Week 9 - Derivative, Automatic Differentiation (AD)

Recap of Neural Networks

- Activation function between each layer
- Neural network input layer: each neural is decided by the **number of attributes** (feature) of input, AKA **dimension of data**.
- **Matrix multiplication**: n must be equal (column of the first matrix and row of the second matrix, or the other way around)

$$R = W_{m*n} * X_{n*p}$$

- **Weight-sharing**, forcing certain connections between nodes to have the same weight, is sensible for certain special applications. CNN is an example of applying it.

- ReLU activation nonlinearity,
    - u' - derivative of u
    - $\mathbb{I}$ - indicator function, f(x)=1 if true, else 0
    - [u ≥ 0] - Only select the part where u ≥ 0

Derivative

- Formulas for calculating derivatives
  - formulas for calculating the derivatives of elementary functions

$$\frac{d}{dt}(t^n) = nt^{n-1}$$

$$\frac{d}{dt}(e^t) = e^t$$

$$\frac{d}{dt}(ln\, t) = \frac{1}{t}$$

$$\frac{d}{dt}(sin\, t) = cos\, t$$

$$\frac{d}{dt}(cos\, t) = -sin\, t$$

$$\frac{d}{dt}(tan\, t) = sec^2 t$$

  - structural type formulas: when applying, it can be said differentiating "term by term"
    - The derivative of a constant times a function equals the constant times the derivative of the function.

$$\frac{d}{dt}cu = c\frac{du}{dt}$$

    - The derivative of a sum equals the sum of the derivatives.

$$\frac{d}{dt}(u+v) = \frac{du}{dt} + \frac{dv}{dt}$$

    - The derivative of a difference equals the difference of the derivatives.

$$\frac{d}{dt}(u-v) = \frac{du}{dt} - \frac{dv}{dt}$$

  - Using structural rules along with power rules

$$\frac{dc}{dt} = 0$$

$$\frac{d}{dt}(ct) = 0$$

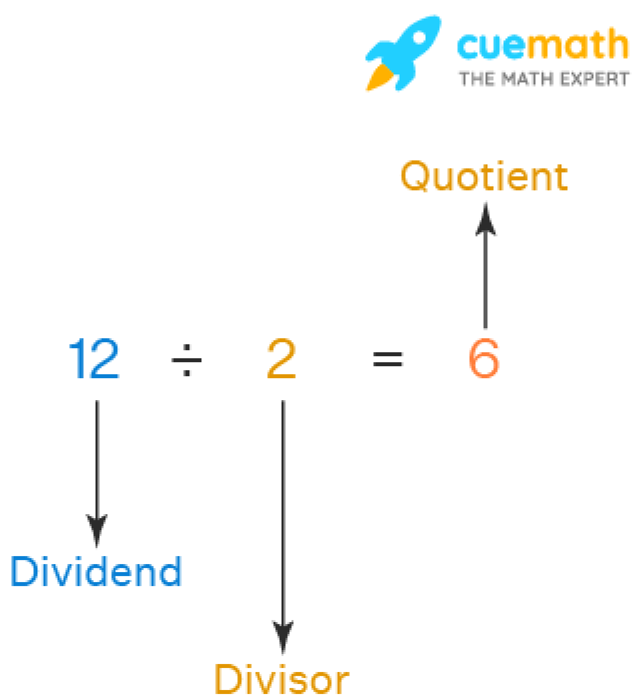
$$\frac{d}{dt}(at + b) = a$$

- Product rule

$$\frac{d}{dt}(uv) = u\frac{dv}{dt} + v\frac{du}{dt}$$

- Quotient rule

$$\frac{d}{dt}(u/v) = \frac{v\frac{du}{dt} - u\frac{dv}{dt}}{v^2}$$



- **Chain** rule

$$\frac{dy}{dx} = \frac{dy}{du} * \frac{du}{dx}$$

- The chain rule is used for calculating the derivatives of composite functions. The easiest way to recognize that you are dealing with a composite function is by the process of **elimination**:
- If none of the other rules apply, then you have a composite function.

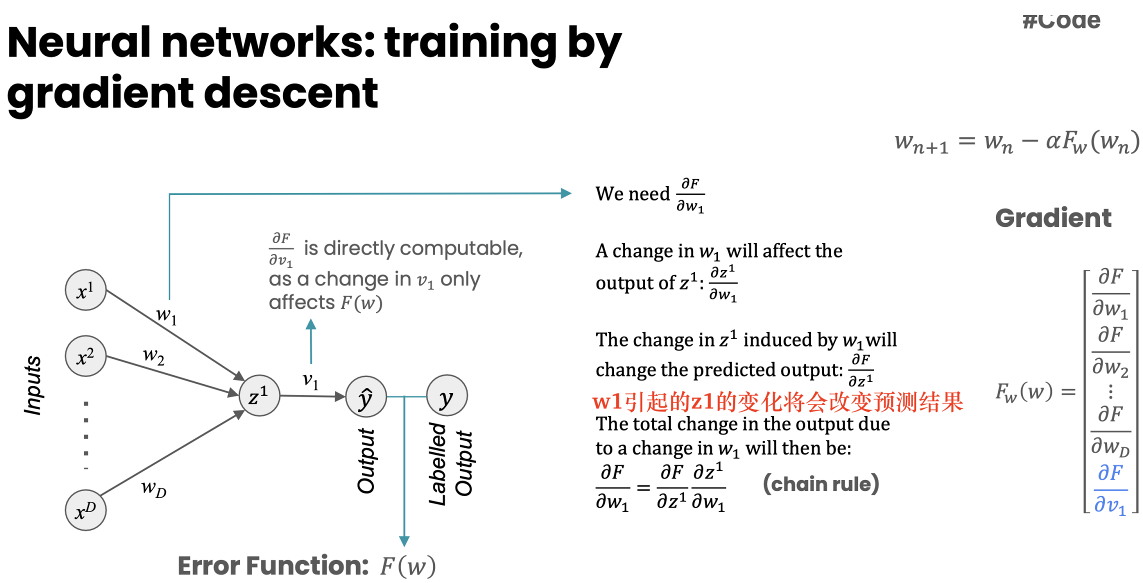
Overall Strategy of dealing with **derivatives**

1. Differentiate term by term. Deal with each term separately and, for each term, recognize any constant factor.
2. For each term, recognize whether it is one of the elementary functions (power, trigonometric, exponential or natural logarithm of the independent variable). If it is, you can easily apply the appropriate formula and you will be done. If it's not, go on to (3).
3. Decide whether the term is a product or a quotient. If it is, use the appropriate formula. Note that the appropriate formula will have you calculating two other derivatives and you will have to go back to (1) to deal with those. If it isn't, go to (4).
4. If you've gotten this far, you have to use the Chain Rule.

- Example:
  - $f(x) = x^2$,
  - $f'(x) = 2x$, the slope of a tangent at a particular point.
    - The rate of change: how much the value of y changes with respect to x
- In Gradient function
  - we compute the derivative of the **Loss function**, with respect to a particular weight
  - $\frac{dL}{dw_1}$, you assume everything else to be constant.
  - Needs review: Derivative $3x^3 - 3x + 5$ is $9x^2 - 3$

How to train deep neural networks

- TRAINING: Iterative procedure for minimization of an error function, with adjustments to the weights being made at each step.
  - STAGE 1: Evaluate derivatives of the error function wrt the weights
  - STAGE 2: Use derivatives to compute adjustments to be made to the weights (e.g., gradient descent)
- Training by gradient descent



  - Error function $F(w) = \sum_{i=1}^{N}(w^T x_i - y_i)^2$
  - Gradient $Fw(w) = 2\sum_{i-1}^{N}(w^T x_i - y_i)^2 \begin{bmatrix} x_i^1 \\ x_i^2 \\ \cdots \\ x_i^D \end{bmatrix}$
  - Update weight parameters $w_{n+1} = w^n - \alpha F_w(w_n)$

$$\frac{\partial F}{\partial w_1} = \frac{\partial F}{\partial z^1}\frac{\partial z^1}{\partial w_1}$$

- Analytical gradient expressions quickly become **intractable**

- We must **propagate gradients** from the output error F(w), all the way through each layer 传播梯度到每一层
- More sophisticated gradient calculation methods: backpropagation and automatic differentiation (AD)

- Chain rule of Calculus
  - Provides a means of differentiating nested functions.
  - Given two functions, $f(x)$ and $g(x)$, and the nested form $h = f(g(x))$
  - account for the actual order of the nesting relationship to correctly differentiate h

$$\frac{dh}{dx} = \frac{d}{dx} f(g(x)) = f'(g(x))g'(x)$$

- **Breakdown of the complete differentiation process:**
  1. **Inner function change:** The input value x changes, causing the inner box (i.e., g(x)) to move by a factor of g′(x).
  2. **Outer function impact:** This movement of the inner box affects the outer function f. The impact on f is determined by how sensitive f is to changes in its input, which is captured by f′(g(x)). In other words, f′(g(x)) tells us how much f will change for a small change in its input, which in this case is the output of the inner function g(x).
- **Combining the effects:**

  By multiplying g′(x) (rate of change of inner function) and f′(g(x)) (impact on outer function due to change in inner function's output), we capture the overall effect on the composite function f(g(x)) due to a change in the input x. This is why both g′(x) and f′(g(x)) are essential for the accurate differentiation of composite functions using the chain rule.

Automatic differentiation (AD)

- What is AD?
  - "meta-programming" approach to gradient calculation

- Obtains the gradients of the output **simultaneous** with the output of the network
- Software packages such as PyTorch, JAX largely avoid the need for any handcomputed gradients in this way

- Algebra of dual numbers
  - Keeps track of current computation's **value** u and its **derivative** u' as a pair: (u, u')
  - In dual number form, the general chain rule is

$$f((u,u')) = (f(u), f'(u)u')$$

  - Addition $f(u,v) = u + v$

$$(u,u') + (v,v') = (u+v, u'+v')$$

  - Multiplication $f(u,v) = uv$

$$(u,u') * (v,v') = (uv, u'v + v'u)$$

  - Maximum $f(u,v) = max(u,v)$

$$max((u,u'),(v,v')) = (max(u,v), u'\mathbb{I}[u > v] + v'\mathbb{I}]u \leq v])$$

  - ReLU activation $f(u) = relu(u)$

$$relu((u,u')) = (max(0,u), u'\mathbb{I}[u >= 0])$$

  - Constants $f(u) = c$

$$f((u,u')) = (c, 0)$$

  - Variable $f(u) = u$

$$f((u,u')) = (u, 1)$$

Here is an example of a chained calculation carried out using dual numbers. Given the constants $y = 3$ and $z = -1$ and variable $x = 2$, compute $u(x, y, z) = \max(yz, y + 2x)$ and its derivative, $u_x(x, y, z)$. Applying the rules above successively (and using additional symbols for intermediate computational results),

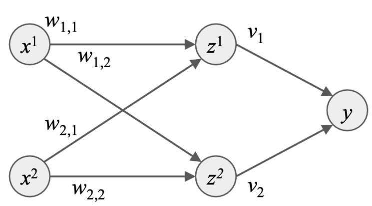
u_x: derivative of u with respect to x

$$
\begin{aligned}
\bar{x} &= (2,1) \\
\bar{y} &= (3,0) \\
\bar{z} &= (-1,0) \\
\text{let} \quad c &= (2,0) \\
c\bar{x} &= (2,0) \times (2,1) = (4,2) \\
r_1 &= \bar{y} \times \bar{z} = (3,0) \times (-1,0) = (-3,0) \\
r_2 &= \bar{y} + c\bar{x} = (3,0) + (4,2) = (7,2) \\
\bar{u} &= \max((-3,0),(7,2)) = (7,2),
\end{aligned}
\tag{14.7}
$$

therefore $u(x, y, z) = 7$ and $u_x(x, y, z) = 2$. While it is, of course, always possible to find the symbolic derivative of the function $u(x, y, z)$, AD enables entirely 'mechanical' calculational steps which lends itself to software implementation.

#Code

Automatic differentiation in action



derivate with respect to w_1,1

$$
\begin{aligned}
z^1 &= \text{relu}(w_{1,1}x^1 + w_{2,1}x^2) && w_{1,1} \to (w_{1,1}, 1),\ w_{2,1} \to (w_{2,1}, 0),\ x^1 \to (x^1, 0),\ x^2 \to (x^2, 0) \\
&= \text{relu}((w_{1,1}, 1) \times (x^1, 0) + (w_{2,1}, 0) \times (x^2, 0)) && (u, u') \times (v, v') = (uv, u'v + v'u) \\
&= \text{relu}((w_{1,1}x^1, x^1) + (w_{2,1}x^2, 0)) && (u, u') + (v, v') = (u + v, u' + v') \\
&= \text{relu}((w_{1,1}x^1 + w_{2,1}x^2, x^1)) && \text{relu}((u, u')) = (\max(0, u), u'\, 1[u \geq 0]) \\
&= (\max(0, w_{1,1}x^1 + w_{2,1}x^2), x^1\ 1[w_{1,1}x^1 + w_{2,1}x^2])
\end{aligned}
$$

Supporting Session

- tangent 切线