

# Terrain Discrimination

Hanna Olbert, Lukas Kyzlik

18. 12. 2019

# Section 1

## Introduction

# Data

The given Data set contains: Data set with Labels 0 to 3  
representing:  
hills,  
rough rocky terrain,  
mountains+plain and  
mountains+valleys.

400 samples  
with 41125 data points each normalized to a range 0 to 1

# Data examples

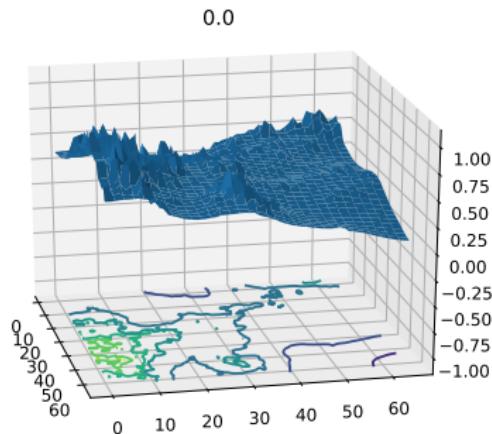


Figure: terrain with label 0

# Data examples

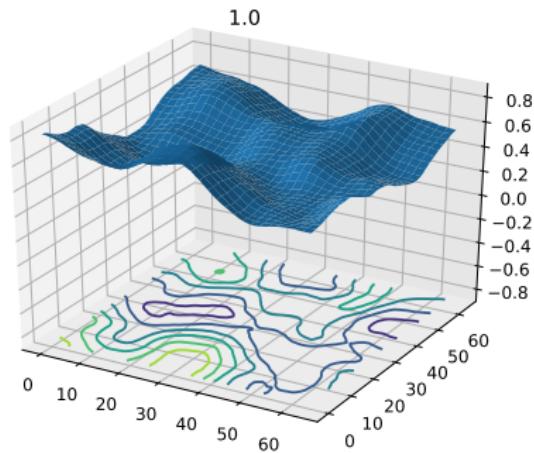


Figure: terrain with label 1

# Data examples

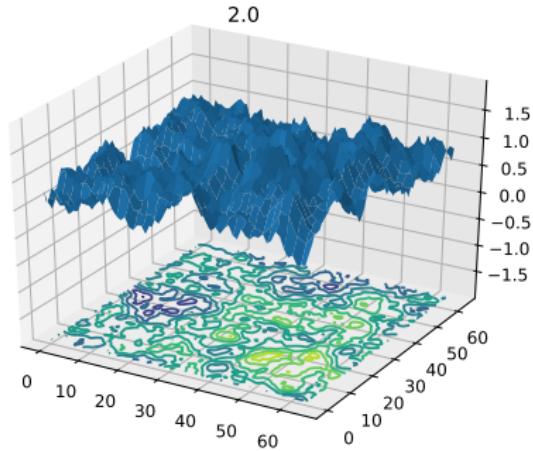


Figure: terrain with label 2

# Data examples

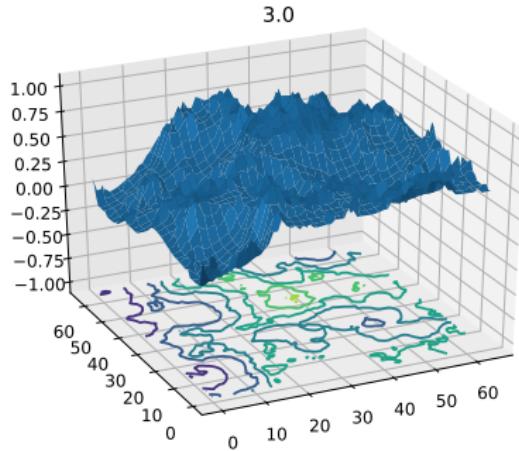
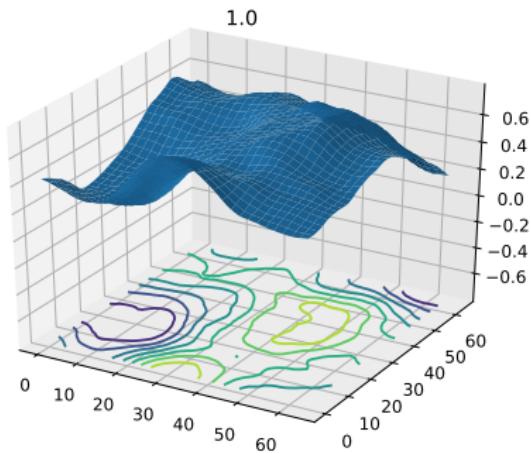
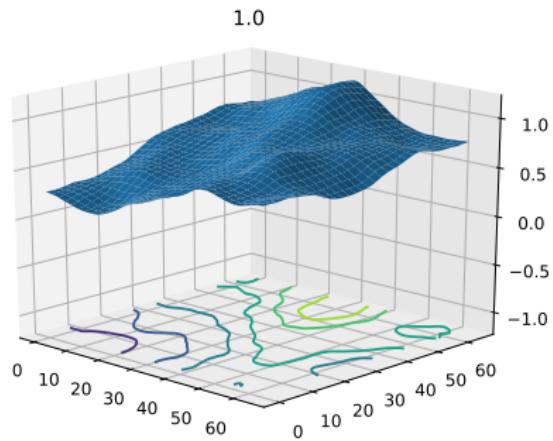


Figure: terrain with label 3

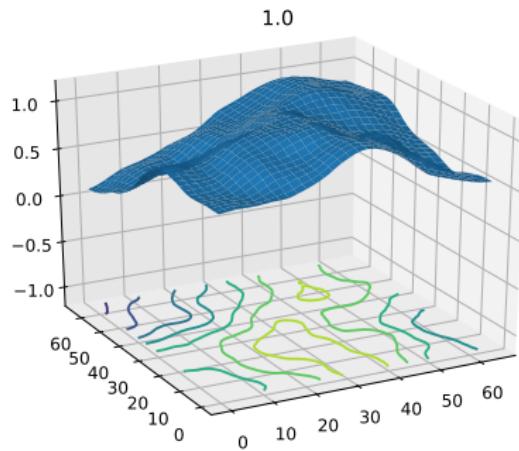
# Examples of class 1



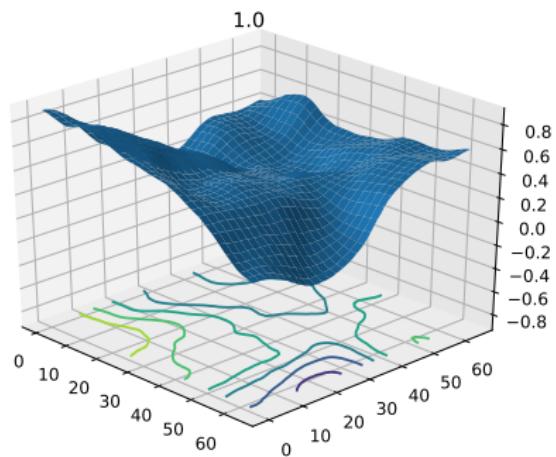
# Examples of class 1



# Examples of class 1



# Examples of class 1



# Task

- a) Differentiate between the terrains with the Label 1 and one other (0)
- b) Differentiate between the terrains with the Label 1 and all others

## Section 2

### Perceptron

# Perceptron – benchmark

Perceptron(max\_iter=100000)

Average training time [s]	69.76
Average score	0.542

Table: Average results from 10 iterations of the test.

## Conclusion

0.542 is benchmark we need to beat with more advanced methods to justify their use.

## Section 3

### Multy-Layer Perceptron (MLP) Classifier

# MLP L-BFGS

**Limited-memory BFGS** is optimization algorithm in the family of quasi-Newton methods. It should perform the best for small datasets like ours.

In the following tests we use this default values for MLP classifier (if some of them are changed, it is noted):

<b>Solver</b>	L-BFGS
<b>Alpha</b>	$10^{-5}$
<b>Hidden layer sizes</b>	1 layer, 15 neurons
<b>Random state</b>	fixed seed
<b>Activation function</b>	ReLU

# MLP L-BFGS – Activation functions

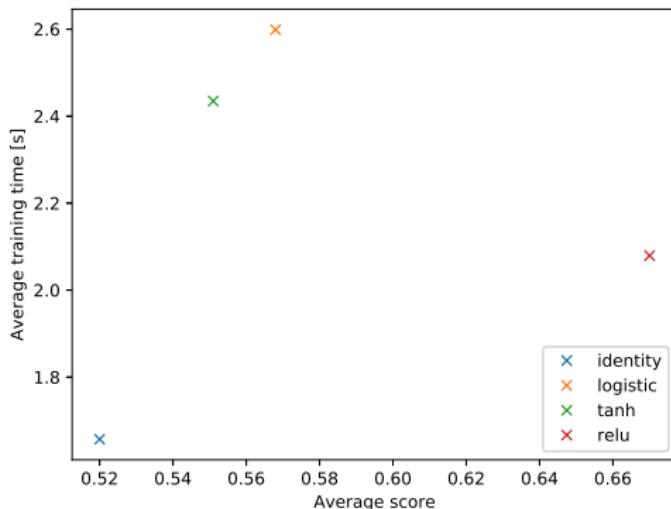


Figure: Average results from 20 iterations with different activation functions.

# MLP L-BFGS – Activation functions

## Conclusion

ReLU activation function gives the best results and is faster than most other functions for this problem.

# MLP L-BFGS – Number of neurons (1 hidden layer)

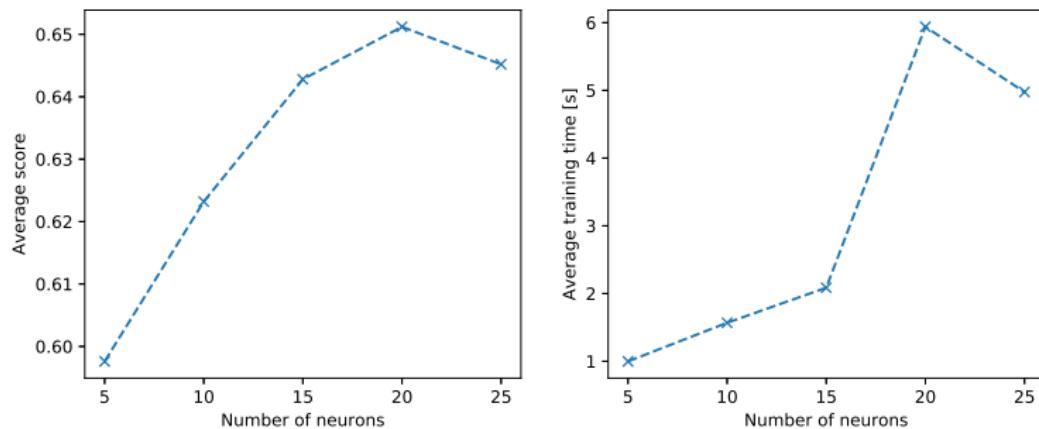


Figure: Average results from 50 iterations with different numbers of neurons in one hidden layer.

# MLP L-BFGS – Number of neurons (1 hidden layer)

## Conclusion

Optimal number of neurons with one hidden layer from perspective accuracy is around 20 for this problem. However, it also has the longest training time. Higher numbers of neurons probably result in over-fitting.

# MLP L-BFGS – Number of neurons (2 hidden layers)

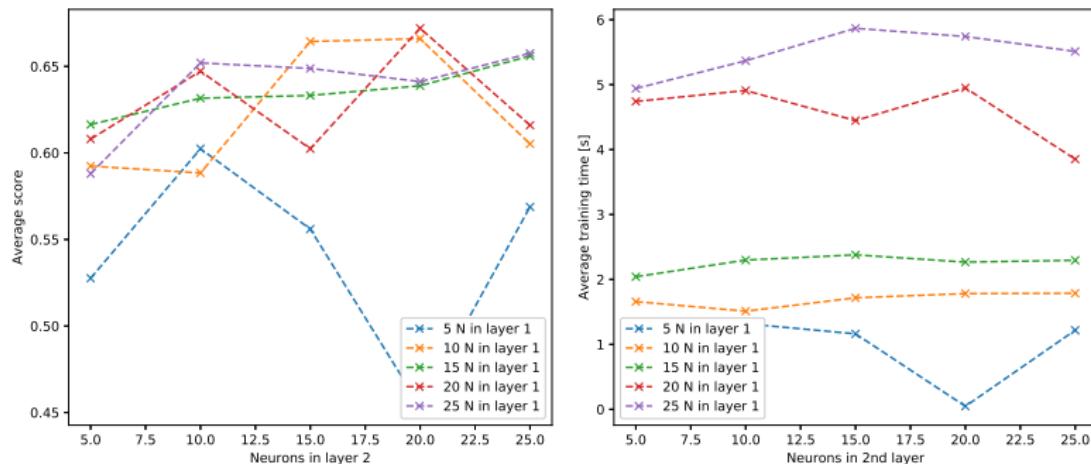


Figure: Average results from 50 iterations with different numbers of neurons in one hidden layer.

# MLP L-BFGS – Number of neurons (2 hidden layers)

## Conclusion

Similar to the previous example the best results are produced by network with around 20 neurons. Overall the best combinations were (20,20) and (10,20).

# MLP SGD

**Stochastic gradient descent** is optimization algorithm for optimizing differentiable or subdifferentiable functions.

In the following tests we use this default values for MLP classifier (if some of them are changed, it is noted):

<b>Solver</b>	SGD
<b>Alpha</b>	$10^{-5}$
<b>Hidden layer sizes</b>	1 layer, 15 neurons
<b>Random state</b>	fixed seed
<b>Activation function</b>	ReLU
<b>Batch size</b>	200
<b>Initial learning rate</b>	0.001

# MLP SGD – Initial learning rate

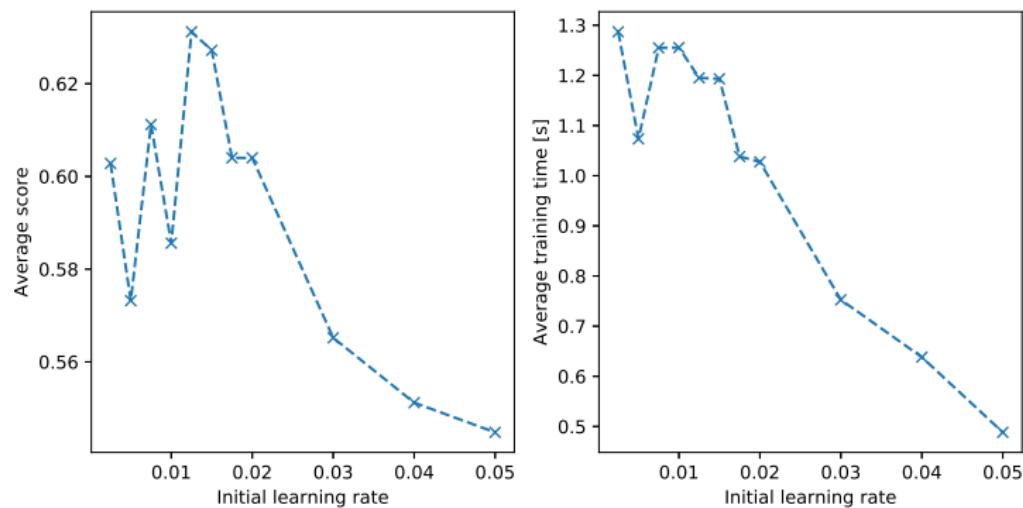


Figure: Average results from 50 iterations with different initial learning rates.

# MLP SGD – Initial learning rate

## Conclusion

The best results are for initial learning rate in interval 0.01 to 0.02. Higher initial learning rates cause the algorithm to "overshoot" and yield significantly worse results.

# MLP SGD – Batch size

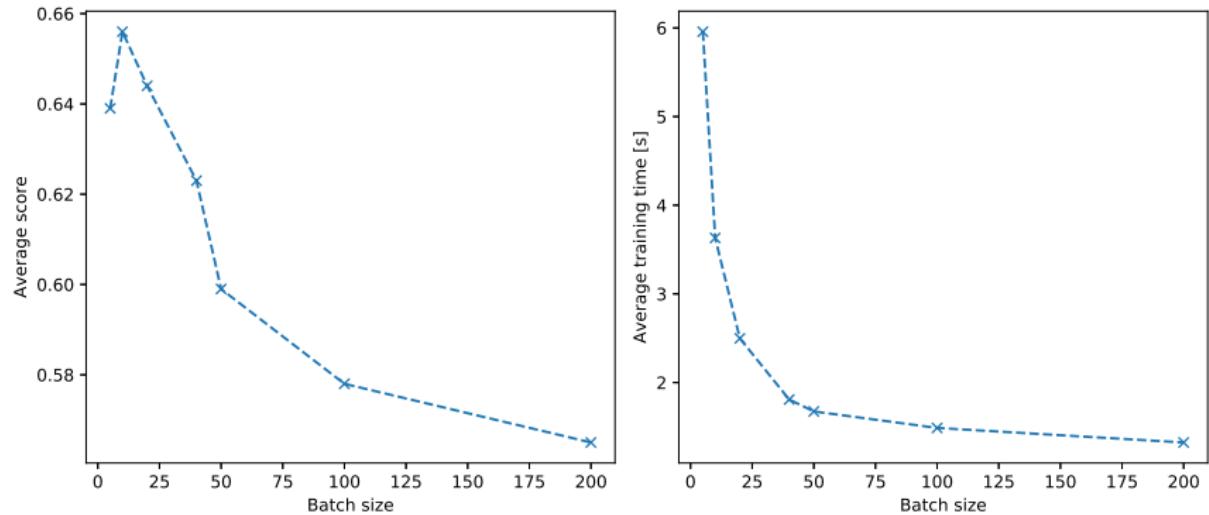


Figure: Average results from 20 iterations with different batch sizes.

# MLP SGD – Batch size

## Conclusion

Smaller batches give generally better results. However, in the case of the batch size 5 maximal number of iterations (200) was exceeded and training stopped prematurely. The lower the batch size the longer the training time.

# Future Tasks

- Optimization of the MLP
- Multilabel differentiation
- Clustering