The fields of mathematics and their applications are constantly evolving. The world has moved towards a heavier use of AI automatic differentiation has found a major use. Automatic differentiation is used for deep machine learning and inside of neural networks. Gradient descent is one of the main methods used in deep machine learning. It is widely seen as a go to method for gradient descent problems.

Gradient descent starts off with a defined function and computes the error from the function when compared to the true values for the function. Partial derivatives are computed on the loss for the function to find gradients. The gradients are then adjusted to minimize the error leading to a learning rate. The gradients are then cleared and the method repeats until convergence is achieved. In more practical terms the method stops improving on its error. Automatic differentiation is useful in solving these gradients as the method takes known rules and applies it to the function. Reverse mode automatic differentiation specifically solves the function in one forward and backward pass.

The major problem for a fast learning rate is the computation of the gradients. Automatic differentiation serves as a superior method when compared to numeric differentiation or symbolic differentiation. Numeric differentiation either has truncation error or round off error due to how the general equation of a derivative works.

$$\frac{df}{dx} = \lim_{h \to 0} \ lim\frac{f(x+h)-f(x)}{h} \tag{1}$$

Decreasing h leads to a smaller truncation error but introduces the round-off error. The round off error occurs due to the limited memory of computers. Symbolic differentiation has the problem of expression swell. The method does not eliminate unnecessary computations or could run the same computation multiple times to solve one function.

Automatic differentiation uses knowledge of derivatives for addition, subtraction, multiplication, and division alongside the chain rule to quicken the process. Transcendental functions are also able to be used with automatic differentiation since their derivatives are known. These equations with known derivatives are considered elementary operations. An evaluation trace to define the function in one term and to compute the function and their derivative information. The input variables, intermediate variables, and the output variables are defined in the forward pass. Reverse mode automatic differentiation defines adjoints that serve as the partial derivatives or gradients for the function.

$$\bar{v}i = \frac{\partial yi}{\partial vj} \qquad (2)$$

Equation 2 serves as the general form of the adjoints. For reverse mode automatic differentiation only the information for the derivative is stored during the forward pass for the function. A backward pass is taken from the output variable that defines the entire function and uses elementary derivative rules, the chain rule, and any stored derivative information to calculate the adjoints.

Forward Primal Trace	**Output**	**Reverse Adjoint Trace**	**Output**
$v_{-1} = x_1$	3	$\bar{v}_{-1} = \bar{x}_1 = \bar{v}_2 \cdot (1 / v_{-1}) + \bar{v}_1 \cdot v_0$	$-1 \cdot (1 / 3) + 1 \cdot -4 = -4.33$
$v_0 = x_2$	-4	$\bar{v}_0 = \bar{x}_2 = \bar{v}_3 \cdot 1 + \bar{v}_1 \cdot v_{-1}$	$1 \cdot 1 + 1 \cdot 3 = 4$
$v_1 = v_{-1}v_0$	$3 \cdot -4 = -12$	$\bar{v}_1 = \bar{v}_3 \cdot 1$	$1 \cdot 1 = 1$
$v_2 = \ln(v_{-1})$	ln(3) = 1.10	$\bar{v}_2 = \bar{v}_4 \cdot -1$	$1 \cdot -1 = -1$
$v_3 = v_1 + v_0$	-12 + -4 = -16	$\bar{v}_3 = \bar{v}_4 \cdot 1$	$1 \cdot 1 = 1$
$v_4 = v_3 - v_2$	-16 - 1.10 = -17.10	$\bar{v}_4 = \bar{y}$	1
$y = v_4$	-17.10	$\bar{y}$	1

For the forward pass the input of 3 for x_1 and the input of 4 for x_2 is used. The forward pass condenses the function into individual parts until the entire function is defined as one part. The output uses the inputs of x_1 and x_2 to compute the outputs of the individual parts of the equation. For example $x_1 = v_{-1}$ with the output of 3 and x_2=

$v_0$with the output of 4. v_1 is $v_{-1} * v_0$ with the output of $4 * 3 = 12$ being computed. This process is repeated all the way down to $y = -17.10$ as the function output.

In the function the final variable ends up as y. Using Equation 2 the partial derivative is $\frac{\partial y}{\partial y} = 1$ which leads to the output table on the right side. The adjoint is computed looking into what informs the part which is being examined. For the table above at v_2 occurs only as part of v_4 so it only informs that part of the table. $\bar{v}_4$ is used alongside the derivative v_2 or the part of the function which we are examining. For v_0 and v_{-1}, they are part of two parts of the equation and both parts must be accounted for when solving for the output of the reverse pass. In a broader stroke the reverse pass uses the chain rule and the derivative information, both known and gathered in the forward pass, to work from the output through each part of the function back to the inputs. The beauty of reverse mode automatic differentiation is that only one pass is necessary to achieve the outputs of the derivatives of each input variable.

Reverse mode automatic differentiation can be used in Jacobian Matrices. When vector valued functions $R^n \rightarrow R^m$ and $a \in R^n$, assigning $x = a$ and setting $\bar{y} = e_j$ for j = 1, …, m each reverse pass generates the jth partial derivative output. A vector jacobian process can be found with $r^T \in R^{1*m}$ and $J \in R^{m*n}$ The matrix created by this process can be used for optimization of the neural networks through the partial derivatives.

The cost is only the number of steps required to work through the backward pass. One reverse pass is necessary to one step of gradient descent which is linear. The only cost is the extra memory to store the necessary derivative information for the adjoints. Reverse mode automatic differentiation works best when n >> m since the linear factor comes from the m passes required for the m rows.

Reverse mode automatic differentiation related to the taking of derivatives from the fourth chapter in the textbook. The method looks at the modern field of deep learning within AI and uses a far superior method than the numeric differentiation that is covered in the book. It uses partial derivatives to solve gradient descent which is currently widely regarded as the go to method.

The coding part of reverse mode automatic differentiation in the github uses operator overloading in python. This process redefines the elementary operations to include the derivative data to allow the automatic differentiation to work. For the reverse mode to work a variable is defined and then a closure function is created. The closure function accepts the adjoint from the variable and adds it to the adjoint from the object. Then it computes the partial derivative of the output to each operator’s derivative and the adjoint to implement the chain rule. Then it goes backward on each input to continue the pass. Finally it returns the resultant variable from the beginning of the process.

Sources

https://huggingface.co/blog/andmholm/what-is-automatic-differentiation

https://mostafa-samir.github.io/auto-diff-pt1/

https://mostafa-samir.github.io/auto-diff-pt2/