Quiz: Neural Networks

STA414/2104 - Winter 2026

1. **Batch vs. Stochastic Gradient Descent**
When training neural networks on very large datasets to minimize an average loss over data points $E(w) = \frac{1}{N}\sum L_n$, why is Stochastic Gradient Descent (SGD) or mini-batch learning heavily preferred over standard batch gradient descent when N is very large?

   (a) Evaluating the exact gradient requires computing a sum over all N data points, making each update step computationally prohibitive, whereas mini-batches provide a computationally efficient, unbiased estimator of the true gradient.

   (b) Mini-batching introduces stochastic noise that yields a biased but lower-variance approximation of the full batch gradient, which implicitly acts as a regularization term to prevent over-fitting.

   (c) SGD guarantees a faster theoretical convergence rate (in terms of the number of iterations) to the global optimum for strongly convex loss functions compared to full batch gradient descent.

   (d) Standard batch gradient descent requires storing and computing the inverse of the $N \times N$ Hessian matrix to guarantee descent, whereas SGD relies exclusively on first-order derivatives.

   Correct Answer: (a)
   Rationale: Batch gradient descent computes the exact gradient by averaging over all N data points, making a single update step $O(N)$, which is intractable for large datasets. SGD uses a random subset to compute an unbiased estimate of the gradient.

   Why the others are wrong: *(b) is incorrect because the mini-batch gradient is an unbiased estimator with higher variance than the full gradient, not lower.*
   (c) is incorrect because full batch gradient descent actually has a faster theoretical convergence rate per iteration (linear convergence) compared to SGD (sublinear convergence) for strongly convex functions (not seen in class). SGD is preferred only because the cost per iteration is drastically lower.
   (d) is incorrect because standard batch gradient descent is still a first-order method and does not require the Hessian matrix (that would be Newton's method).

2. **Universal Approximation Theorem**
The Universal Approximation Theorem guarantees that a feed-forward neural network can approximate any continuous function arbitrarily well, if deep/wide enough. A drawback is that neural networks can be more subject to overfitting. Which of the following architectural choices would completely prevent a multi-layer neural network from satisfying this property?

   (a) Using the Rectified Linear Unit (ReLU) activation function instead of a smooth, differentiable function like the logistic sigmoid.

   (b) Applying L^2 regularization (weight decay) to the network's parameters during optimization.

   (c) Using strictly linear activation functions (i.e., the identity function) across all hidden layers.

   (d) Training the network using Stochastic Gradient Descent (SGD) rather than full batch gradient descent.

   Correct Answer: (c)
   Rationale: Multi-layer feed-forward neural networks require non-linear activation functions to act as Universal Function Approximators. Any sequence of linear layers is mathematically equivalent to a single linear transformation (e.g., $y = W^{(3)}W^{(2)}W^{(1)}x = W'x$). Therefore, a deep network with strictly linear activation functions can only represent linear functions, completely preventing it from satisfying the universal approximation property for arbitrary continuous functions.

3. **Backpropagation and Computational Scaling**
In the context of evaluating gradients for a neural network, why do we use the backpropagation algorithm (computing error signals layer by layer) rather than calculating the full derivative of the loss with respect to each weight independently from scratch?

   (a) The multivariate chain rule is only mathematically valid when evaluated in reverse topological order.

   (b) Forward passes are not capable of evaluating the loss function accurately due to numerical instability in deep networks.

   (c) Independent derivative calculations from scratch involve an enormous amount of repeated terms, but backpropagation systematically computes and caches error signals to efficiently apply the chain rule.

   (d) Backpropagation strictly guarantees that the optimization landscape becomes convex, ensuring gradient descent converges.

   Correct Answer: (c)
   Rationale: For larger networks, calculating the gradient from scratch for every parameter becomes extremely cumbersome because terms like the derivative of the activation function ($\sigma'(z)$) appear repeatedly. By caching error signals (like $\overline{y}$ and $\overline{z}$) and passing them backward via the multivariate chain rule, backpropagation performs this efficiently.

4. **Over-fitting in Highly Expressive Networks**
Because multi-layer feed-forward neural networks with non-linear activation functions are universal function approximators, they possess extremely high expressivity. How does this impact training, and what is a standard approach to mitigate potential issues?

   (a) High expressivity naturally acts as a regularizer, so no additional techniques are needed to prevent over-fitting.

   (b) High expressivity makes learning trivial, causing gradient descent to converge almost immediately to the true underlying function.

   (c) High expressivity can easily lead to over-fitting (memorizing the training data), which can be mitigated by strategies like L^2 regularization or early stopping.

   (d) High expressivity causes persistent under-fitting, which must be fixed by removing hidden layers to simplify the model.

   Correct Answer: (c)
   Rationale: If a neural network can approximate any function arbitrarily well, it has the capacity to simply memorize noise in the training data, making over-fitting a serious concern. Applying L^2 regularization or stopping the training early (before generalization error starts to increase) are common ways to regularize highly expressive networks.

5. **Automatic Differentiation and Network Scaling**
Backpropagation is an implementation of reverse-mode automatic differentiation (AD). Why is backpropagation the standard algorithm for training modern deep neural networks?

   (a) Neural networks typically optimize a single scalar loss function with respect to millions of input parameters. Reverse-mode AD computes the full gradient in a single backward pass which is more efficient than with a forward pass.

   (b) Neural networks typically map a small number of input features to a massive number of output classes and reverse-mode AD efficiently handles large output spaces.

   (c) Reverse-mode AD requires significantly less memory.

   (d) Forward-mode AD is mathematically restricted to linear operations and cannot apply the chain rule through non-linear activation functions like ReLU or Sigmoid.

   Correct Answer: (a)
   Rationale: The computational efficiency of automatic differentiation depends on the function's dimensions: $f : \mathbb{R}^N \to \mathbb{R}^M$. Forward-mode AD computes the Jacobian one column at a time (requiring N passes), making it efficient when outputs outnumber inputs ($M \gg N$).

   Conversely, reverse-mode AD computes the Jacobian one row at a time (requiring M passes). Because training a neural network involves finding the derivative of a single scalar loss ($M = 1$) with respect to millions or billions of parameters ($N \gg 1$), reverse-mode AD can compute the entire gradient in just 1 pass.

   Note that option (c) is false: reverse-mode AD actually requires more memory because it must store the intermediate computational graph and forward activations to compute the chain rule backwards, compared to a single forward pass.