What is backpropagation?

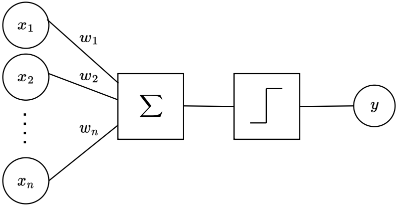
Shpresim Sadiku

(Technische Universität Berlin & Zuse Institute Berlin)

What is ...? Seminar · February 10, 2023

The Perceptron

Structure of the perceptron:



- A weighted sum of the input features

$$z = \sum_{i=1}^{n} w_i x_i + b = \mathbf{w}^T \mathbf{x} + b$$

- Followed by the activation function

$$y = \sigma(z)$$

Problem formulation: Let

$$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(m)} \in \mathbb{R}^n$$

be our input data and $t^{(1)}, t^{(2)}, ..., t^{(m)} \in \{-1, 1\}$ our corresponding labels (aka. targets). The goal of the perceptron is to learn a collection of parameters $(\mathbf{w}, b)$ such that all points are correctly classified, i.e.

$$\forall_{k=1}^{m} : y^{(k)} = t^{(k)}$$

The Perceptron Algorithm

For each datapoint, predictions of our perceptron are computed as

$$
\begin{aligned}
z^{(k)} &= \mathbf{w}^T \mathbf{x}^{(k)} + b \\
y^{(k)} &= \sigma(z^{(k)})
\end{aligned}
$$

Perceptron algorithm

- Iterate (multiple times from $k = 1, ..., m$)
  - If $\mathbf{x}^{(k)}$ is correctly classified ($y^{(k)} = t^{(k)}$), continue.
  - If $\mathbf{x}^{(k)}$ is wrongly classified ($y^{(k)}) \neq t^{(k)}$, update the perception:

$$
\begin{aligned}
\mathbf{w} &\leftarrow \mathbf{w} + \eta \cdot \mathbf{x}^{(k)} t^{(k)} \\
b &\leftarrow b + \eta \cdot t^{(k)}
\end{aligned}
$$

  where η is a learning rate.
- Stop once all examples are correctly classified.

The Perceptron: Optimization View

Proposition

The perceptron can be seen as a gradient descent of the error function

$$\mathcal{E}(\mathbf{w}, b) = \frac{1}{N}\sum_{k=1}^{m} \underbrace{\max(0, -z^{(k)}t^{(k)})}_{\mathcal{E}_k(\mathbf{w}, b)}$$

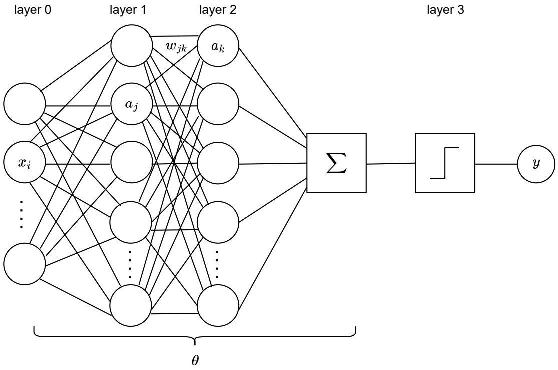
Also known as the Hinge Loss.

Proof.

$$\begin{aligned}
\mathbf{w} - \eta \frac{\partial \mathcal{E}_k}{\partial \mathbf{w}} &= \mathbf{w} - \eta \cdot 1_{-z^{(k)}t^{(k)}>0} \cdot \left(-\frac{\partial z^{(k)}}{\partial \mathbf{w}} t^{(k)}\right) \\
&= \mathbf{w} - \eta \cdot 1_{y^{(k)} \neq t^{(k)}} \cdot \left(-\frac{\partial z^{(k)}}{\partial \mathbf{w}} t^{(k)}\right) \\
&= \mathbf{w} + \eta \cdot 1_{y^{(k)} \neq t^{(k)}} \cdot \mathbf{x}^k t^{(k)}
\end{aligned}$$

which is the parameter update equation of the perceptron algorithm. we proceed similarly for the parameter b.

From Perceptron to Deep Neural Networks



Idea:

Generalize the formulation where z is not the output of the perceptron, but of any multilayer neural network with parameters θ.

$\hookrightarrow$ Updated error function $\mathcal{E}(\theta)$.

Numerical Differentiation

Question:

How hard is it to compute the gradient of the newly defined error function w.r.t. the model parameters?

$$\theta = \theta - \eta \frac{\partial \mathcal{E}}{\partial \theta}$$

Formula for numerical differentiation:

$$\forall_t : \frac{\partial \mathcal{E}}{\partial \theta_t} = \lim_{\varepsilon \to 0} \frac{\mathcal{E}(\theta + \varepsilon \cdot \delta_t) - \mathcal{E}(\theta)}{\varepsilon}$$

where δ_t is an indicator vector for the parameter t.

Properties:

- Can be applied to any error function $\mathcal{E}$ (not necessary the error of a neural network).
- Need to evaluate the function as many times as there are parameters ($\rightarrow$ slow when the number of parameters is large).
- A neural network typically has between 10^3 and 10^9 parameters (numerical differentiation unfeasible).
- Still useful as a unit test for verifying gradient computation.
- Because ε and the numerator are very small, for numerical differentiation to work, one must use high-precision (e.g. float64 rather than float32).

The Chain Rule

Suppose that some parameter of interest θ_q (one element of the parameter vector θ) is linked to the output of the network through some sequence of functions.

$$\boldsymbol{\theta_q} \longrightarrow \boldsymbol{a} \longrightarrow \boldsymbol{b} \longrightarrow \boldsymbol{z}$$

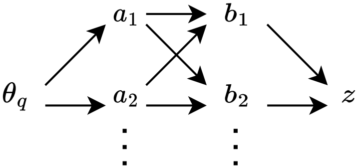
The chain rule for derivatives states that

$$\frac{\partial z}{\partial \theta_q} = \frac{\partial a}{\partial \theta_q} \frac{\partial b}{\partial a} \frac{\partial z}{\partial b}$$

i.e., the derivative w.r.t. the parameter of interest is the product of local derivatices along the path connecting θ_q to z.

The Multivariate Chain Rule

In practice, some parameter of interest may be linked to the output of the network through multiple path (formed by all neurons in layers between θ_q and z:)

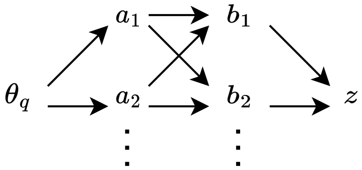


The chain rule can be extended to this multivariate scenario by enumerating all the paths between θ_q and z

$$\frac{\partial z}{\partial \theta_q} = \sum_i \sum_j \frac{\partial a_i}{\partial \theta_q} \frac{\partial b_j}{\partial a_j} \frac{\partial z}{\partial b_j}$$

where $\sum_i$ and $\sum_j$ run over all indices of the nodes in the corresponding layers. This is a nested sum (its complexity grows exponentially with the number of layers).

Factor Structure in the Multivariate Chain Rule

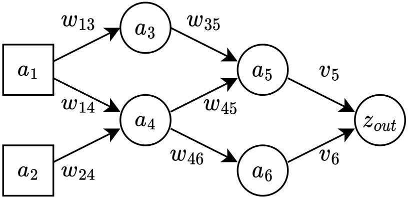


- Computation can be rewritten in a way that summing operation can be perfomed incrementally.
- Intermediate computation can be reused for different paths, and for different parameters for which we would like to compute the gradient.

$$\frac{\partial z}{\partial \theta_q} = \sum_i \frac{\partial a_i}{\partial \theta_q} \underbrace{\sum_j \frac{\partial b_j}{\partial a_j} \underbrace{\frac{\partial z}{\partial b_j}}_{\delta_j}}_{\delta_i}$$

- Overall, the resulting gradient computation (w.r.t. all parameters in the network) becomes linear with the size of the network (⇒ fast!)
- The algorithm is known as the Error Backpropagation algorithm (Rumehart 1986).

Worked through Example

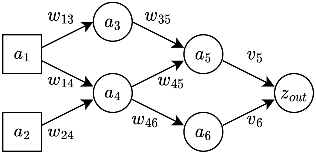


Forward pass:

$$
\begin{aligned}
z_3 &= a_1 w_{13} \\
z_4 &= a_1 w_{14} + a_2 w_{24} \\
z_5 &= a_3 w_{35} + a_4 w_{45} \\
z_6 &= a_4 w_{46} \\
z_{out} &= a_5 v_5 + a_6 v_6 \\
\mathcal{E} &= \max(0, -z_{out} \cdot t)
\end{aligned}
$$

$$
\begin{aligned}
a_1 &= x_1 \\
a_2 &= x_2 \\
a_3 &= \tanh(z_3) \\
a_4 &= \tanh(z_4) \\
a_5 &= \tanh(z_5) \\
a_6 &= \tanh(z_6)
\end{aligned}
$$

Worked through Example



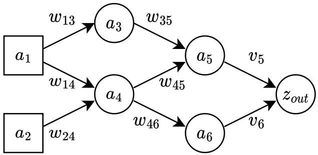
$$
\begin{aligned}
z_3 &= a_1 w_{13} \\
z_4 &= a_1 w_{14} + a_2 w_{24} \\
z_5 &= a_3 w_{35} + a_4 w_{45} \\
z_6 &= a_4 w_{46} \\
z_{out} &= a_5 v_5 + a_6 v_6 \\
\mathcal{E} &= \max(0, -z_{out} \cdot t)
\end{aligned}
$$

$$
\begin{aligned}
a_1 &= x_1 \\
a_2 &= x_2 \\
a_3 &= \tanh(z_3) \\
a_4 &= \tanh(z_4) \\
a_5 &= \tanh(z_5) \\
a_6 &= \tanh(z_6)
\end{aligned}
$$

Backward pass:

$$
\begin{aligned}
\delta_{out} \quad = \quad & \frac{\partial \mathcal{E}}{\partial z_{out}} = 1_{\{-z_{out} \cdot t > 0\}} \cdot (-t) \\
& \frac{\partial \mathcal{E}}{\partial v_6} = \frac{\partial z_{out}}{\partial v_6} \frac{\partial \mathcal{E}}{\partial z_{out}} = a_6 \cdot \delta_{out} \\
& \frac{\partial \mathcal{E}}{\partial v_5} = \frac{\partial z_{out}}{\partial v_5} \frac{\partial \mathcal{E}}{\partial z_{out}} = a_5 \cdot \delta_{out}
\end{aligned}
$$

Worked through Example



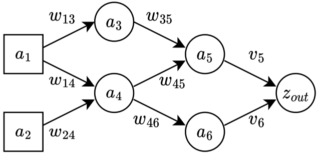
$$
\begin{aligned}
z_3 &= a_1 w_{13} \\
z_4 &= a_1 w_{14} + a_2 w_{24} \\
z_5 &= a_3 w_{35} + a_4 w_{45} \\
z_6 &= a_4 w_{46} \\
z_{out} &= a_5 v_5 + a_6 v_6 \\
\mathcal{E} &= \max(0, -z_{out} \cdot t)
\end{aligned}
$$

$$
\begin{aligned}
a_1 &= x_1 \\
a_2 &= x_2 \\
a_3 &= \tanh(z_3) \\
a_4 &= \tanh(z_4) \\
a_5 &= \tanh(z_5) \\
a_6 &= \tanh(z_6)
\end{aligned}
$$

Backward pass:

$$
\begin{aligned}
\delta_{out} &= \frac{\partial \mathcal{E}}{\partial z_{out}} = 1_{\{-z_{out} \cdot t > 0\}} \cdot (-t) \\
\delta_6 &= \frac{\partial \mathcal{E}}{\partial a_6} = \frac{\partial z_{out}}{\partial a_6} \frac{\partial \mathcal{E}}{\partial z_{out}} = v_6 \cdot \delta_{out} \\
\delta_5 &= \frac{\partial \mathcal{E}}{\partial a_5} = \frac{\partial z_{out}}{\partial a_5} \frac{\partial \mathcal{E}}{\partial z_{out}} = v_5 \cdot \delta_{out}
\end{aligned}
$$

Worked through Example



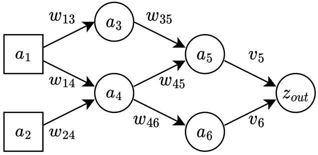
$$
\begin{aligned}
z_3 &= a_1 w_{13} \\
z_4 &= a_1 w_{14} + a_2 w_{24} \\
z_5 &= a_3 w_{35} + a_4 w_{45} \\
z_6 &= a_4 w_{46} \\
z_{out} &= a_5 v_5 + a_6 v_6 \\
\mathcal{E} &= \max(0, -z_{out} \cdot t)
\end{aligned}
$$

$$
\begin{aligned}
a_1 &= x_1 \\
a_2 &= x_2 \\
a_3 &= \tanh(z_3) \\
a_4 &= \tanh(z_4) \\
a_5 &= \tanh(z_5) \\
a_6 &= \tanh(z_6)
\end{aligned}
$$

Backward pass:

$$
\begin{aligned}
\delta_6 &= \frac{\partial \mathcal{E}}{\partial a_6} = \frac{\partial z_{out}}{\partial a_6} \frac{\partial \mathcal{E}}{\partial z_{out}} = v_6 \cdot \delta_{out} \\
\delta_5 &= \frac{\partial \mathcal{E}}{\partial a_5} = \frac{\partial z_{out}}{\partial a_5} \frac{\partial \mathcal{E}}{\partial z_{out}} = v_5 \cdot \delta_{out} \\
&\frac{\partial \mathcal{E}}{\partial w_{46}} = \frac{\partial z_6}{\partial w_{46}} \frac{\partial a_6}{\partial z_6} \frac{\partial \mathcal{E}}{\partial a_6} = a_4 \cdot \tanh'(z_6) \cdot \delta_6 \\
&\frac{\partial \mathcal{E}}{\partial w_{45}} = \frac{\partial z_5}{\partial w_{45}} \frac{\partial a_5}{\partial z_5} \frac{\partial \mathcal{E}}{\partial a_5} = a_4 \cdot \tanh'(z_5) \cdot \delta_5 \\
&\frac{\partial \mathcal{E}}{\partial w_{35}} = \frac{\partial z_5}{\partial w_{35}} \frac{\partial a_5}{\partial z_5} \frac{\partial \mathcal{E}}{\partial a_3} = a_5 \cdot \tanh'(z_5) \cdot \delta_5
\end{aligned}
$$

Worked through Example



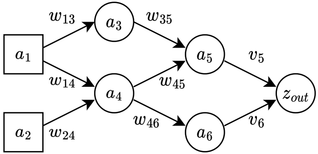
$$\begin{aligned} z_3 &= a_1 w_{13} \\ z_4 &= a_1 w_{14} + a_2 w_{24} \\ z_5 &= a_3 w_{35} + a_4 w_{45} \\ z_6 &= a_4 w_{46} \\ z_{out} &= a_5 v_5 + a_6 v_6 \\ \mathcal{E} &= \max(0, -z_{out} \cdot t) \end{aligned}$$

$$\begin{aligned} a_1 &= x_1 \\ a_2 &= x_2 \\ a_3 &= \tanh(z_3) \\ a_4 &= \tanh(z_4) \\ a_5 &= \tanh(z_5) \\ a_6 &= \tanh(z_6) \end{aligned}$$

Backward pass:

$$\begin{aligned} \delta_6 &= \frac{\partial \mathcal{E}}{\partial a_6} = \frac{\partial z_{out}}{\partial a_6}\frac{\partial \mathcal{E}}{\partial z_{out}} = v_6 \cdot \delta_{out} \\ \delta_5 &= \frac{\partial \mathcal{E}}{\partial a_5} = \frac{\partial z_{out}}{\partial a_5}\frac{\partial \mathcal{E}}{\partial z_{out}} = v_5 \cdot \delta_{out} \\ \delta_4 &= \frac{\partial \mathcal{E}}{\partial a_4} = \frac{\partial z_6}{\partial a_4}\frac{\partial a_6}{\partial z_6}\frac{\partial \mathcal{E}}{\partial a_6} + \frac{\partial z_5}{\partial a_4}\frac{\partial a_5}{\partial z_5}\frac{\partial \mathcal{E}}{\partial a_5} = w_{46} \cdot \tanh'(z_6) \cdot \delta_6 + w_{45} \cdot \tanh'(z_5) \cdot \delta_5 \\ \delta_3 &= \frac{\partial \mathcal{E}}{\partial a_3} = \frac{\partial z_5}{\partial a_3}\frac{\partial a_5}{\partial z_5}\frac{\partial \mathcal{E}}{\partial a_5} = w_{35} \cdot \tanh'(z_5) \cdot \delta_5 \end{aligned}$$

Worked through Example



$$
\begin{aligned}
z_3 &= a_1 w_{13} \\
z_4 &= a_1 w_{14} + a_2 w_{24} \\
z_5 &= a_3 w_{35} + a_4 w_{45} \\
z_6 &= a_4 w_{46} \\
z_{out} &= a_5 v_5 + a_6 v_6 \\
\mathcal{E} &= \max(0, -z_{out} \cdot t)
\end{aligned}
$$

$$
\begin{aligned}
a_1 &= x_1 \\
a_2 &= x_2 \\
a_3 &= \tanh(z_3) \\
a_4 &= \tanh(z_4) \\
a_5 &= \tanh(z_5) \\
a_6 &= \tanh(z_6)
\end{aligned}
$$

Backward pass:

$$
\begin{aligned}
\delta_4 &= \frac{\partial \mathcal{E}}{\partial a_4} = \frac{\partial z_6}{\partial a_4}\frac{\partial a_6}{\partial z_6}\frac{\partial \mathcal{E}}{\partial a_6} + \frac{\partial z_5}{\partial a_4}\frac{\partial a_5}{\partial z_5}\frac{\partial \mathcal{E}}{\partial a_5} = w_{46} \cdot \tanh'(z_6) \cdot \delta_6 + w_{45} \cdot \tanh'(z_5) \cdot \delta_5 \\
\delta_3 &= \frac{\partial \mathcal{E}}{\partial a_3} = \frac{\partial z_5}{\partial a_3}\frac{\partial a_5}{\partial z_5}\frac{\partial \mathcal{E}}{\partial a_5} = w_{35} \cdot \tanh'(z_5) \cdot \delta_5 \\
& \frac{\partial \mathcal{E}}{\partial w_{24}} = \frac{\partial z_4}{\partial w_{24}}\frac{\partial a_4}{\partial z_4}\frac{\partial \mathcal{E}}{\partial a_4} = a_2 \cdot \tanh'(z_4) \cdot \delta_4 \\
& \frac{\partial \mathcal{E}}{\partial w_{14}} = \frac{\partial z_4}{\partial w_{14}}\frac{\partial a_4}{\partial z_4}\frac{\partial \mathcal{E}}{\partial a_4} = a_1 \cdot \tanh'(z_4) \cdot \delta_4 \\
& \frac{\partial \mathcal{E}}{\partial w_{13}} = \frac{\partial z_3}{\partial w_{13}}\frac{\partial a_3}{\partial z_3}\frac{\partial \mathcal{E}}{\partial a_3} = a_1 \cdot \tanh'(z_3) \cdot \delta_3
\end{aligned}
$$

Formalization for a Standard Neural Network

The gradient of the error can be propagated from layer to layer using the chain rule:

$$\underbrace{\frac{\partial \mathcal{E}}{\partial a_j}}_{\delta_j} = \sum_k \underbrace{\frac{\partial a_k}{\partial a_j}}_{w_{jk} g'(z_k)} \cdot \underbrace{\frac{\partial \mathcal{E}}{\partial a_k}}_{\delta_k}$$

And gradients w.r.t. parameters at each layer can be extracted as:

$$\frac{\partial \mathcal{E}}{\partial w_{jk}} = \sum_k \underbrace{\frac{\partial a_k}{\partial w_{jk}}}_{a_j g'(z_k)} \cdot \underbrace{\frac{\partial \mathcal{E}}{\partial a_k}}_{\delta_k}$$

which can be written as matrix-vector products

$$\begin{aligned} \delta^{(l-1)} &= W^{(l-1,l)} \cdot (g'(\mathbf{z}^{(l)}) \odot \delta^{(l)}) \\ \frac{\partial \mathcal{E}}{\partial W^{(l-1,l)}} &= \mathbf{a} \cdot (g'(\mathbf{z}^{(l)}) \odot \delta^{(l)})^T \end{aligned}$$

Choice of Nonlinear Activation Function

In practice, for training to proceed, the nonlinear function must be chosen in a way that:

- Its gradient is defined (almost) everywhere.
- There is a significant portion of the input domain where the gradient is non-zero.
- Gradient must be informative, i.e. indicate decrease/increase of the activation function.

Example of commonly used activation functions:

- $g(z) = \exp(z)/(1+\exp(z))$
- $g(z) = \tanh(z)$
- $g(z) = \max(0, z)$

Example of problematic activation functions:

- $g(z) = \max(0, z-100)$
- $z(z) = 1_{z>0}$
- $g(z) = \sin(100 \cdot z)$

Automatic Differentiation

- Automatically generate backpropagation equations from the forward equations
- Automatic differentiation became widely available in neural network libraries (PyTorch, Tensorflow, JAX, etc.)

Consequences:

- In practice, we do not need to do backpropagation anymore. We just need to program the forward pass, and the backward pass comes for free.
- This has enabled researchers to develop neural networks that are way more complex, and with much more heterogeneous structures (e.g. ResNet, Yolo, transformers, etc.)
- Only in few cases, it is still useful to express the gradient analytically (e.g. to analyze theoretically the stability of a gradient descent procedure, such as the vanishing/exploding gradients problem in recurrent neural networks).

Simple algorithm for training a neural network

Basic gradient descent algorithm:

- Initialize vector of parameters θ at random.
- Iterate for T steps:
  - Compute the forward pass
  - Extract the gradient $\partial\mathcal{E}/\partial\theta$ using backpropagation
  - Perform a gradient step

    $$\theta = \theta - \gamma \partial\mathcal{E}/\partial\theta$$

    where γ is a learning rate that needs to be set by the user.

Summary

- Error of a classifier can be minimized using gradient descent (e.g. Perceptron, neural network + backpropagation).
- Error backpropagation is computationally efficient way of computing the gradient (much faster than using the limit formulation of the derivative).
- Error backpropagation is an application of the multivariate chain rule, where the different terms can be factored due to the structure of the neural network graph.
- In practice, we most of the time do not need to program error backpropagation manually, and we can instead use automatic differentiation techniques available in most modern neural network libraries.

THANK YOU!

Slides available at:

`www.shpresimsadiku.com`

Check related information on Twitter at:

`@shpresimsadiku`