Exercise.L.MLE.SS24. Exercise 02_ Basic neural networks

Samstag, 25. Mai 2024 14:19

Exercise.L....
02_ Basic...

Exercise 2

Deadline: 22.05.2023, 4:00 pm

This exercise focuses on basics of neural networks.

Regulations

Please create three files for your solution: `network.pdf` for your answers to the non-programming tasks (e.g. a scan of your hand-written solution), a jupyter notebook `network.ipynb` for the completed neural network code and the same file exported as a `.html` file (`network.html`). Zip all files into a single archive `ex02.zip` and upload this file to MaMPF before the given deadline.
Moreover, please set your **Anzeigename**/**display name** and **Name in Uebungsgruppen**/**name in tutorials** in MaMPF to your real name, which should be identical to your name in `muesli` and make sure you **join the submission** of your team via the invitation code before the submission deadline. Check out `https://mampf.blog/handing-in-homework-assignments` for instructions.

A Neural Playground

Have a look at `http://playground.tensorflow.org/` and play around with it for a while to get some feeling for neural networks.

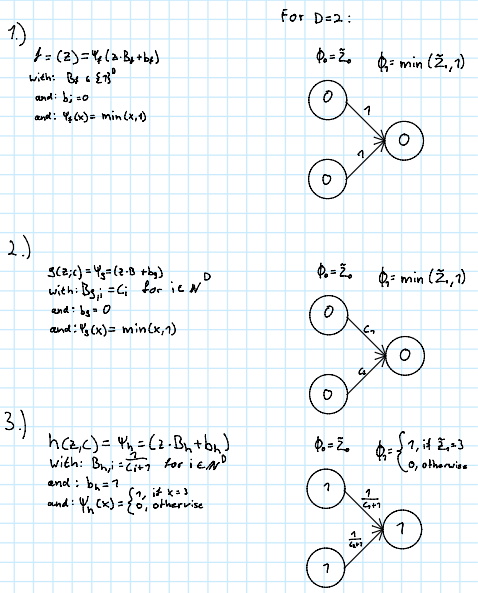
This is not an official exercise, so you don't need to hand in anything.

1 Hand-Crafted Network (15 Points)

In this exercise we want to construct neural networks that classify an arbitrary training set with zero training error. As preparation, design single neurons (i.e. specify their weights, bias, and activation function) for the following tasks:

1. logical OR: map a binary input vector $z \in \{0,1\}^D$ to
$$z \to f(z) = \begin{cases} 1 & \exists j \text{ such that } z_j = 1 \\ 0 & \text{otherwise, i.e. } z = 0 \end{cases}$$
2. masked logical OR: for an arbitrary but fixed binary vector $c \in \{0,1\}^D$ map the input vector $z \in \{0,1\}^D$ to
$$z \to g(z;c) = \begin{cases} 1 & \exists j \text{ such that } c_j = 1 \text{ and } z_j = 1 \\ 0 & \text{otherwise, i.e. } z = 0 \text{ or } z_j = 1 \text{ occurs only at indices where } c_j = 0 \end{cases}$$
3. perfect match: for an arbitrary but fixed binary vector $c \in \{0,1\}^D$ map the input vector $z \in \{0,1\}^D$ to
$$z \to h(z;c) = \begin{cases} 1 & z = c \\ 0 & \text{otherwise} \end{cases}$$

Now use these building blocks to design a three-layer network for the dataset X, Y in Figure 1 (with 2-dimensional feature vectors X_i and class labels $Y_i \in$ {"red minus","blue plus","green circle"}). The first layer maps every X_i onto one **corner** of a hypercube $\{0,1\}^M$ such that each corner contains only points of the same class (you can map one class to multiple corners, but not multiple



$M=3 \Rightarrow$ 3 decision boundaries $\Rightarrow 2^3 = 8$ corners

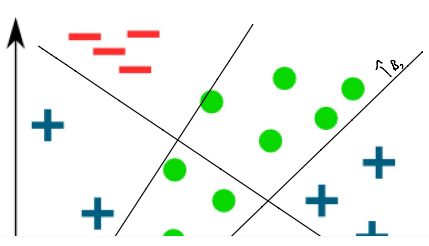
Is enough to divide the 3 classes without training error

classes:
red := 1 => 1 0 0
blue := 2 => 0 1 0
green := 3 => 0 0 1

(0,0,0) -> (1,0,0) -> 2
(0,0,1) -> (1,0,1) -> 1
(0,1,0) -> 1 (1,1,0) -> 3
(0,1,1) -> 2 (1,1,1) -> 3

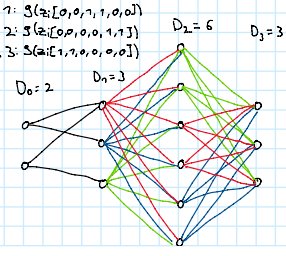
So the decision boundaries classify the data into 8 regions, 6 of them correspond to a class.

Machine Learning Essentials Ullrich Köthe
Summer Semester 2023 **Exercise 2** ullrich.koethe@iwr.uni-heidelberg.de



$N2.1 : h(z;[1,1,0])$ } green
$N2.2 : h(z;[1,1,1])$ }
$N2.3 : h(z;[0,1,0])$ } red
$N2.4 : h(z;[1,0,1])$ }
$N2.5 : h(z;[0,1,1])$ } blue
$N2.6 : h(z;[1,0,0])$ }

$N3.1 : S(z;[0,0,1,1,0,0])$
$N3.2 : S(z;[0,0,0,0,1,1])$
$N3.3 : S(z;[1,1,0,0,0,0])$



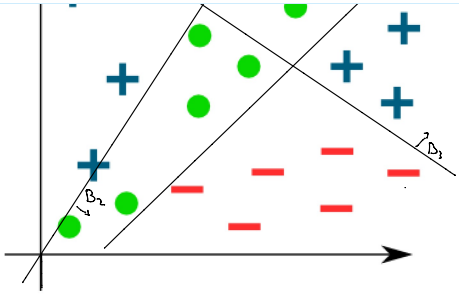


Figure 1: Example dataset with three classes (red minus, blue plus, green circle). Sketch the decision boundaries of your network here and indicate hypercube corners as described in the text.

classes onto one corner). The dimension M of the hypercube is a hyperparameter that you have to adjust for the given training set.

Copy figure 1 to your solution and draw the first layer's decision boundaries and their normal vectors. You do not need to specify precise equations for these boundaries. The normal vectors should point in the direction of positive pre-activation. Indicate for each decision region to which hypercube corner the corresponding feature points will be mapped.

Now specify the equations for the network's second and third (=output) layer to produce a one-hot encoding of the class labels with zero training error. A one-hot encoding for three classes is a binary vector that takes values $(1,0,0),(0,1,0),(0,0,1)$ for classes 1, 2, 3 respectively.

Sketch the resulting network and describe in words how it could be generalized to arbitrary many input dimensions and arbitrary (non degenerate) label distributions. Do you see potential problems with this zero training loss classifier?

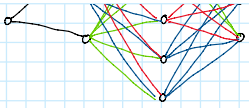
2 Linear Activation Function (5 Points)

For a feed forward network the output of each layer l is calculated iteratively by

$$
\begin{aligned}
Z_0 &= X && (1)\\
\tilde{Z}_l &= Z_{l-1} \cdot B_l + b_l \quad \text{(multiply with weights and add bias vector)} && (2)\\
Z_l &= \phi_l(\tilde{Z}_l) \quad \text{(apply the activation function element-wise to pre-activations)} && (3)
\end{aligned}
$$

Where X and the Z_l and b_l are understood as row vectors.

Prove that if ϕ_l is the identity function then any network (with depth $L > 1$) is equivalent to a 1-layer neural network.



Generalize:

Input Layer: Feed forward all coordinates

1. Hidden Layer:

2^M corners with at least $2^M \geq K$ to have enough decision boundaries

2. Hidden Layer

$D_2 = K$ maps corner to class

Output Layer

$D = |Y_i|$

Problems:

The biggest problem could be when $2^M \geq k$ isn't enough because there aren't any good decision regions to divide the classes.
Not good scalability.

0% train error & 0% test error.

Machine Learning exercise 2

Task 2

(1) $z_0 = X$

(2) $\tilde{z}_l = z_{l-1} \cdot B_l + b_l$

(3) $z_l = \phi_l(\tilde{z}_l)$

if the activation function ϕ_l is the identity function, any neural network with depth $L > 1$ is equivalent to a 1-layer neural network

$\Rightarrow$ output of this network has to be represented as a single affine transformation of the input

ϕ_l is identity function $\Rightarrow$ $\phi_l(z) = z$

For $L > 1$:

Layer 1 $\tilde{z}_1 = z_0 \cdot B_1 + b_1 = X \cdot B_1 + b_1$

$z_1 = \phi_1(\tilde{z}_1) = \tilde{z}_1 = X \cdot B_1 + b_1$

Layer 2: $\tilde{z}_2 = z_1 \cdot B_2 + b_2 = (X \cdot B_1 + b_1) \cdot B_2 + b_2$

$z_2 = \phi_2(\tilde{z}_2) = \tilde{z}_2$

$= (X \cdot B_1 + b_1) \cdot B_2 + b_2$

$= X \cdot (B_1 \cdot B_2) + b_1 \cdot B_2 + b_2$

Generally with (2) and (3), lets assume for layer $l-1$ we have:

$z_{l-1} = X \cdot \underbrace{w_{l-1}}_{\text{product of weight matrices up to } l-1} + \underbrace{b'_{l-1}}_{\text{cumulative bias}}$

Now prove for l:

$\tilde{z}_l = z_{l-1} \cdot B_l + b_l = (X \cdot w_{l-1} + b'_{l-1}) \cdot B_l + b_l$

$z_l = \phi_l(\tilde{z}_l) = \tilde{z}_l$

$= (X \cdot w_{l-1} + b'_{l-1}) \cdot B_l + b_l$

$= X \cdot (w_{l-1} \cdot B_l) + b'_{l-1} \cdot B_l + b_l$

Define: $w_l = w_{l-1} \cdot B_l$

$b'_l = b'_{l-1} \cdot B_l + b_l$

So: $z_l = X \cdot w_l + b'_l$ (4)

From (1), (2), (3) and (4), it is proven that: if the activation function ϕ_l is the identity function, any neural network with depth $L > 1$ is equivalent to a 1-layer neural network

□