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$_d ims[l-1])*np.random.randn(layers_d ims[l], layers_d ims[l-1])$

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$_d ims[l-1])*np.random.randn(layers_d ims[l], layers_d ims[l-1])$

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)