AI Research Labs

Deep Learning Foundations · Practice test 5 · 15 questions

The Universal Approximation Theorem

How to use this test. Answer all 15 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.

5.1 The Universal Approximation Theorem

Q1 concept Which is a correct statement of the universal approximation theorem?

- A A deep network with many layers can approximate any function, but one hidden layer can't
- B A network trained by gradient descent always finds the best approximation to any function
- C Any neural network with one hidden layer approximates every function exactly, on any domain
- D One hidden layer with enough units can approximate any continuous function on a compact set

Q2 concept What does the "compact" in "compact set K" require of the inputs?

- A They lie on a straight line in the input space
- B They lie in a closed and bounded region
- C They have been standardised to mean zero
- D They are binary values, 0 or 1 only

Q3 concept Why does the theorem measure error with $\sup_{x \in K} |h(x) - \mathrm{NN}(x)|$?

- A The bound must hold at every point of K, the worst case
- B The bound only has to hold on average over all of K
- C It is the derivative of the loss with respect to x
- D It measures the error on the training points only

Q4 interpret Does the theorem apply to networks with ReLU activations?

- A Yes: it extends to any continuous non-polynomial activation
- B Only when the ReLU network has at least two hidden layers
- C No: it is only proved for the sigmoid activation function
- D No: ReLU networks can only represent linear functions of inputs

Q5 concept Why must the activation not be a polynomial for the theorem to hold?

- A Polynomials cannot be differentiated by backpropagation
- B Polynomials are too expensive to compute for large networks
- C Polynomials always produce outputs that are greater than 1
- D Polynomial units only build polynomials of bounded degree

5.2 Building Functions from Bumps

Q6 `concept` What does $\sigma(k(x-a)) - \sigma(k(x-b))$ look like for large k and $a < b$?

- A A smooth S-curve from 0 to 1 centred on $(a+b)/2$
- B About 1 to the left of a and 0 to the right of b
- C About 1 between a and b, about 0 elsewhere
- D About 2 between a and b, and 1 everywhere else

Q7 `calculate` Approximating a curve with 16 bumps, each built from two sigmoid units, needs how many hidden units?

- A 8
- B 32
- C 256
- D 16

Q8 `interpret` Going from 4 bumps to 16 bumps on the same interval mainly changes what?

- A The approximation becomes worse because there are more bumps to add
- B Smaller steps and lower error, at the cost of more units
- C Nothing, since the theorem says 4 bumps are enough for any function
- D The network becomes deeper, adding one more layer per extra bump

Q9 `concept` In the bump construction, where is the height of each bump set?

- A By the steepness k of its two sigmoids
- B By the number of hidden layers in the net
- C By the output weight on that bump's units
- D By the bias of the output unit on its own

Q10 `apply` Why is the bump construction not how a trained network actually works?

- A Because real networks never use sigmoid activations
- B Because bumps can only approximate linear functions
- C It's a proof idea, not what gradient descent finds
- D Because trained networks cannot represent bumps

5.3 What the Theorem Does Not Promise

Q11 `concept` Which of these does the universal approximation theorem guarantee?

- A That a suitable one-hidden-layer network exists
- B That the network will generalise well to new data
- C That the required number of units is reasonably small
- D That gradient descent will find the network's weights

Q12 interpret A 2–2–1 sigmoid network can represent XOR, yet about 20% of random starts fail to learn it. How does this relate to the theorem?

A. It shows the theorem is false for small networks
B. Representable is not the same as easy to train
C. It shows XOR is not a continuous function at all
D. It means the training data must contain errors

Q13 apply A student argues: "the theorem proves one hidden layer is enough, so deep networks are pointless." What's the best response?

A. One layer may need impractically many units; depth is often far more efficient
B. They're right, but only for regression; classification always needs depth
C. They're right: deep networks only exist because GPUs make them faster to train
D. They're wrong: the theorem actually requires at least two hidden layers to work

Q14 concept The theorem is about continuous functions. How does it still apply to classifiers with discrete labels?

A. It doesn't apply: classifiers are not covered by any approximation theorem
B. Discrete labels are continuous functions of the inputs, so nothing changes
C. The network outputs labels directly and rounds the approximation error away
D. The network approximates class probabilities, then a decision rule picks a label

Q15 interpret Does the theorem imply that a big enough network can't overfit?

A. No: enough capacity to fit any function can also fit noise
B. Yes: an exact approximation leaves no room for overfitting
C. Yes: the theorem bounds the error on new data as well
D. No, but only because the theorem needs infinite data

Answer key

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

Q1 **D** · lesson 5.1

The theorem: for a continuous target on a closed, bounded set and any tolerance ε, some one-hidden-layer network with finitely many units is within ε everywhere. It's approximate (not exact), needs a compact domain, says one layer suffices, and says nothing about training.

Q2 **B** · lesson 5.1

A compact subset of $\mathbb{R}^d$ is closed and bounded, such as $[0, 1]^d$. The guarantee is only about inputs in that region; it says nothing about extrapolating outside it. Binary or standardised inputs aren't what compactness means.

Q3 **A** · lesson 5.1

The supremum is the largest error anywhere on K, so keeping it below ε means the approximation is good everywhere at once (uniform approximation). An average-error bound would allow large errors on small regions.

Q4 **A** · lesson 5.1

Cybenko's original proof used sigmoids; Leshno and colleagues (1993) extended it to every continuous activation that isn't a polynomial, which includes ReLU. ReLU networks are piecewise linear, not linear.

Q5 **D** · lesson 5.1

If each unit is a polynomial of degree d, a one-hidden-layer network is a polynomial of degree at most d, and such a fixed-degree family can't approximate every continuous function. Polynomials are easy to differentiate and cheap; that isn't the issue.

Q6 **C** · lesson 5.2

The first steep sigmoid switches on at a, the second at b; subtracting leaves a "bump" of height about 1 on $[a, b]$. It's the difference of two steps, not a single S-curve.

Q7 **B** · lesson 5.2

Each bump uses two hidden units (one step up, one step down), so 16 bumps need 32. 16 counts bumps, not units; 256 squares the number of bumps.

Q8 **B** · lesson 5.2

Narrower intervals mean a finer staircase that tracks the target more closely, but each extra bump costs two more hidden units. The bumps all sit in the same single hidden layer; depth doesn't change.

Q9 **C** · lesson 5.2

The output unit sums the bumps weighted by its incoming weights, so each weight scales one bump to match the target's height in that interval. k controls how sharp the edges are, not the height.

Q10 **C** · lesson 5.2

The construction shows a one-hidden-layer network *can* represent any continuous function. Training finds whatever weights reduce the loss, usually much more distributed solutions. Sigmoids are used, bumps can approximate non-linear functions, and networks can represent bumps.

Q11 **A** · lesson 5.3

It is purely an existence result. It gives no training method or convergence guarantee, no practical bound on size, and nothing about generalisation or how much data is needed.

Q12 **B** · lesson 5.3

The theorem (and the explicit XOR construction) say suitable weights exist; they say nothing about whether gradient descent finds them from a given start. Failed runs get stuck with identical or saturated hidden units. Width 3+ succeeded every time.

Q13 **A** · lesson 5.3

The theorem allows an enormous number of units. Deep networks compose features and can represent many functions with far fewer units, and often train and generalise better on structured data. The theorem itself needs only one hidden layer, for regression or classification.

Q14 **D** · lesson 5.3

A classifier outputs probabilities through a sigmoid or softmax, which are continuous functions of the input that the network can approximate. A decision rule (threshold or argmax) then turns them into labels. Labels themselves jump, so they aren't continuous.

Q15 **A** · lesson 5.3

The theorem is about representing a function, not learning one from a finite noisy sample. A network flexible enough to approximate anything can memorise noise, which is exactly overfitting. The theorem says nothing about data or generalisation.