CSCE 790: Neural Networks and Their Applications

AIISC and Dept. Computer Science and Engineering
Email: vignar@sc.edu

© Vignesh Narayanan

September 19, 2023

UNIVERSITY OF
South Carolina

Two-layer feedforward NN - Example

$$y_i = \sigma(\sum_{l=0}^{L} w_{il} z_l), \quad i = 1, 2, \ldots, m.$$

$$z_l = \sigma(\sum_{j=0}^{n} v_{lj} x_j), \quad l = 1, 2, 3, \ldots, L.$$

$$L = \frac{1}{2} e' e = \frac{1}{2} \sum_{i=1}^{2} e_i^2, \quad e_i = t_i - y_i$$

$$w_{il} = w_{il} - \alpha \frac{\partial L}{\partial w_{il}}, \quad \frac{\partial L}{\partial w_{il}} = -z_l[\sigma'(u_i^2) e_i]$$

$$v_{lj} = v_{lj} - \alpha \frac{\partial L}{\partial v_{lj}}, \quad \frac{\partial L}{\partial v_{lj}} = -x_j[\sigma'(u_l^1) \sum_{i=1}^{2} w_{il}[\sigma'(u_i^2) e_i]]$$

Back-propagation - Training an N-layer NN

$$y_k = \sigma\left(\ldots(\sigma(\sum_{j=1}^{n} v_{lj}x_j + v_{l0})\right), \quad k = 1, 2, \ldots, m.$$

- Design choices
  - Number of hidden layers
  - Number of neurons in the hidden layers
  - Activation function
- Can the number of hidden layers $\to \infty$?
- Minimum number of layers?
- What happens when number of neurons $\to \infty$?

Vanishing Gradient

- What is the value of the derivative of sigmoid activation?
- Gradients 'vanish'?
- Recall: For sigmoid (σ) activation, $\sigma' = \sigma(1 - \sigma)$

Vanishing Gradient

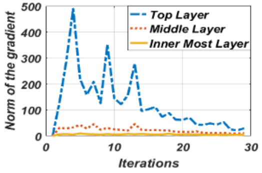

Figure: Norm of gradients used in error backpropagation

- What is the value of the derivative of sigmoid activation?
- Gradients 'vanish'?
- Recall: For sigmoid (σ) activation, $\sigma' = \sigma(1-\sigma)$

Rectified linear units (RelU)

- Rectified linear activation function?
- $f(x) = \max(0, x)$

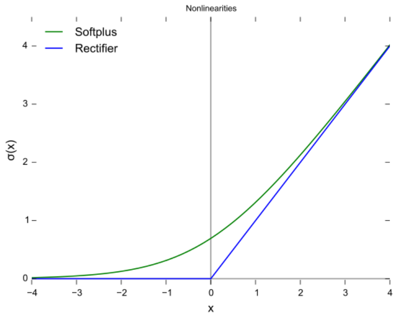

Figure: Rectified linear activation (deepai.org)

Universal approximation theorem (Representative result)

- Let $C(X, Y)$ denote the set of continuous functions from $X \to Y$.
- Let $\sigma \in C(\mathbb{R}, \mathbb{R})$
- Note that $(\sigma \circ x)_i = \sigma(x_i)$, where $\sigma \circ x$ denotes σ applied to each component of x.
- Then σ is not polynomial if and only if for every $n \in \mathbb{N}$, $m \in \mathbb{N}$, compact $K \subseteq \mathbb{R}^n$, $f \in C(K, \mathbb{R}^m), \varepsilon > 0$ there exist $k \in \mathbb{N}$, $A \in \mathbb{R}^{k\times n}$, $b \in \mathbb{R}^k$, $C \in \mathbb{R}^{m\times k}$ such that $\sup_{x\in K} \| f(x) - g(x) \| < \varepsilon$, where $g(x) = C \cdot (\sigma \circ (A \cdot x + b))$.

Allan Pinkus, January 1999. “Approximation theory of the MLP model in neural networks”

Rounding errors

- Representing continuous math on a digital computer is difficult - (infinitely many real numbers - finite number of bit patterns)
- For almost all real numbers, we incur some approximation error when we represent the number in the computer
- This is just the rounding error
- Algorithms work in theory yet fail in practice if they are not designed to minimize the accumulation of rounding error
  - Underflow occurs when numbers near zero are rounded to zero
  - Overflow occurs when numbers with large magnitude are approximated as ∞ or $-\infty$

Conditioning

- Conditioning - how rapidly a function changes with respect to small changes in its inputs
- Functions that change rapidly when their inputs are perturbed slightly can be problematic for scientific computation because rounding errors in the inputs can result in large changes in the output
- In a matrix, this is the ratio of the magnitude of the largest and smallest eigenvalue
- When this number is large, matrix inversion is particularly sensitive to error in the input
- This sensitivity is an intrinsic property of the matrix itself, not the result of rounding error during matrix inversion
- Poorly conditioned matrices amplify pre-existing errors when we multiply by the true matrix inverse
- The error will be compounded further by numerical errors in the inversion process itself

Gradients and Hessian

- Suppose we have a quadratic function (many functions that arise in practice are not quadratic but can be approximated well as quadratic functions locally)
  - If such a function has a second derivative of zero, then there is no curvature
  - It is a perfectly flat line, and its value can be predicted using only the gradient.
  - If the gradient is 1, then we can make a step of size ϵ along the negative gradient, and the cost function will decrease by ϵ

Curvature

- If the second derivative is negative, the function curves downward
- The function will actually decrease by more than ϵ
- If the second derivative is positive, the function curves upward
- The cost function can decrease by less than ϵ
- The (directional) second derivative tells us how well we can expect a gradient descent step to perform

Ill conditioning (DL, 2016)

- Some challenges arise even when optimizing convex functions
- Of these, the most prominent is ill-conditioning of the Hessian matrix H

$$f(\theta - \alpha\nabla f(\theta)) \approx f(\theta) - \alpha[\nabla f(\theta)]'\nabla f(\theta) + \alpha^2[\nabla f(\theta)]'\frac{H(\theta)}{2}\nabla f(\theta)$$

- Ill-conditioning of the gradient becomes a problem when the net sum to the cost is dominated by the term with Hessian
- One can monitor the squared gradient norm $[\nabla f(\theta)]'\nabla f(\theta)$ and the $[\nabla f(\theta)]'\frac{H(\theta)}{2}\nabla f(\theta)$ term

Gradient, Hessian, and Ill-conditioning

- In many cases, the gradient norm does not shrink significantly throughout learning, but the second term grows by more than an order of magnitude
- The result is that learning becomes very slow despite the presence of a strong gradient because the learning rate must be shrunk to compensate for even stronger curvature
- There are three terms here: the original value of the function, the expected improvement due to the slope of the function, and the correction we must apply to account for the curvature of the function
- When this last term is too large, the gradient descent step can actually move uphill

Gradient, Hessian, and Ill-conditioning

- When $[\nabla f(\theta)]' H [\nabla f(\theta)]$ is zero or negative,the Taylor series approximation predicts that increasing α forever will decrease f forever
- In practice, the Taylor series is unlikely to remain accurate for large α, so one must resort to more heuristic choices of α in this case
- When $[\nabla f(\theta)]' H [\nabla f(\theta)]$ is positive, solving for the optimal step size that decreases the Taylor series approximation of the function the most yields

$$\alpha^* = \frac{[\nabla f(\theta)]'[\nabla f(\theta)]}{[\nabla f(\theta)]' H [\nabla f(\theta)]}$$

Gradient descent

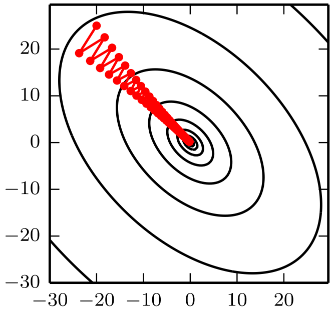

Figure: Conditioning and its effect

- Gradient descent fails to exploit the curvature information contained in the Hessian matrix
- Here we use gradient descent to minimize a quadratic function $f(\theta)$ whose Hessian matrix has condition number 5
- This means that the direction of most curvature has five times more curvature than the direction of least curvature
- In this case, the most curvature is in the direction $[1, 1]$, and the least curvature is in the direction $[1, -1]$
- The red lines indicate the path followed by gradient descent

Gradient descent

- This very elongated quadratic function resembles a long canyon
- Gradient descent wastes time repeatedly descending canyon walls because they are the steepest feature
- Since the step size is somewhat too large, it has a tendency to overshoot the bottom of the function and thus needs to descend the opposite canyon wall on the next iteration
- The large positive eigenvalue of the Hessian corresponding to the eigenvector pointed in this direction indicates that this directional derivative is rapidly increasing
- An optimization algorithm based on the Hessian could predict that the steepest direction is not actually a promising search direction in this context – There are other algorithms that exploit this information!