   2.2 differentiable
   2.3 is Constant throughout
   2.4 can take two values.
   2.5 can never be negative

   DE
3. Consider an MLP with 2 inputs, 3 neurons in hidden and one output. Hidden neurons and output neuron uses ReLU Activation.
   Let the input be x_1 and x_2 and output be y. We train this with MSE loss.

3.1 If x_1, x_2 are negative, and y is positive for all the samples, this network can not be used for effective problem solving.

3.2 If x_1, x_2 are positive, and y is negative for all the samples, this network can not be used for effective problem solving.

3.3 This network can be effectively used irrespective of whether input or output is negative.

3.4 This network can not be useful if either input or output is negative.

3.5 None of the above.

B

4. Make the necessary minimal changes (if any required) and rewrite as true sentences in the space provided. Avoid changing the words in bold.
Consider a deep neural network with ReLU activations.
Since the gradient is same as input (which can be very large quantity), there is a chance of vanishing gradient problem.
gradient is 0 or 1; there is no vanishing gradient

5. Make the necessary minimal changes (if any required) and rewrite as true sentences in the space provided. Avoid changing the words in bold.

For leaky ReLu, gradients are *either positive or negative*.

gradients are always positive

1. We know that the VC dimension of a set of lines in 2D is 3. What is the VC dimension of a set of planes in 3D?
   - 1.1 $3+1=4$
   - 1.2 $2+2=2$
   - 1.3 $2 \times \frac{3}{4} = 6$
   - 1.4 Remains the same. i.e., 3
   - 1.5 None of the above

   Ans A urlhttp://work.caltech.edu/slides/slides07.pdf proves that VC dimension for linear perceptron in R^d is $d+1$

2. Make the necessary minimal changes (if any required) and rewrite as true sentences in the space provided. Avoid changing the words in bold.
   A Single Layer Perceptron **can solve ExOR problem**.
   FIB A multi layer perceptron can solve ExOR problem.

3. We know that $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. What is the derivative of $tanh(x)$
   - 3.1 $1 + \tanh(x)$
   - 3.2 $1 - \tanh^2(x)$
   - 3.3 $\tanh(x)(1 - \tanh(x))$

3.4 $1 + \tanh^2(x)$
3.5 None of the above

Ans: B

4. Make the necessary minimal changes (if any required) and rewrite as true sentences in the space provided. Avoid changing the words in bold.
Backpropagation algorithm *can guarantee (always find) the optimal solution/weights* **for a Multilayer Perceptron**.
FIB Backpropagation algorithm can not guarantee (always find) the optimal solution/weights for a Multilayer Perceptron. (MLP loss functions are non-convex in general)
5. An MLP has two inputs, two hidden layers of 3 neurons each and an output of two neurons. All the neurons have biases. The number of weights (or learnable parameters) is:
5.1 24
5.2 21
5.3 29
5.4 37
5.5 None of the above

Ans: C 3*(2+1) + 3*(3+1) + 2*(3+1)

1. About Backpropagation Algorithm:
   - 1.1 When the algorithm is allowed to run for many (say ∞) iterations, the loss becomes zero.
   - 1.2 When the algorithm is allowed to run for many (say ∞) iterations, the network will overfit.
   - 1.3 When the algorithm is allowed to run for many (say ∞) iterations, we will reach a local minima.
   - 1.4 When the algorithm is allowed to run for many (say ∞) iterations, we will reach the same local minima, irrespective of the initialization
   - 1.5 All the above are true.

   Ans: C

2. About Backpropagation Algorithm:
   Which may be a really **bad** termination crieria
   - 2.1 When no major change in loss, end.
   - 2.2 When all gradients are near zero, end.
   - 2.3 When learning rate is near zero, end.
   - 2.4 When loss is near zero, end.
   - 2.5 All the above are terrible termination crieria.

   Ans: C

3. Consider an MLP getting used for a three class classification problem.
   Output layer has three neurons and we use a cross entropy loss.
   - 3.1 If the accuracy of the training data is 100%, it implies that the loss might have been zero.
   - 3.2 If the loss is zero, implies that the MLP as a classifier has 100% accuracy on the training data.
   - 3.3 If the loss is zero, implies that the MLP as a classifier has 100% accuracy on the test data.
   - 3.4 If the accuracy of the test data is 100%, it implies that the loss computed on the training data might have been zero.
   - 3.5 None of the above.

   Ans: B Acc 100% only requires that the logits(unnormalized class scores) of correct class be higher than other classes.

4. Make the necessary minimal changes (if any required) and rewrite as true sentences in the space provided. Avoid changing the words in bold.
   An MLP has no activation in the output. It has sigmoid activity in all the hidden layers. *It can not be used to*

output negative values **because** *sigmoid outputs in [0,1] (no negative)* can be used output=wx+b can be -ve even if $x \geq 0$ if w or b are -ve for last layer

5. Make the necessary minimal changes (if any required) and rewrite as true sentences in the space provided. Avoid changing the words in bold.
   Backpropagation algorithm *can be understood as an iterative optimization algorithm.*

1. You might have read the notes on Kernels and SVMs at:
   `https://www.dropbox.com/s/qryziuo3u143q5e/KERNEL-REVIEW.pdf?dl=0`
   See the Sec 3.3 (equations)
   Why is $a \in R^+$ written there?
   1.1 It could have been $a \in R^-$
   1.2 a negative does not lead to K being PSD
   1.3 It is a typo.

   Ans: B

2. You might have read the notes on Kernels and SVMs at:
   `https://www.dropbox.com/s/qryziuo3u143q5e/KERNEL-REVIEW.pdf?dl=0`
   See the Sec 3.3. Assume $\alpha_i \in R^+$; $\beta_i \in R^+$; $\kappa_i(\mathbf{p}, \mathbf{q})$ being a valid kernel. Also K and L are some positive integers.
   Then a new kernel $\kappa(\cdot, \cdot) =$
   2.1 $\sum_{i=1}^{K} \kappa_i(\mathbf{p}, \mathbf{q})$ is a valid kernel.
   2.2 $\prod_{i=1}^{L} \kappa_i(\mathbf{p}, \mathbf{q})$ is a valid kernel.
   2.3 $\sum_{i=1}^{K} \alpha_i \kappa_i(\mathbf{p}, \mathbf{q})$ is a valid kernel.
   2.4 $\prod_{i=1}^{L} \beta_i \kappa_i(\mathbf{p}, \mathbf{q})$ is a valid kernel.

2.5 $\sum_{i=1}^{K} \alpha_i \kappa_i(\mathbf{p}, \mathbf{q}) + \prod_{i=1}^{L} \beta_i \kappa_i(\mathbf{p}, \mathbf{q})$ is a valid kernel

Ans: ABCDE

3. You might have read the notes on Kernels and SVMs at:
`https://www.dropbox.com/s/qryziuo3u143q5e/KERNEL-REVIEW.pdf?dl=0`
See the pseudo-code for Kernel Perceptron (Algorithm 3).
Assime the kernel to be $(\mathbf{x}^T \mathbf{y})^2$
   3.1 The initialization $\alpha_i = 0$ is a must. With no other initialization, this algorithm will not work (say will not converge)
   3.2 Step of computing Kernel Matrix (step 2) should have been in side the loop (repeat structure).
   3.3 Since this is now Kernelized, with any data (irrespective of whether the data is linearly separable or not), this algorithm will converge.
   3.4 For data that is linearly separable, this algorithm will give you a linear decision boundary.
   3.5 None of the above.

   Ans: E

4. You might have read the notes on Kernels and SVMs at:
`https://www.dropbox.com/s/qryziuo3u143q5e/`

KERNEL-REVIEW.pdf?dl=0

Look at the equation (94) related to the objective function:

4.1 This is an L1 softmargin SVM

4.2 This is an L2 softmargin SVM

4.3 There is a typo. ξ_i should be replaced as ξ_i^2

4.4 There is a typo. LHS will have to be $j(\mathbf{w}\xi)$, since ξ is another variable that we need to optimize.

4.5 None of the above.

Ans: A

5. You might have read the notes on Kernels and SVMs at:
https://www.dropbox.com/s/qryziuo3u143q5e/KERNEL-REVIEW.pdf?dl=0
Consider the decision making rule. "one side of a line (in 2D) is +ve class and other side of a line is -ve class" Figure 7 shows that VC dimension of a class of functions (lines) in 2D is 3.
What is the VC dimension in 1D for a function class. If $x > \theta$, positive, else negative.
Write your answer in the space provided.

(Sample answer (possibly incorrect): 1)
FIB Ans: 2 Can shatter every set of 2 points (though needs to shatter only one) but cannot shatter any set of 3 points

SMAI-M20-Lec 20 Review questions

IIIT Hyderabad

September 25, 2020

Review Question - I (one, none or more correct)

1. Consider the following three samples and their labels $((x_1, x_2), y)$:

$$\{((1,1),+),\ \ ((2,2),-),\ \ ((0,0),+)\}$$

Look at the perceptron update rule with $\eta = 0.1$

$$\mathbf{w}^{k+1} \leftarrow \mathbf{w}^k + \eta \sum_{\mathbf{x}_i \in \mathcal{E}} y_i \mathbf{x}_i$$

Classify as + ve if $\mathbf{w}^T\mathbf{x} \geq 0$ else - ve. Start $\mathbf{w}^0 = [-1, -1, 4]^T$. Find $\mathbf{w}^1$?

1.1 $\mathbf{w}^1$ is independent of η
1.2 $\mathbf{w}^1$ is parallel to $\mathbf{w}^0$, but different.
1.3 $\mathbf{w}^1$ will be the same as $\mathbf{w}^0$
1.4 Algorithm has converged. $\mathbf{w}^2$ will be the same as $\mathbf{w}^1$
1.5 None of the above

Ans: E

Review Question - II (one, none or more correct)

Consider the following three samples and their labels $((x_1, x_2), y)$:

$$\{((1,1),+),\ \ ((2,2),-),\ \ ((0,0),+)\}$$

Look at the perceptron update rule with $\eta = 0.1$

$$\mathbf{w}^{k+1} \leftarrow \mathbf{w}^k + \eta \sum_{\mathbf{x}_i \in \mathcal{E}} y_i \mathbf{x}_i$$

Classify as + ve if $\mathbf{w}^T\mathbf{x} \geq 0$ else - ve.
Start $\mathbf{w}^0 = [-1, -1, 2]^T$. Find $\mathbf{w}^1$?

1. $\mathbf{w}^1$ is independent of η
2. $\mathbf{w}^1$ is parallel to $\mathbf{w}^0$, but different.
3. $\mathbf{w}^1$ will be the same as $\mathbf{w}^0$
4. $\mathbf{w}^2$ will be the same as $\mathbf{w}^1$
5. None of the above

Ans: E

Review Question - III (one, none or more correct)

Consider the following three samples and their labels $((x_1, x_2), y)$:

$$\{((1,1),+),\ \ ((2,2),-),\ \ ((0,0),+)\}$$

Look at the perceptron update rule with $\eta = 0.1$

$$\mathbf{w}^{k+1} \leftarrow \mathbf{w}^k + \eta \sum_{\mathbf{x}_i \in \mathcal{E}} y_i \mathbf{x}_i$$

Classify as + ve if $\mathbf{w}^T\mathbf{x} \geq 0$ else - ve.
Start $\mathbf{w}^0 = [-1, -1, 1.9]^T$. Find $\mathbf{w}^1$?

1. $\mathbf{w}^1$ is independent of η
2. $\mathbf{w}^1$ is parallel to $\mathbf{w}^0$, but different.
3. $\mathbf{w}^1$ will be the same as $\mathbf{w}^0$
4. $\mathbf{w}^2$ will be the same as $\mathbf{w}^1$
5. None of the above

Ans: E

Review Question -IV (one, none or more correct)

Consider the following three samples and their labels $((x_1, x_2), y)$:

$$\{((1,1),+),\ \ ((2,2),-),\ \ ((0,0),+)\}$$

Look at the perceptron update rule with $\eta = 0.1$

$$\mathbf{w}^{k+1} \leftarrow \mathbf{w}^k + \eta \sum_{\mathbf{x}_i \in \mathcal{E}} y_i \mathbf{x}_i$$

Classify as + ve if $\mathbf{w}^T\mathbf{x} \geq 0$ else - ve.
Start $\mathbf{w}^0 = [1, -1, 0]^T$. Find $\mathbf{w}^1$?

1. $\mathbf{w}^1$ is independent of η
2. $\mathbf{w}^1$ is parallel to $\mathbf{w}^0$, but different.
3. $\mathbf{w}^1$ will be the same as $\mathbf{w}^0$
4. $\mathbf{w}^2$ will be the same as $\mathbf{w}^1$
5. None of the above

Ans: ACD

Review Question -V (one, none or more correct)

Consider the following three samples and their labels $((x_1, x_2), y)$:

$$\{((1,1),+),\ \ ((2,2),-),\ \ ((0,0),+)\}$$

Look at the perceptron update rule with $\eta = 0.1$

$$\mathbf{w}^{k+1} \leftarrow \mathbf{w}^k + \eta \sum_{\mathbf{x}_i \in \mathcal{E}} y_i \mathbf{x}_i$$

Classify as + ve if $\mathbf{w}^T\mathbf{x} \geq 0$ else - ve.
Start $\mathbf{w}^0 = [1, -1, 0]^T$. $\mathbf{w}^2$ is:

1. $[1, -1, 0]^T$
2. $[1.2, -0.8, 0.1]^T$
3. $[0.8, -1.2, -0.1]^T$
4. $[1.4, -0.6, 0.2]^T$
5. None of the above

Ans: A

1. A Fully connected layer has 100 neurons at input and 100 neurons at output. The number of parameters to learn is:
   - 1.1 1000
   - 1.2 10000
   - 1.3 200
   - 1.4 50
   - 1.5 None of the above

   Assume there is no bias. B
2. A convolutional layer has 100 inputs and 100 outputs there is sufficient zero padding. The number of learnable parameters is:
   - 2.1 10
   - 2.2 3
   - 2.3 5
   - 2.4 1
   - 2.5 Any of the above

   E
3. A convolutional layer has 100 inputs and 5 channels of 100 outputs there with sufficient zero padding. The number of

learnable parameters is: Each output channel is computed with 7 learnable weights.
Total number of learnable parameters is:

3.1 7
3.2 5
3.3 12
3.4 35
3.5 None of the above

D

4. A convolution layer has 3 input channels of size 100 each. Output is computed over a 1-D window of length 5. There are 7 output channels. Stride is 2.
   How many learnable parameters exist in this layer?
   FIB 105

5. We know that if there is no zero padding, the convolution output is smaller than the original. Consider an input of size/length 100. Convolution is carried over a window of length 7 with stride of 1. What is the length/size of output?
   FIB 94

1. Consider an MLP: 2 input, one hidden layer (3 neurons) and one output. All neurons use ReLU activation. No bias. We use MSE loss.
   Network is initialized with all weights as zero.
   1.1 Output is zero.
   1.2 All derivatives ($\frac{\partial L}{\partial w}$) are zero.
   1.3 Loss is zero.
   1.4 With Back propagation, weights won't change.
   1.5 None of the above.

   ABD

2. Consider an MLP: 2 input, one hidden layer (3 neurons) and one output. All neurons use ReLU activation. No bias. We use MSE loss.
   Network is initialized with all weights as non-zero but a small constant.
   2.1 Output is zero.
   2.2 All derivatives ($\frac{\partial L}{\partial w}$) are zero.
   2.3 Loss is zero.
   2.4 With Back propagation, weights won't change.
   2.5 None of the above.

E

3. Consider an MLP: 2 input, one hidden layer (3 neurons) and one output. All neurons use ReLU activation. No bias. We use MSE loss.

   What may be a better initialization, among the following?

   3.1 All weights the same.
   3.2 All the weights random and positive
   3.3 All the weights random and negative.
   3.4 Some weights random and positive and some weights random and negative.
   3.5 All of the above are equally good or equally bad.

   BCD

4. Consider an MLP: 2 input, one hidden layer (3 neurons) and one output. All neurons use ReLU activation. No bias. We use MSE loss.

   We use this network for regression to predict, say the mean temperature of tomorrow in Hyderabad, which is always positive.

   We train the network with sufficient amount of data, and follow good practices of training.

4.1 At the end of training, we are at a local minima.
4.2 At the end of training, we will be at a global minima.
4.3 At the end of training Loss will become zero.
4.4 At the end of training, all our weights will be positive.
4.5 None of the above.

A

5. Consider an MLP: 2 input, one hidden layer (3 neurons) and one output. All neurons use ReLU activation. No bias. We use MSE loss.
We initialize the network with good practices available/reported. Starting from the same initialization, we train the network multiple times with BP/GD. Starting from the same initialization, we train the network multiple times with BP/GD with momentum term also.
   5.1 Starting from the same initialization, the solution at epoch 100 remains same in all the runs, when the implementation was SGD.
   5.2 Starting from the same initialization, the solution at epoch 100 remains same in all the runs, when the implementation was batch GD.

5.3 Starting from the same initialization, the solution at epoch 100 remains same with or without momentum.

5.4 With an appropriate but fixed convergence criteria (say early stopping), models trained with and without momentum will be the same.

5.5 With an appropriate but fixed convergence criteria (say early stopping), models trained with and without momentum could be different.

BE the model without momentum is more likely to be in local minima. In GD, the gradient is from all samples; so will always be same in both runs whereas in SGD the sample are chosen in random, so the gradients are different

1. Which of the following regularization in NN (implemented in PyTorch) lead to sparse solutions?
   - 1.1 L1 regularization
   - 1.2 L2 regularization
   - 1.3 Dropout
   - 1.4 Data Augmentation
   - 1.5 None of the above

   AC
2. A sparse set of weights in a Deep MLP is preferred:
   - 2.1 it could lead to better generalization
   - 2.2 it is compact and fit in lesser memory
   - 2.3 it has many zeros and lesser amount of operations in forward pass
   - 2.4 it is easy to train when the number of weights/parameters are less
   - 2.5 All the above

   ABCDE
3. While re-using a trained network for a new task:
   - 3.1 We always prefer to take the later (towards the end) layer
   - 3.2 We always prefer to take an early(in the beginning) layer

3.3 Which layer is more appropriate depends on the tasks.

3.4 All the layers are equally useful.

3.5 None of the above.

B

4. It is believed that adding noise is some sort of regularization.

4.1 Adding noise to the input is useful.

4.2 Adding noise to the output/labels is useful. (for simplicity, assume the task is regression!).

4.3 Adding noise to the weights is useful.

4.4 Higher the noise the better the regularization.

4.5 Lower the noise the better the regularization

ACD

5. Consider a problem where we do data augmentation and early stopping.

5.1 With data augmentation, training accuracy is expected to increase.

5.2 With data augmentation, training accuracy may decrease.

5.3 With data augmentation, performance on the validation set is expected to increase.

5.4 With data augmentation, the iteration where we do early stop, will increase.

5.5 With data augmentation, the iteration where we do early stop, will decrease.

BCE

1. Which of the following regularization in NN (implemented in PyTorch) lead to sparse solutions?
   - 1.1 L1 regularization
   - 1.2 L2 regularization
   - 1.3 Dropout
   - 1.4 Data Augmentation
   - 1.5 None of the above

   AC
2. A sparse set of weights in a Deep MLP is preferred:
   - 2.1 it could lead to better generalization
   - 2.2 it is compact and fit in lesser memory
   - 2.3 it has many zeros and lesser amount of operations in forward pass
   - 2.4 it is easy to train when the number of weights/parameters are less
   - 2.5 All the above

   ABCDE
3. While re-using a trained network for a new task:
   - 3.1 We always prefer to take the later (towards the end) layer
   - 3.2 We always prefer to take an early(in the beginning) layer

3.3 Which layer is more appropriate depends on the tasks.

3.4 All the layers are equally useful.

3.5 None of the above.

B

4. It is believed that adding noise is some sort of regularization.

4.1 Adding noise to the input is useful.

4.2 Adding noise to the output/labels is useful. (for simplicity, assume the task is regression!).

4.3 Adding noise to the weights is useful.

4.4 Higher the noise the better the regularization.

4.5 Lower the noise the better the regularization

ACD

5. Consider a problem where we do data augmentation and early stopping.

5.1 With data augmentation, training accuracy is expected to increase.

5.2 With data augmentation, training accuracy may decrease.

5.3 With data augmentation, performance on the validation set is expected to increase.

5.4 With data augmentation, the iteration where we do early stop, will increase.

5.5 With data augmentation, the iteration where we do early stop, will decrease.

BCD

1. We saw an implementation of MLP in numpy and PyTorch. You may have noticed that the weights are initialized randomly. What happens if we we set all weights and biases to 0
   - 1.1 Weights do not change while training
   - 1.2 No problem: the model will converge nicely
   - 1.3 Overfitting issue
   - 1.4 Underfitting issue
   - 1.5 None of these

   Assume we are using ReLU activation A/AD
2. We saw this implementation of MLP model in PyTorch: class Net(nn.Module): def

   $_{i}nit_{(self):super(Net,self).}{}_{i}nit_{()Initializeallthelayerswithlearnableparametersself.fc1=nn.Linear(2,2,True)self.fc2=n}$

   def forward(self, x): Write the forward pass x = self.fc1(x) x = torch.sigmoid(x) x = self.fc2(x) x = torch.sigmoid(x) return x The output of this model is
   - 2.1 Always greater than zero
   - 2.2 Always less than one
   - 2.3 Can be negative as well as positive
   - 2.4 Always zero

2.5 None of these

AB

3. We saw this implementation of MLP in PyTorch: class Net(nn.Module): def $_{i}nit_{(self):super(Net,self).}{}_{i}nit_{()Initializeallthelayerswithlearnableparametersself.fc1=nn.Linear(2,2,True)self.fc2=n}$

def forward(self, x): Write the forward pass x = self.fc1(x) x = torch.sigmoid(x) x = self.fc2(x) x = torch.sigmoid(x) return x What happens if we remove the 2 lines with code x=torch.sigmoid(x) in the forward function

3.1 Syntax error

3.2 Math error

3.3 Model becomes Single layer perceptron

3.4 Model remains multi layer perceptron

3.5 None of these

Assume that we are working with the XOR data as given in the notebook shared. C

4. We saw the implementation of MLP in PyTorch with XOR data. What happens if we add 10 more hidden layers with 100 weights each with non-linear activation and train the model till loss is minimized.

4.1 Results in the same decision boundary
4.2 Results in a different decision boundary but still able to classify all 4 samples correctly.
4.3 Can not classify all 4 samples correctly.
4.4 None of these

B

5. Suppose instead of XOR data, we now want to work on NAND data. Model 1 is a MLP with a hidden layer with 2 neurons as we saw. Model 2 is a SLP.
   5.1 Model 1 can classify all 4 samples correctly but not model 2
   5.2 Model 2 can classify all 4 samples correctly but not model 1
   5.3 Both model 1 and model 2 can classify all 4 samples correctly.
   5.4 Neither model 1 nor model 2 can classify all 4 samples correctly.
   5.5 None of these

   C