**Introduction to deep learning**

**LATEST SUBMISSION GRADE**

100%

1.

Question 1

What does the analogy “AI is the new electricity” refer to?

**1 / 1 point**

Similar to electricity starting about 100 years ago, AI is transforming multiple industries.

Through the “smart grid”, AI is delivering a new wave of electricity.

AI runs on computers and is thus powered by electricity, but it is letting computers do things not possible before.

AI is powering personal devices in our homes and offices, similar to electricity.

**Correct**

Yes. AI is transforming many fields from the car industry to agriculture to supply-chain...

2.

Question 2

Which of these are reasons for Deep Learning recently taking off? (Check the three options that apply.)

**1 / 1 point**

Neural Networks are a brand new field.

Deep learning has resulted in significant improvements in important applications such as online advertising, speech recognition, and image recognition.

**Correct**

These were all examples discussed in lecture 3.

We have access to a lot more data.

**Correct**

Yes! The digitalization of our society has played a huge role in this.

We have access to a lot more computational power.

**Correct**

Yes! The development of hardware, perhaps especially GPU computing, has significantly improved deep learning algorithms' performance.

3.

Question 3

Recall this diagram of iterating over different ML ideas. Which of the statements below are true? (Check all that apply.)

**1 / 1 point**

Being able to try out ideas quickly allows deep learning engineers to iterate more quickly.

**Correct**

Yes, as discussed in Lecture 4.

Faster computation can help speed up how long a team takes to iterate to a good idea.

**Correct**

Yes, as discussed in Lecture 4.

It is faster to train on a big dataset than a small dataset.

Recent progress in deep learning algorithms has allowed us to train good models faster (even without changing the CPU/GPU hardware).

**Correct**

Yes. For example, we discussed how switching from sigmoid to ReLU activation functions allows faster training.

4.

Question 4

When an experienced deep learning engineer works on a new problem, they can usually use insight from previous problems to train a good model on the first try, without needing to iterate multiple times through different models. True/False?

**1 / 1 point**

True

False

**Correct**

Yes. Finding the characteristics of a model is key to have good performance. Although experience can help, it requires multiple iterations to build a good model.

5.

Question 5

Which one of these plots represents a ReLU activation function?

**1 / 1 point**

Figure 1:

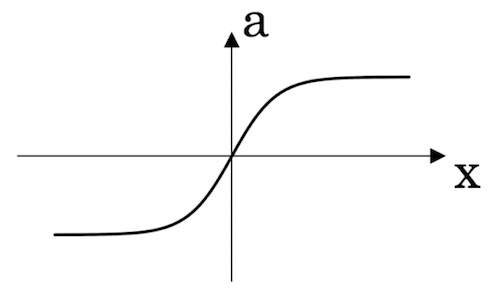


Figure 2:

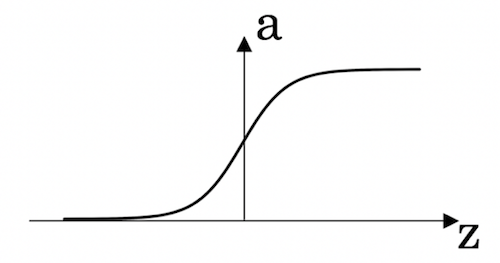


Figure 3:

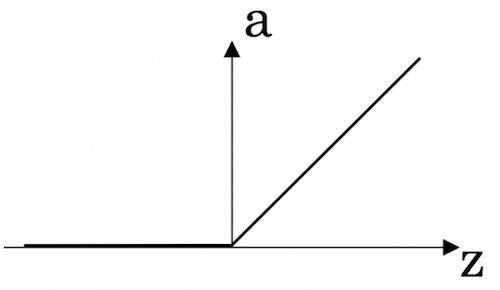
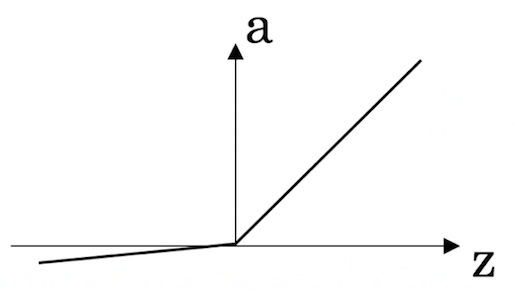


Figure 4:



**Correct**

Correct! This is the ReLU activation function, the most used in neural networks.

6.

Question 6

Images for cat recognition is an example of “structured” data, because it is represented as a structured array in a computer. True/False?

**1 / 1 point**

True

False

**Correct**

Yes. Images for cat recognition is an example of “unstructured” data.

7.

Question 7

A demographic dataset with statistics on different cities' population, GDP per capita, economic growth is an example of “unstructured” data because it contains data coming from different sources. True/False?

**1 / 1 point**

True

False

**Correct**

A demographic dataset with statistics on different cities' population, GDP per capita, economic growth is an example of “structured” data by opposition to image, audio or text datasets.

8.

Question 8

Why is an RNN (Recurrent Neural Network) used for machine translation, say translating English to French? (Check all that apply.)

**1 / 1 point**

It can be trained as a supervised learning problem.

**Correct**

Yes. We can train it on many pairs of sentences x (English) and y (French).

It is strictly more powerful than a Convolutional Neural Network (CNN).

It is applicable when the input/output is a sequence (e.g., a sequence of words).

**Correct**

Yes. An RNN can map from a sequence of english words to a sequence of french words.

RNNs represent the recurrent process of Idea->Code->Experiment->Idea->....

9.

Question 9

In this diagram which we hand-drew in lecture, what do the horizontal axis (x-axis) and vertical axis (y-axis) represent?

**1 / 1 point**

* x-axis is the input to the algorithm
* y-axis is outputs.
* x-axis is the amount of data
* y-axis (vertical axis) is the performance of the algorithm.
* x-axis is the amount of data
* y-axis is the size of the model you train.
* x-axis is the performance of the algorithm
* y-axis (vertical axis) is the amount of data.

**Correct**

10.

Question 10

Assuming the trends described in the previous question's figure are accurate (and hoping you got the axis labels right), which of the following are true? (Check all that apply.)

**1 / 1 point**

Decreasing the size of a neural network generally does not hurt an algorithm’s performance, and it may help significantly.

Decreasing the training set size generally does not hurt an algorithm’s performance, and it may help significantly.

Increasing the training set size generally does not hurt an algorithm’s performance, and it may help significantly.

**Correct**

Yes. Bringing more data to a model is almost always beneficial.

Increasing the size of a neural network generally does not hurt an algorithm’s performance, and it may help significantly.

**Correct**

Yes. According to the trends in the figure above, big networks usually perform better than small networks.

**Neural Network Basics**

**LATEST SUBMISSION GRADE**

90%

1.

Question 1

What does a neuron compute?

**1 / 1 point**

A neuron computes an activation function followed by a linear function (z = Wx + b)

A neuron computes the mean of all features before applying the output to an activation function

A neuron computes a linear function (z = Wx + b) followed by an activation function

A neuron computes a function g that scales the input x linearly (Wx + b)

**Correct**

Correct, we generally say that the output of a neuron is a = g(Wx + b) where g is the activation function (sigmoid, tanh, ReLU, ...).

2.

Question 2

Which of these is the "Logistic Loss"?

**1 / 1 point**

L(i)(y^(i),y(i))=∣y(i)−y^(i)∣2L(*i*)(*y*^​(*i*),*y*(*i*))=∣*y*(*i*)−*y*^​(*i*)∣2

L(i)(y^(i),y(i))=∣y(i)−y^(i)∣L(*i*)(*y*^​(*i*),*y*(*i*))=∣*y*(*i*)−*y*^​(*i*)∣

L(i)(y^(i),y(i))=−(y(i)log⁡(y^(i))+(1−y(i))log⁡(1−y^(i))L(*i*)(*y*^​(*i*),*y*(*i*))=−(*y*(*i*)log(*y*^​(*i*))+(1−*y*(*i*))log(1−*y*^​(*i*)))

L(i)(y^(i),y(i))=max(0,y(i)−y^(i))L(*i*)(*y*^​(*i*),*y*(*i*))=*max*(0,*y*(*i*)−*y*^​(*i*))

**Correct**

Correct, this is the logistic loss you've seen in lecture!

3.

Question 3

Suppose img is a (32,32,3) array, representing a 32x32 image with 3 color channels red, green and blue. How do you reshape this into a column vector?

**1 / 1 point**

x = img.reshape((32\*32\*3,1))

x = img.reshape((1,32\*32,\*3))

x = img.reshape((3,32\*32))

x = img.reshape((32\*32,3))

**Correct**

4.

Question 4

Consider the two following random arrays "a" and "b":

1

2

3

a = np.random.randn(2, 3) # a.shape = (2, 3)

b = np.random.randn(2, 1) # b.shape = (2, 1)

c = a + b

What will be the shape of "c"?

**1 / 1 point**

The computation cannot happen because the sizes don't match. It's going to be "Error"!

c.shape = (2, 1)

c.shape = (3, 2)

c.shape = (2, 3)

**Correct**

Yes! This is broadcasting. b (column vector) is copied 3 times so that it can be summed to each column of a.

5.

Question 5

Consider the two following random arrays "a" and "b":

1

2

3

a = np.random.randn(4, 3) # a.shape = (4, 3)

b = np.random.randn(3, 2) # b.shape = (3, 2)

c = a\*b

What will be the shape of "c"?

**1 / 1 point**

c.shape = (3, 3)

The computation cannot happen because the sizes don't match. It's going to be "Error"!

c.shape = (4, 3)

c.shape = (4,2)

**Correct**

Indeed! In numpy the "\*" operator indicates element-wise multiplication. It is different from "np.dot()". If you would try "c = np.dot(a,b)" you would get c.shape = (4, 2).

6.

Question 6

Suppose you have nx*nx*​ input features per example. Recall that X=[x(1)x(2)...x(m)]*X*=[*x*(1)*x*(2)...*x*(*m*)]. What is the dimension of X?

**1 / 1 point**

(nx,m)(*nx*​,*m*)

(m,nx)(*m*,*nx*​)

(1,m)(1,*m*)

(m,1)(*m*,1)

**Correct**

7.

Question 7

Recall that "np.dot(a,b)" performs a matrix multiplication on a and b, whereas "a\*b" performs an element-wise multiplication.

Consider the two following random arrays "a" and "b":

1

2

3

a = np.random.randn(12288, 150) # a.shape = (12288, 150)

b = np.random.randn(150, 45) # b.shape = (150, 45)

c = np.dot(a,b)

What is the shape of c?

**1 / 1 point**

The computation cannot happen because the sizes don't match. It's going to be "Error"!

c.shape = (150,150)

c.shape = (12288, 45)

c.shape = (12288, 150)

**Correct**

Correct, remember that a np.dot(a, b) has shape (number of rows of a, number of columns of b). The sizes match because :

"number of columns of a = 150 = number of rows of b"

8.

Question 8

Consider the following code snippet:

1

2

3

4

5

6

# a.shape = (3,4)

# b.shape = (4,1)

for i in range(3):

for j in range(4):

c[i][j] = a[i][j] + b[j]

How do you vectorize this?

**1 / 1 point**

c = a + b

c = a.T + b

c = a.T + b.T

c = a + b.T

**Correct**

9.

Question 9

Consider the following code:

1

2

3

a = np.random.randn(3, 3)

b = np.random.randn(3, 1)

c = a\*b

What will be c? (If you’re not sure, feel free to run this in python to find out).

**0 / 1 point**

This will invoke broadcasting, so b is copied three times to become (3,3), and ∗∗ is an element-wise product so c.shape will be (3, 3)

This will invoke broadcasting, so b is copied three times to become (3, 3), and ∗∗ invokes a matrix multiplication operation of two 3x3 matrices so c.shape will be (3, 3)

This will multiply a 3x3 matrix a with a 3x1 vector, thus resulting in a 3x1 vector. That is, c.shape = (3,1).

It will lead to an error since you cannot use “\*” to operate on these two matrices. You need to instead use np.dot(a,b)

**Incorrect**

10.

Question 10

Consider the following computation graph.

What is the output J?

**1 / 1 point**

J = (c - 1)\*(b + a)

J = (a - 1) \* (b + c)

J = a\*b + b\*c + a\*c

J = (b - 1) \* (c + a)

**Correct**

Yes. J = u + v - w = a\*b + a\*c - (b + c) = a \* (b + c) - (b + c) = (a - 1) \* (b + c).

**Shallow Neural Networks**

**LATEST SUBMISSION GRADE**

90%

1.

Question 1

Which of the following are true? (Check all that apply.)

**1 / 1 point**

a4[2]*a*4[2]​ is the activation output by the 4th4*th* neuron of the 2nd2*nd* layer

**Correct**

a4[2]*a*4[2]​ is the activation output of the 2nd2*nd* layer for the 4th4*th* training example

X*X* is a matrix in which each column is one training example.

**Correct**

a[2](12)*a*[2](12) denotes the activation vector of the 2nd2*nd* layer for the 12th12*th* training example.

**Correct**

a[2](12)*a*[2](12) denotes activation vector of the 12th12*th* layer on the 2nd2*nd* training example.

X*X* is a matrix in which each row is one training example.

a[2]*a*[2] denotes the activation vector of the 2nd2*nd* layer.

**Correct**

2.

Question 2

The tanh activation usually works better than sigmoid activation function for hidden units because the mean of its output is closer to zero, and so it centers the data better for the next layer. True/False?

**1 / 1 point**

True

False

**Correct**

Yes. As seen in lecture the output of the tanh is between -1 and 1, it thus centers the data which makes the learning simpler for the next layer.

3.

Question 3

Which of these is a correct vectorized implementation of forward propagation for layer l*l*, where 1≤l≤L1≤*l*≤*L*?

**1 / 1 point**

* Z[l]=W[l]A[l]+b[l]*Z*[*l*]=*W*[*l*]*A*[*l*]+*b*[*l*]
* A[l+1]=g[l+1](Z[l])*A*[*l*+1]=*g*[*l*+1](*Z*[*l*])
* Z[l]=W[l]A[l−1]+b[l]*Z*[*l*]=*W*[*l*]*A*[*l*−1]+*b*[*l*]
* A[l]=g[l](Z[l])*A*[*l*]=*g*[*l*](*Z*[*l*])
* Z[l]=W[l]A[l]+b[l]*Z*[*l*]=*W*[*l*]*A*[*l*]+*b*[*l*]
* A[l+1]=g[l](Z[l])*A*[*l*+1]=*g*[*l*](*Z*[*l*])
* Z[l]=W[l−1]A[l]+b[l−1]*Z*[*l*]=*W*[*l*−1]*A*[*l*]+*b*[*l*−1]
* A[l]=g[l](Z[l])*A*[*l*]=*g*[*l*](*Z*[*l*])

**Correct**

4.

Question 4

You are building a binary classifier for recognizing cucumbers (y=1) vs. watermelons (y=0). Which one of these activation functions would you recommend using for the output layer?

**1 / 1 point**

ReLU

Leaky ReLU

sigmoid

tanh

**Correct**

Yes. Sigmoid outputs a value between 0 and 1 which makes it a very good choice for binary classification. You can classify as 0 if the output is less than 0.5 and classify as 1 if the output is more than 0.5. It can be done with tanh as well but it is less convenient as the output is between -1 and 1.

5.

Question 5

Consider the following code:

1

2

A = np.random.randn(4,3)

B = np.sum(A, axis = 1, keepdims = True)

What will be B.shape? (If you’re not sure, feel free to run this in python to find out).

**1 / 1 point**

(1, 3)

(, 3)

(4, 1)

(4, )

**Correct**

Yes, we use (keepdims = True) to make sure that A.shape is (4,1) and not (4, ). It makes our code more rigorous.

6.

Question 6

Suppose you have built a neural network. You decide to initialize the weights and biases to be zero. Which of the following statements is true?

**1 / 1 point**

Each neuron in the first hidden layer will perform the same computation. So even after multiple iterations of gradient descent each neuron in the layer will be computing the same thing as other neurons.

Each neuron in the first hidden layer will perform the same computation in the first iteration. But after one iteration of gradient descent they will learn to compute different things because we have “broken symmetry”.

Each neuron in the first hidden layer will compute the same thing, but neurons in different layers will compute different things, thus we have accomplished “symmetry breaking” as described in lecture.

The first hidden layer’s neurons will perform different computations from each other even in the first iteration; their parameters will thus keep evolving in their own way.

**Correct**

7.

Question 7

Logistic regression’s weights w should be initialized randomly rather than to all zeros, because if you initialize to all zeros, then logistic regression will fail to learn a useful decision boundary because it will fail to “break symmetry”, True/False?

**1 / 1 point**

True

False

**Correct**

Yes, Logistic Regression doesn't have a hidden layer. If you initialize the weights to zeros, the first example x fed in the logistic regression will output zero but the derivatives of the Logistic Regression depend on the input x (because there's no hidden layer) which is not zero. So at the second iteration, the weights values follow x's distribution and are different from each other if x is not a constant vector.

8.

Question 8

You have built a network using the tanh activation for all the hidden units. You initialize the weights to relative large values, using np.random.randn(..,..)\*1000. What will happen?

**1 / 1 point**

This will cause the inputs of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow.

This will cause the inputs of the tanh to also be very large, causing the units to be “highly activated” and thus speed up learning compared to if the weights had to start from small values.

It doesn’t matter. So long as you initialize the weights randomly gradient descent is not affected by whether the weights are large or small.

This will cause the inputs of the tanh to also be very large, thus causing gradients to also become large. You therefore have to set α*α* to be very small to prevent divergence; this will slow down learning.

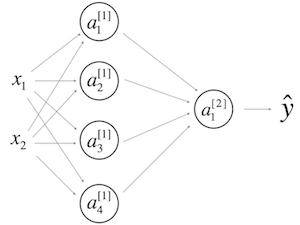
**Correct**

Yes. tanh becomes flat for large values, this leads its gradient to be close to zero. This slows down the optimization algorithm.

9.

Question 9

Consider the following 1 hidden layer neural network:



Which of the following statements are True? (Check all that apply).

**1 / 1 point**

W[1]*W*[1] will have shape (2, 4)

b[1]*b*[1] will have shape (4, 1)

**Correct**

W[1]*W*[1] will have shape (4, 2)

**Correct**

b[1]*b*[1] will have shape (2, 1)

W[2]*W*[2] will have shape (1, 4)

**Correct**

b[2]*b*[2] will have shape (4, 1)

W[2]*W*[2] will have shape (4, 1)

b[2]*b*[2] will have shape (1, 1)

**Correct**

10.

Question 10

In the same network as the previous question, what are the dimensions of Z[1]*Z*[1] and A[1]*A*[1]?

**0 / 1 point**

Z[1]*Z*[1] and A[1]*A*[1] are (1,4)

Z[1]*Z*[1] and A[1]*A*[1] are (4,m)

Z[1]*Z*[1] and A[1]*A*[1] are (4,1)

Z[1]*Z*[1] and A[1]*A*[1] are (4,2)

**Incorrect**

#### Key concepts on Deep Neural Networks

**LATEST SUBMISSION GRADE**

100%

1.

Question 1

What is the "cache" used for in our implementation of forward propagation and backward propagation?

**1 / 1 point**

It is used to cache the intermediate values of the cost function during training.

It is used to keep track of the hyperparameters that we are searching over, to speed up computation.

We use it to pass variables computed during forward propagation to the corresponding backward propagation step. It contains useful values for backward propagation to compute derivatives.

We use it to pass variables computed during backward propagation to the corresponding forward propagation step. It contains useful values for forward propagation to compute activations.

**Correct**

Correct, the "cache" records values from the forward propagation units and sends it to the backward propagation units because it is needed to compute the chain rule derivatives.

2.

Question 2

Among the following, which ones are "hyperparameters"? (Check all that apply.)

**1 / 1 point**

number of iterations

**Correct**

bias vectors b[l]*b*[*l*]

activation values a[l]*a*[*l*]

learning rate α*α*

**Correct**

number of layers L*L* in the neural network

**Correct**

weight matrices W[l]*W*[*l*]

size of the hidden layers n[l]*n*[*l*]

**Correct**

3.

Question 3

Which of the following statements is true?

**1 / 1 point**

The deeper layers of a neural network are typically computing more complex features of the input than the earlier layers.

The earlier layers of a neural network are typically computing more complex features of the input than the deeper layers.

**Correct**

4.

Question 4

Vectorization allows you to compute forward propagation in an L*L*-layer neural network without an explicit for-loop (or any other explicit iterative loop) over the layers l=1, 2, …,L. True/False?

**1 / 1 point**

True

False

**Correct**

Forward propagation propagates the input through the layers, although for shallow networks we may just write all the lines (a[2]=g[2](z[2])*a*[2]=*g*[2](*z*[2]), z[2]=W[2]a[1]+b[2]*z*[2]=*W*[2]*a*[1]+*b*[2], ...) in a deeper network, we cannot avoid a for loop iterating over the layers: (a[l]=g[l](z[l])*a*[*l*]=*g*[*l*](*z*[*l*]), z[l]=W[l]a[l−1]+b[l]*z*[*l*]=*W*[*l*]*a*[*l*−1]+*b*[*l*], ...).

5.

Question 5

Assume we store the values for n[l]*n*[*l*] in an array called layers, as follows: layer\_dims = [nx*nx*​, 4,3,2,1]. So layer 1 has four hidden units, layer 2 has 3 hidden units and so on. Which of the following for-loops will allow you to initialize the parameters for the model?

**1 / 1 point**

1

2

3

for(i in range(1, len(layer\_dims)/2)):

parameter[‘W’ + str(i)] = np.random.randn(layers[i], layers[i-1])) \* 0.01

parameter[‘b’ + str(i)] = np.random.randn(layers[i], 1) \* 0.01

1

2

3

for(i in range(1, len(layer\_dims)/2)):

parameter[‘W’ + str(i)] = np.random.randn(layers[i], layers[i-1])) \* 0.01

parameter[‘b’ + str(i)] = np.random.randn(layers[i-1], 1) \* 0.01

1

2

3

for(i in range(1, len(layer\_dims))):

parameter[‘W’ + str(i)] = np.random.randn(layers[i-1], layers[i])) \* 0.01

parameter[‘b’ + str(i)] = np.random.randn(layers[i], 1) \* 0.01

1

2

3

for(i in range(1, len(layer\_dims))):

parameter[‘W’ + str(i)] = np.random.randn(layers[i], layers[i-1])) \* 0.01

parameter[‘b’ + str(i)] = np.random.randn(layers[i], 1) \* 0.01

**Correct**

6.

Question 6

Consider the following neural network.



How many layers does this network have?

**1 / 1 point**

The number of layers L*L* is 4. The number of hidden layers is 3.

The number of layers L*L* is 3. The number of hidden layers is 3.

The number of layers L*L* is 4. The number of hidden layers is 4.

The number of layers L*L* is 5. The number of hidden layers is 4.

**Correct**

Yes. As seen in lecture, the number of layers is counted as the number of hidden layers + 1. The input and output layers are not counted as hidden layers.

7.

Question 7

During forward propagation, in the forward function for a layer l*l* you need to know what is the activation function in a layer (Sigmoid, tanh, ReLU, etc.). During backpropagation, the corresponding backward function also needs to know what is the activation function for layer l*l*, since the gradient depends on it. True/False?

**1 / 1 point**

True

False

**Correct**

Yes, as you've seen in the week 3 each activation has a different derivative. Thus, during backpropagation you need to know which activation was used in the forward propagation to be able to compute the correct derivative.

8.

Question 8

There are certain functions with the following properties:

(i) To compute the function using a shallow network circuit, you will need a large network (where we measure size by the number of logic gates in the network), but (ii) To compute it using a deep network circuit, you need only an exponentially smaller network. True/False?

**1 / 1 point**

True

False

**Correct**

9.

Question 9

Consider the following 2 hidden layer neural network:



Which of the following statements are True? (Check all that apply).

**1 / 1 point**

W[1]*W*[1] will have shape (4, 4)

**Correct**

Yes. More generally, the shape of W[l]*W*[*l*] is (n[l],n[l−1])(*n*[*l*],*n*[*l*−1]).

b[1]*b*[1] will have shape (4, 1)

**Correct**

Yes. More generally, the shape of b[l]*b*[*l*] is (n[l],1)(*n*[*l*],1).

W[1]*W*[1] will have shape (3, 4)

b[1]*b*[1] will have shape (3, 1)

W[2]*W*[2] will have shape (3, 4)

**Correct**

Yes. More generally, the shape of W[l]*W*[*l*] is (n[l],n[l−1])(*n*[*l*],*n*[*l*−1]).

b[2]*b*[2] will have shape (1, 1)

W[2]*W*[2] will have shape (3, 1)

b[2]*b*[2] will have shape (3, 1)

**Correct**

Yes. More generally, the shape of b[l]*b*[*l*] is (n[l],1)(*n*[*l*],1).

W[3]*W*[3] will have shape (3, 1)

b[3]*b*[3] will have shape (1, 1)

**Correct**

Yes. More generally, the shape of b[l]*b*[*l*] is (n[l],1)(*n*[*l*],1).

W[3]*W*[3] will have shape (1, 3)

**Correct**

Yes. More generally, the shape of W[l]*W*[*l*] is (n[l],n[l−1])(*n*[*l*],*n*[*l*−1]).

b[3]*b*[3] will have shape (3, 1)

10.

Question 10

Whereas the previous question used a specific network, in the general case what is the dimension of W^{[l]}, the weight matrix associated with layer l*l*?

**1 / 1 point**

W[l]*W*[*l*] has shape (n[l],n[l−1])(*n*[*l*],*n*[*l*−1])

W[l]*W*[*l*] has shape (n[l−1],n[l])(*n*[*l*−1],*n*[*l*])

W[l]*W*[*l*] has shape (n[l+1],n[l])(*n*[*l*+1],*n*[*l*])

W[l]*W*[*l*] has shape (n[l],n[l+1])(*n*[*l*],*n*[*l*+1])

**Correct**

True

#### Practical aspects of deep learning

**LATEST SUBMISSION GRADE**

90%

1.

Question 1

If you have 10,000,000 examples, how would you split the train/dev/test set?

**1 / 1 point**

33% train . 33% dev . 33% test

60% train . 20% dev . 20% test

98% train . 1% dev . 1% test

**Correct**

2.

Question 2

The dev and test set should:

**1 / 1 point**

Come from the same distribution

Come from different distributions

Be identical to each other (same (x,y) pairs)

**Have the same number of examples**

**Correct**

3.

Question 3

If your Neural Network model seems to have high bias, what of the following would be promising things to try? (Check all that apply.)

**1 / 1 point**

Increase the number of units in each hidden layer

**Correct**

Get more training data

Add regularization

Get more test data

Make the Neural Network deeper

**Correct**

4.

Question 4

You are working on an automated check-out kiosk for a supermarket, and are building a classifier for apples, bananas and oranges. Suppose your classifier obtains a training set error of 0.5%, and a dev set error of 7%. Which of the following are promising things to try to improve your classifier? (Check all that apply.)

**1 / 1 point**

Increase the regularization parameter lambda

**Correct**

Decrease the regularization parameter lambda

Get more training data

**Correct**

Use a bigger neural network

5.

Question 5

What is weight decay?

**1 / 1 point**

A technique to avoid vanishing gradient by imposing a ceiling on the values of the weights.

A regularization technique (such as L2 regularization) that results in gradient descent shrinking the weights on every iteration.

The process of gradually decreasing the learning rate during training.

Gradual corruption of the weights in the neural network if it is trained on noisy data.

**Correct**

6.

Question 6

What happens when you increase the regularization hyperparameter lambda?

**1 / 1 point**

Weights are pushed toward becoming smaller (closer to 0)

Weights are pushed toward becoming bigger (further from 0)

Doubling lambda should roughly result in doubling the weights

Gradient descent taking bigger steps with each iteration (proportional to lambda)

**Correct**

7.

Question 7

With the inverted dropout technique, at test time:

**0 / 1 point**

You do not apply dropout (do not randomly eliminate units), but keep the 1/keep\_prob factor in the calculations used in training.

You do not apply dropout (do not randomly eliminate units) and do not keep the 1/keep\_prob factor in the calculations used in training

You apply dropout (randomly eliminating units) and do not keep the 1/keep\_prob factor in the calculations used in training

You apply dropout (randomly eliminating units) but keep the 1/keep\_prob factor in the calculations used in training.

**Incorrect**

8.

Question 8

Increasing the parameter keep\_prob from (say) 0.5 to 0.6 will likely cause the following: (Check the two that apply)

**1 / 1 point**

Increasing the regularization effect

Reducing the regularization effect

**Correct**

Causing the neural network to end up with a higher training set error

Causing the neural network to end up with a lower training set error

**Correct**

9.

Question 9

Which of these techniques are useful for reducing variance (reducing overfitting)? (Check all that apply.)

**1 / 1 point**

Gradient Checking

L2 regularization

**Correct**

Xavier initialization

Data augmentation

**Correct**

Dropout

**Correct**

Exploding gradient

Vanishing gradient

10.

Question 10

Why do we normalize the inputs x*x*?

**1 / 1 point**

It makes it easier to visualize the data

It makes the cost function faster to optimize

Normalization is another word for regularization--It helps to reduce variance

It makes the parameter initialization faster

**Correct**

#### Optimization algorithms

**LATEST SUBMISSION GRADE**

100%

1.

Question 1

Which notation would you use to denote the 3rd layer’s activations when the input is the 7th example from the 8th minibatch?

**1 / 1 point**

a[3]{7}(8)*a*[3]{7}(8)

a[8]{7}(3)*a*[8]{7}(3)

a[8]{3}(7)*a*[8]{3}(7)

a[3]{8}(7)*a*[3]{8}(7)

**Correct**

2.

Question 2

Which of these statements about mini-batch gradient descent do you agree with?

**1 / 1 point**

One iteration of mini-batch gradient descent (computing on a single mini-batch) is faster than one iteration of batch gradient descent.

Training one epoch (one pass through the training set) using mini-batch gradient descent is faster than training one epoch using batch gradient descent.

You should implement mini-batch gradient descent without an explicit for-loop over different mini-batches, so that the algorithm processes all mini-batches at the same time (vectorization).

**Correct**

3.

Question 3

Why is the best mini-batch size usually not 1 and not m, but instead something in-between?

**1 / 1 point**

If the mini-batch size is 1, you end up having to process the entire training set before making any progress.

If the mini-batch size is m, you end up with batch gradient descent, which has to process the whole training set before making progress.

**Correct**

If the mini-batch size is 1, you lose the benefits of vectorization across examples in the mini-batch.

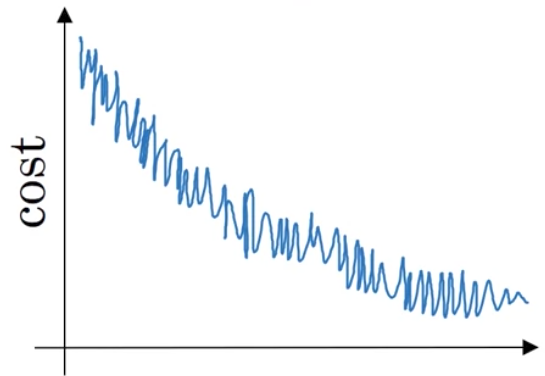
**Correct**

If the mini-batch size is m, you end up with stochastic gradient descent, which is usually slower than mini-batch gradient descent.

4.

Question 4

Suppose your learning algorithm’s cost J*J*, plotted as a function of the number of iterations, looks like this:



Which of the following do you agree with?

**1 / 1 point**

Whether you’re using batch gradient descent or mini-batch gradient descent, something is wrong.

If you’re using mini-batch gradient descent, something is wrong. But if you’re using batch gradient descent, this looks acceptable.

If you’re using mini-batch gradient descent, this looks acceptable. But if you’re using batch gradient descent, something is wrong.

Whether you’re using batch gradient descent or mini-batch gradient descent, this looks acceptable.

**Correct**

5.

Question 5

Suppose the temperature in Casablanca over the first three days of January are the same:

Jan 1st: θ1=10oC*θ*1​=10*oC*

Jan 2nd: θ210oC*θ*2​10*oC*

(We used Fahrenheit in lecture, so will use Celsius here in honor of the metric world.)

Say you use an exponentially weighted average with β=0.5*β*=0.5 to track the temperature: v0=0*v*0​=0, vt=βvt−1+(1−β)θt*vt*​=*βvt*−1​+(1−*β*)*θt*​. If v2*v*2​ is the value computed after day 2 without bias correction, and v2corrected*v*2*corrected*​ is the value you compute with bias correction. What are these values? (You might be able to do this without a calculator, but you don't actually need one. Remember what is bias correction doing.)

**1 / 1 point**

v2=10*v*2​=10, v2corrected=10*v*2*corrected*​=10

v2=7.5*v*2​=7.5, v2corrected=7.5*v*2*corrected*​=7.5

v2=7.5*v*2​=7.5, v2corrected=10*v*2*corrected*​=10

v2=10*v*2​=10, v2corrected=7.5*v*2*corrected*​=7.5

**Correct**

6.

Question 6

Which of these is NOT a good learning rate decay scheme? Here, t is the epoch number.

**1 / 1 point**

α=0.95tα0*α*=0.95*tα*0​

α=etα0*α*=*etα*0​

α=11+2∗tα0*α*=1+2∗*t*1​*α*0​

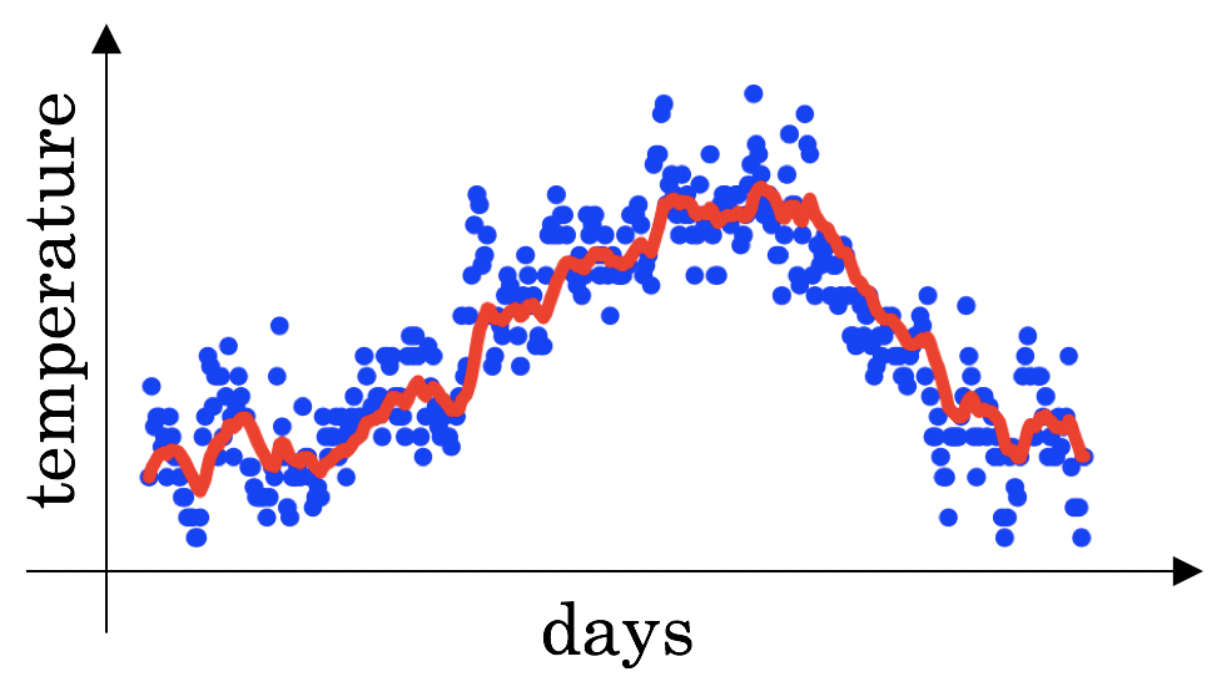
α=1tα0*α*=*t*​1​*α*0​

**Correct**

7.

Question 7

You use an exponentially weighted average on the London temperature dataset. You use the following to track the temperature: vt=βvt−1+(1−β)θt*vt*​=*βvt*−1​+(1−*β*)*θt*​. The red line below was computed using β=0.9*β*=0.9. What would happen to your red curve as you vary β*β*? (Check the two that apply)



**1 / 1 point**

Decreasing β*β* will shift the red line slightly to the right.

Increasing β*β* will shift the red line slightly to the right.

**Correct**

True, remember that the red line corresponds to β=0.9*β*=0.9. In lecture we had a green line $$\beta = 0.98) that is slightly shifted to the right.

Decreasing β*β* will create more oscillation within the red line.

**Correct**

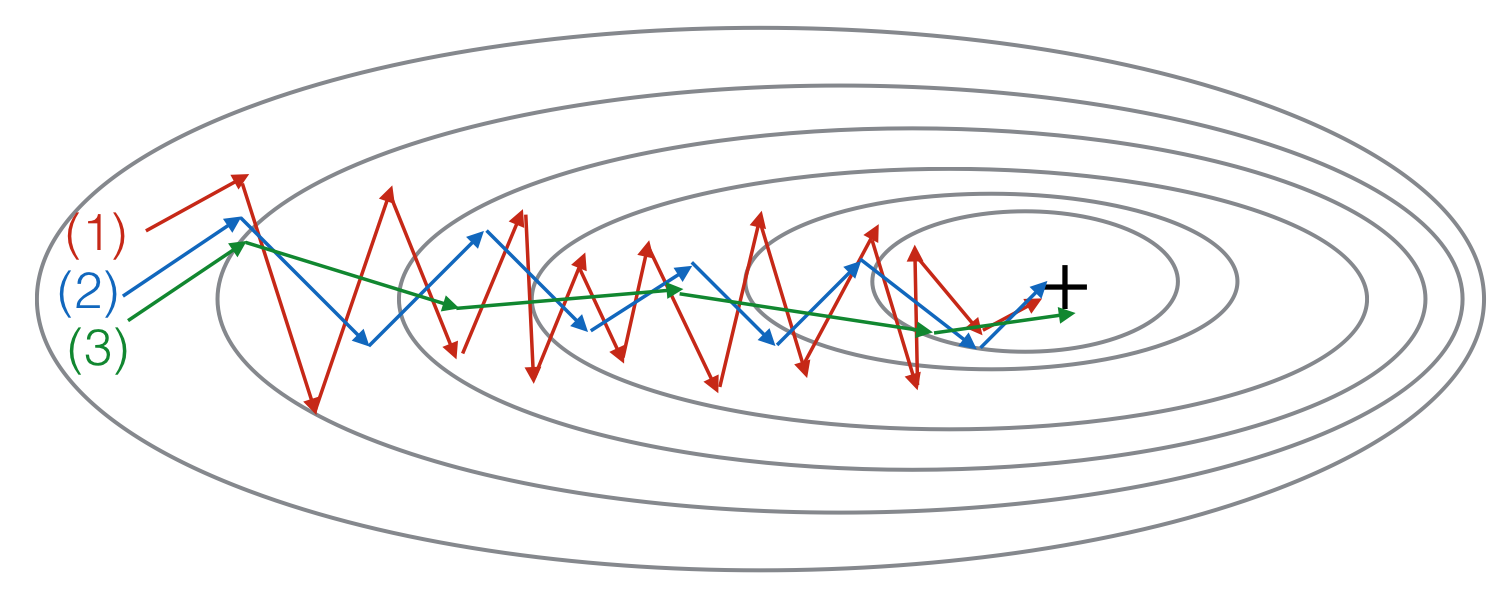
True, remember that the red line corresponds to β=0.9*β*=0.9. In lecture we had a yellow line $$\beta = 0.98 that had a lot of oscillations.

Increasing β*β* will create more oscillations within the red line.

8.

Question 8

Consider this figure:



These plots were generated with gradient descent; with gradient descent with momentum (β*β* = 0.5) and gradient descent with momentum (β*β* = 0.9). Which curve corresponds to which algorithm?

**1 / 1 point**

(1) is gradient descent. (2) is gradient descent with momentum (small β*β*). (3) is gradient descent with momentum (large β*β*)

(1) is gradient descent with momentum (small β*β*). (2) is gradient descent. (3) is gradient descent with momentum (large β*β*)

(1) is gradient descent. (2) is gradient descent with momentum (large β*β*) . (3) is gradient descent with momentum (small β*β*)

(1) is gradient descent with momentum (small β*β*), (2) is gradient descent with momentum (small β*β*), (3) is gradient descent

**Correct**

9.

Question 9

Suppose batch gradient descent in a deep network is taking excessively long to find a value of the parameters that achieves a small value for the cost function J(W[1],b[1],...,W[L],b[L])J(*W*[1],*b*[1],...,*W*[*L*],*b*[*L*]). Which of the following techniques could help find parameter values that attain a small value forJJ? (Check all that apply)

**1 / 1 point**

Try mini-batch gradient descent

**Correct**

Try better random initialization for the weights

**Correct**

Try initializing all the weights to zero

Try tuning the learning rate α*α*

**Correct**

Try using Adam

**Correct**

10.

Question 10

Which of the following statements about Adam is False?

**1 / 1 point**

We usually use “default” values for the hyperparameters β1,β2*β*1​,*β*2​ and ε*ε* in Adam (β1=0.9*β*1​=0.9, β2=0.999*β*2​=0.999, ε=10−8*ε*=10−8)

Adam should be used with batch gradient computations, not with mini-batches.

The learning rate hyperparameter α*α* in Adam usually needs to be tuned.

Adam combines the advantages of RMSProp and momentum

**Correct**

#### Hyperparameter tuning, Batch Normalization, Programming Frameworks

**LATEST SUBMISSION GRADE**

100%

1.

Question 1

If searching among a large number of hyperparameters, you should try values in a grid rather than random values, so that you can carry out the search more systematically and not rely on chance. True or False?

**1 / 1 point**

True

False

**Correct**

2.

Question 2

Every hyperparameter, if set poorly, can have a huge negative impact on training, and so all hyperparameters are about equally important to tune well. True or False?

**1 / 1 point**

True

False

**Correct**

Yes. We've seen in lecture that some hyperparameters, such as the learning rate, are more critical than others.

3.

Question 3

During hyperparameter search, whether you try to babysit one model (“Panda” strategy) or train a lot of models in parallel (“Caviar”) is largely determined by:

**1 / 1 point**

Whether you use batch or mini-batch optimization

The presence of local minima (and saddle points) in your neural network

The amount of computational power you can access

The number of hyperparameters you have to tune

**Correct**

4.

Question 4

If you think β*β* (hyperparameter for momentum) is between on 0.9 and 0.99, which of the following is the recommended way to sample a value for beta?

**1 / 1 point**

1

2

r = np.random.rand()

beta = r\*0.09 + 0.9

1

2

r = np.random.rand()

beta = 1-10\*\*(- r - 1)

1

2

r = np.random.rand()

beta = 1-10\*\*(- r + 1)

1

2

r = np.random.rand()

beta = r\*0.9 + 0.09

**Correct**

5.

Question 5

Finding good hyperparameter values is very time-consuming. So typically you should do it once at the start of the project, and try to find very good hyperparameters so that you don’t ever have to revisit tuning them again. True or false?

**1 / 1 point**

True

False

**Correct**

6.

Question 6

In batch normalization as presented in the videos, if you apply it on the l*l*th layer of your neural network, what are you normalizing?

**1 / 1 point**

a[l]*a*[*l*]

W[l]*W*[*l*]

z[l]*z*[*l*]

b[l]*b*[*l*]

**Correct**

7.

Question 7

In the normalization formula znorm(i)=z(i)−μσ2+ε*znorm*(*i*)​=*σ*2+*ε*​*z*(*i*)−*μ*​, why do we use epsilon?

**1 / 1 point**

In case μ*μ* is too small

To speed up convergence

To have a more accurate normalization

To avoid division by zero

**Correct**

8.

Question 8

Which of the following statements about γ*γ* and β*β* in Batch Norm are true?

**1 / 1 point**

β*β* and γ*γ* are hyperparameters of the algorithm, which we tune via random sampling.

The optimal values are γ=σ2+ε*γ*=*σ*2+*ε*​, and β=μ*β*=*μ*.

There is one global value of γ∈ℜ*γ*∈ℜ and one global value of β∈ℜ*β*∈ℜ for each layer, and applies to all the hidden units in that layer.

They set the mean and variance of the linear variable z[l]*z*[*l*] of a given layer.

**Correct**

They can be learned using Adam, Gradient descent with momentum, or RMSprop, not just with gradient descent.

**Correct**

9.

Question 9

After training a neural network with Batch Norm, at test time, to evaluate the neural network on a new example you should:

**1 / 1 point**

Use the most recent mini-batch’s value of μ*μ* and σ2*σ*2 to perform the needed normalizations.

Skip the step where you normalize using μ*μ* and σ2*σ*2 since a single test example cannot be normalized.

If you implemented Batch Norm on mini-batches of (say) 256 examples, then to evaluate on one test example, duplicate that example 256 times so that you’re working with a mini-batch the same size as during training.

Perform the needed normalizations, use μ*μ* and σ2*σ*2 estimated using an exponentially weighted average across mini-batches seen during training.

**Correct**

10.

Question 10

Which of these statements about deep learning programming frameworks are true? (Check all that apply)

**1 / 1 point**

A programming framework allows you to code up deep learning algorithms with typically fewer lines of code than a lower-level language such as Python.

**Correct**

Deep learning programming frameworks require cloud-based machines to run.

Even if a project is currently open source, good governance of the project helps ensure that the it remains open even in the long term, rather than become closed or modified to benefit only one company.

**Correct**