

SDZ

Marko Ruman

supervisor: Ing. Tatiana Valentine Guy, Ph.D.

Outline

- 1 Neural network definition
- 2 Backpropagation Algorithm
- 3 CNNs

Definition of fully connected NN

A fully connected neural network is a tuple

$$(\mathbf{G}, \Phi, \Theta_0, \mathcal{L})$$

- \mathbf{G} is a directed graph defining the architecture of the network, $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$, where
 - $\mathbf{V} = \left\{ \{n_i^k\}_{i=1}^{r_k}, k \in \{1, 2, \dots, m\} \right\}$ is a set of neurons, the set $\{n_i^k\}_{i=1}^{r_k}$ is k -th layer,
 - m is the number of layers,
 - r_k is the number of neurons in the k -th layer,
 - \mathbf{E} is the set of edges, $\mathbf{E} = \left\{ \left(n_i^k, n_j^{k+1} \right), i \in \{1, 2, \dots, r_k\}, j \in \{1, 2, \dots, r_{k+1}\}, k \in \{1, 2, \dots, m\} \right\}$ and
 - to each edge of the network (n_i^k, n_j^{k+1}) , there is a corresponding weight $\theta_{ij}^k \in \mathbb{R}$
 - the weights can be organized into matrices Θ^k , where θ_{ij}^k is its (i,j) -th entry, $\Theta = \{\Theta^k\}_{k=1}^m$

Definition of fully connected NN

A fully connected neural network is a tuple

$$(\mathbf{G}, \Phi, \Theta_0, \mathcal{L})$$

- Φ is a set of activation functions, $\Phi = \{\phi^k\}_{k=1}^m$, where $\phi^k : \mathbb{R} \mapsto \mathbb{R}$ is activation function of layer k
- Θ_0 is an initial value of weights, $\Theta_0 = \{\Theta_0^k\}_{k=1}^m$

Definition of fully connected NN

A fully connected neural network is a tuple

$$(\mathbf{G}, \Phi, \Theta_0, \mathcal{L})$$

- $\mathcal{L} = \mathcal{L}(\mathbf{X}, \Theta)$ is the loss function, that has the following form

$$\mathcal{L}(\mathbf{X}, \Theta) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}_p(x_i, y_i, \Theta) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}_i, \quad (1)$$

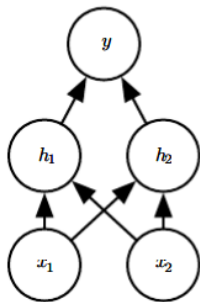
where:

- \mathcal{L}_p is chosen usually as squared L2 norm or L1 norm,
- $\mathbf{X} = \{(x_i, y_i)\}_{i=1}^N$ is *dataset*, i.e. set of input-output pairs of the size $N \in \mathbb{N}$,

The functional form that NN constructs

$$f(x|\Theta) = \phi^m (\Theta^m \phi^{m-1} (\Theta^{m-1} \dots \phi^1 (\Theta^1 x) \dots))$$

NN example



- $m = 2, r_1 = 2, r_2 = 1$
- $\phi^1(x) = \text{ReLU}(x) = \max(0, x)$
- $\phi^2(x) = x$

$$f(x|\Theta) = \Theta_2 \max(0, \Theta_1 x)$$

Backpropagation algorithm

Gradient descent

$$\Theta_{t+1} = \Theta_t - \alpha \frac{\partial \mathcal{L}(\mathbf{X}, \Theta_t)}{\partial \Theta}, \quad \text{i.e.} \quad \theta_{t+1;ij}^k = \theta_{t;ij}^k - \alpha \frac{1}{N} \sum_{l=1}^N \frac{\partial \mathcal{L}_l}{\partial \theta_{ij}^k}$$

Backpropagation formula

$$\frac{\partial \mathcal{L}_i}{\partial \theta_{ij}^k} = \delta_i^k o_j^{k-1} = \frac{d}{de_i^k} \phi^k(e_i^k) o_j^{k-1} \sum_{l=1}^{r_k} \delta_l^{k+1} \theta_{li}^k$$

- $e^k = \Theta^{k-1} o^{k-1}$ *activation of k-th layer*
- $o^k = \phi^k(e^k)$ *output of k-th layer*
- $\delta_i^k = \frac{\partial \mathcal{L}_i}{\partial e_i^k}$

Backpropagation algorithm

Backpropagation algorithm

- 1 Calculate the forward phase for each input-output pair (x_i, y_i) from a mini-batch and store the results $f(x_i)$ ($??$), e_j^k and o_j^k by proceeding from layer $k = 0$, the input layer, to layer $k = m$, the output layer.
- 2 Calculate the backward phase for each input-output pair (x_i, y_i) from a mini-batch and store the results $\frac{\partial \mathcal{L}_i}{\partial \theta_{ij}^k}$.
 - Evaluate the loss term for the final layer $\delta_j^m = \frac{\partial \mathcal{L}_i}{\partial e_j^m}$ for each node.
 - Backpropagate the loss terms for the other layers δ_j^k , working backwards from the layer $k = m - 1$.
 - Evaluate the partial derivatives of the individual loss $\frac{\partial \mathcal{L}_i}{\partial \theta_{ij}^k}$.
- 3 Update the weights by gradient descent.

Convolutional NN

$$[\Theta * x]_{i_1, i_2, \dots, i_n} = \sum_{j_1} \sum_{j_2} \dots \sum_{j_n} x_{i_1 - j_1, i_2 - j_2, \dots, i_n - j_n} \Theta_{j_1, j_2, \dots, j_n}$$

Convolution layer

A 2-D convolutional layer *input channels* C_{in} , *output channels* C_{out} and *kernel size* k contains:

- C_{out} parameter matrices $\Theta_1, \dots, \Theta_{C_{out}}$. Each $\Theta_i \in \mathbb{R}^{k, k, C_{in}}$.
- activation function ϕ .

It takes $x \in \mathbb{R}^{C_{in}, n, n}$ as an input and gives the following output o :

- $e = [\Theta_1 * x, \Theta_2 * x, \dots, \Theta_{C_{out}} * x] \in \mathbb{R}^{C_{out}, n_{out}, n_{out}}$ is the *activation*,
- $o = \phi(e)$ is *output* of the layer.