AI Research Labs

Deep Learning Foundations · Practice test 3 · 20 questions

Multi-layer Perceptrons

How to use this test. Answer all 20 questions without notes, then mark yourself with the answer key at the back. Each question has exactly one best answer. Score 75% or more and you've mastered the material; below that, reread the lessons for the questions you missed.

3.1 Layers, Matrices and Shapes

Q1 `calculate` How many trainable parameters does a 2 → 3 → 1 fully connected network have?

A 11
B 9
C 6
D 13

Q2 `calculate` How many parameters does a 100 → 50 → 10 network have?

A 6,000
B 5,060
C 5,560
D 5,500

Q3 `concept` In the lecture convention $y = f(Wx + b)$ with column vectors, a layer has 4 inputs and 6 neurons. What is the shape of W?

A 4×6
B 6×4
C 4×4
D 6×6

Q4 `apply` In NumPy you compute a layer as `X @ W + b`, with one example per row of `X`. A layer maps 4 inputs to 6 units. What shape must `W` have?

A 4×6
B 6×4
C 4×1
D 6×1

Q5 `interpret` A 784 → 128 → 10 network has 101,770 parameters. Where are almost all of them?

A In the 128 → 10 output connections
B Spread evenly between the two layers
C In the 784 → 128 connections
D In the biases of the two layers

3.2 Solving XOR with a Hidden Layer

Q6 concept In the 2–2–1 XOR network with sign units, what do the two hidden units compute?

A NAND and NOR of the two inputs
B OR and AND of the inputs
C XOR and its exact negation
D The two inputs, unchanged

Q7 calculate In that network the output is $\text{sign}(h_1 - h_2 - 1)$. For input $(1, 1)$ the hidden outputs are $(+1, +1)$. What is the output?

A −3
B 0
C +1
D −1

Q8 concept Why can one output neuron separate XOR once the hidden layer has done its work?

A The hidden layer adds extra dimensions, and every dataset is separable in more dimensions
B The output neuron uses a curved decision boundary, unlike the hidden units
C The hidden layer memorises the four input patterns and looks up the answer
D The hidden layer maps the inputs into coordinates where the classes are separable

Q9 interpret A classmate writes: "the hidden layer solves XOR by mapping the data to a higher-dimensional space." Is that accurate for the 2–2–1 network?

A Yes: the hidden layer always has more units than the inputs
B Yes: separability needs a jump from 2 to 3 dimensions here
C No: it stays 2-dimensional; the non-linearity is what helps
D No: the hidden layer actually reduces the dimension to 1

Q10 apply You replace the sign units with sigmoids and train a 2–2–1 network on XOR from random starts. What should you expect?

A It never succeeds, because sigmoids can't compute XOR
B It usually succeeds, but some starts get stuck
C It succeeds only if the learning rate is exactly 1
D It always finds the exact OR/AND weights shown above

3.3 Why Depth Helps: Composing Regions

Q11 concept With threshold units, what region can a single first-layer unit describe?

A A union of separate regions
B A convex polygon
C A circle around a point
D A half-plane

Q12 concept An AND-like unit in the second layer combines several half-plane tests. What shape can it carve out?

A. A convex region, such as a triangle
B. Any non-convex shape at all, such as a ring
C. Only a single straight line through the data
D. Two disconnected regions at the same time

Q13 apply You need to classify points that fall in either of two separate triangles. How many layers of threshold units does the construction need?

A. Four: each triangle needs its own separate hidden layer
B. One: a single layer can describe any union of shapes
C. Two: the second layer can already take unions of triangles
D. Three: lines, then triangles, then their union

Q14 interpret What is the general lesson of the half-plane → convex → union construction?

A. Deeper networks always need fewer parameters than shallow ones
B. Depth matters only for classification, never for regression
C. Every layer must use the threshold activation to be useful
D. Later layers compose features detected by earlier layers

Q15 concept Why does the geometric construction come with a warning not to overstate it?

A. It is simply wrong: depth adds no expressive power over a single layer
B. It only applies to networks with exactly three layers and no more than that
C. It assumes that the data are linearly separable before the first layer
D. It uses idealised threshold units, and depth also makes training harder

3.4 Why Non-linearity Is Essential

Q16 calculate Two linear layers: $W_1 = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$, $b_1 = (1, 0)$, $W_2 = \begin{bmatrix} 1 & -1 \end{bmatrix}$, $b_2 = 2$. What single layer are they equivalent to?

A. $y = x_1 + 2x_2 + 2$
B. $y = x_1 + x_2 + 3$
C. $y = x_1 - x_2 + 3$
D. $y = x_1 + x_2 + 2$

Q17 concept What do ten stacked layers with identity activations compute?

A. A single affine function $Wx + b$
B. A piecewise-linear function with ten pieces
C. Any continuous function, given enough units
D. A polynomial of degree ten in the inputs

Q18 apply Can a deep network whose hidden layers all use the identity activation solve XOR?

A Yes, if each hidden layer is wide enough

B Yes, if it has at least two hidden layers

C No: it collapses to one linear layer

D Only if it is trained for long enough with a small learning rate

Q19 interpret Where is a linear (identity) activation the right choice?

A At the output of a regression network

B In every hidden layer of a deep network

C At the output of a multi-class classifier

D Anywhere, since all activations are equivalent

Q20 concept What do non-linear activations give a multi-layer network that linear ones cannot?

A Faster training, because non-linear functions are cheaper

B Fewer parameters, because the activations have their own weights

C Guaranteed convergence to the global minimum of the loss

D Layers that bend the space, so depth adds expressive power

Answer key

Each answer comes with the reasoning, including why the most tempting wrong option is wrong.

Q1 **D** · lesson 3.1

Layer 1: 3×2 weights $+3$ biases $= 9$. Layer 2: $1 \times 3 + 1 = 4$. Total 13. 9 forgets the second layer, 6 counts only layer-1 weights, and 11 drops a layer's biases.

Q2 **C** · lesson 3.1

$100 \times 50 + 50 = 5{,}050$ and $50 \times 10 + 10 = 510$, total 5,560. 5,500 leaves out all biases, 5,060 drops the second layer's weights, and 6,000 multiplies the widths wrongly.

Q3 **B** · lesson 3.1

Each of the 6 neurons has a row of 4 weights, so W is outputs × inputs, 6×4, and Wx with x of length 4 gives 6 values. 4×6 is the code convention (XW), used with one example per row.

Q4 **A** · lesson 3.1

X is $N \times 4$; for XW to give $N \times 6$, W must be 4×6 (inputs × outputs), the transpose of the lecture's convention. Mixing the two conventions is a common source of shape errors.

Q5 **C** · lesson 3.1

$784 \times 128 = 100{,}352$ weights sit in the first layer; the second has only $1{,}290$ parameters. Parameters concentrate where wide layers meet, which is why flattened images need so many weights.

Q6 **B** · lesson 3.2

Unit 1 computes $x_1 + x_2 - 0.5$ (OR) and unit 2 computes $x_1 + x_2 - 1.5$ (AND). The output $h_1 - h_2 - 1$ is then "OR and not AND", which is XOR.

Q7 **D** · lesson 3.2

$1 - 1 - 1 = -1$, so the sign is -1: XOR(1,1) is false. $+1$ would be wrong for (1,1); 0 is the pre-activation of a different input; -3 would come from subtracting all terms.

Q8 **D** · lesson 3.2

The non-linear hidden mapping sends (1,0) and (0,1) to the same point and moves (0,0) and (1,1) apart, so a single line separates them. Here the hidden layer has the same dimension (2) as the input, so extra dimensions aren't the reason.

Q9 **C** · lesson 3.2

The hidden layer has 2 units, as many as the inputs. What makes the classes separable is the non-linear (sign) mapping, which folds the space. More dimensions can help in other settings, but they aren't needed here.

Q10 **B** · lesson 3.2

Sigmoid networks can represent XOR, and gradient descent finds a solution in most runs, but with only 2 hidden units a share of starts get stuck (units become identical or saturate). It finds some solution, rarely the hand-built one, and no single magic learning rate is required.

Q11 **D** · lesson 3.3

One unit tests which side of a line a point is on: a half-plane. Intersections of half-planes (convex polygons) need a second, AND-like layer; unions need a third, OR-like layer.

Q12 **A** · lesson 3.3

"All of these half-planes" is their intersection, which is always convex. Non-convex or disconnected regions need a further OR-like layer that takes the union of several convex pieces.

Q13 **D** · lesson 3.3

Layer 1 draws the six lines, layer 2 forms each triangle (AND of three lines), and layer 3 joins the two triangles (OR). That's the compositional picture of what each extra layer adds.

Q14 **D** · lesson 3.3

Depth lets a network build complex functions from simpler pieces, like edges → parts → objects in images. It doesn't guarantee fewer parameters in every case, doesn't require threshold units, and applies to regression too.

Q15 **D** · lesson 3.3

The picture assumes sign units, which can't be trained by gradients, and real deep networks face problems such as vanishing gradients. The construction itself is correct and general; it just isn't the whole story.

Q16 **B** · lesson 3.4

$W_2W_1 = [1 \cdot 1 + (-1) \cdot 0,\ 1 \cdot 2 + (-1) \cdot 1] = [1, 1]$ and $W_2b_1 + b_2 = 1 + 2 = 3$. Forgetting to push b_1 through W_2 gives a bias of 2; using only W_1's first row gives $x_1 + 2x_2$.

Q17 **A** · lesson 3.4

Each layer is affine, and composing affine maps gives another affine map: $W_2(W_1x + b_1) + b_2 = (W_2W_1)x + (W_2b_1 + b_2)$, and so on. Depth adds nothing without non-linear activations.

Q18 **C** · lesson 3.4

However many linear layers you stack, the result is one affine map whose decision boundary is a hyperplane, and XOR is not linearly separable. Width, depth and training time can't change that.

Q19 **A** · lesson 3.4

A regression output must be able to take any real value, so a linear output fits. In hidden layers it would make the layers collapse; a classifier needs softmax or sigmoid outputs to produce probabilities.

Q20 **D** · lesson 3.4

Between layers, a non-linearity stops the composition from simplifying to one affine map, so each layer can transform the representation (as the XOR hidden layer did). It doesn't make training cheaper, add parameters, or guarantee convergence.