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

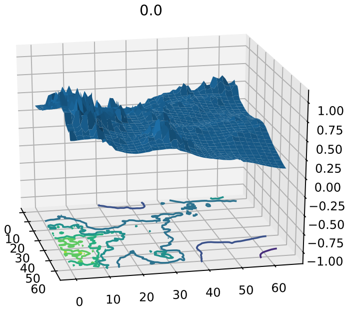


Figure: terrain with label 0

Data examples

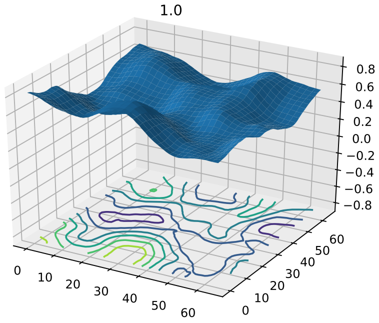


Figure: terrain with label 1

Data examples

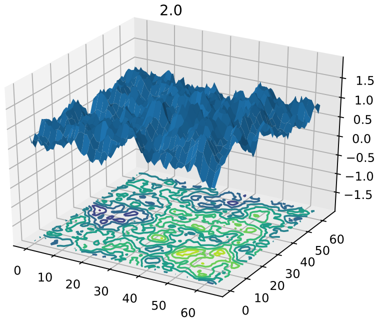


Figure: terrain with label 2

Data examples

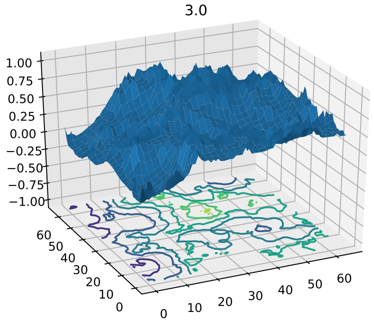
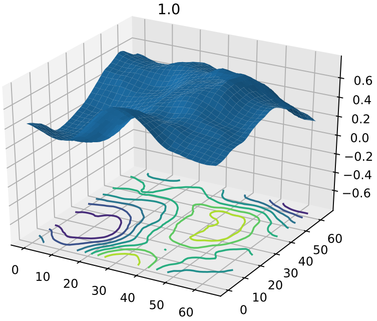
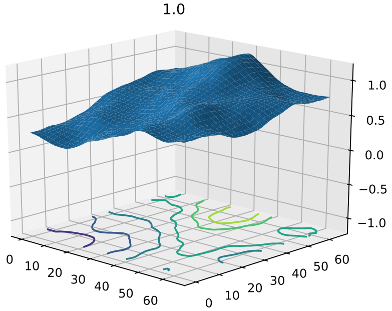


Figure: terrain with label 3

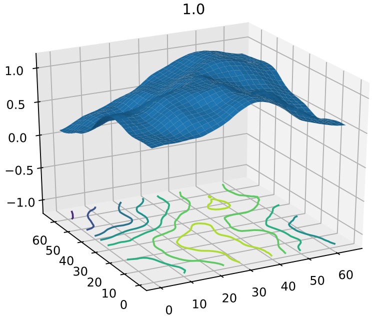
Examples of class 1



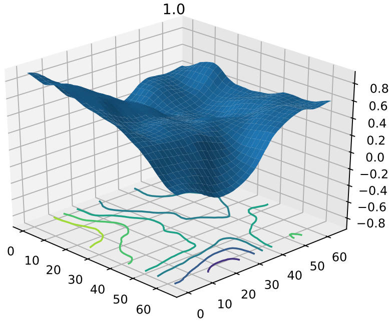
Examples of class 1



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

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 the best average score we could achieve with a Perceptron. This is negligibly higher than choosing classes for data points at random and shows therefore that a linear approach is not suited for this data set.

Section 3

K-Nearest Neighbor

K-Nearest Neighbor benchmark

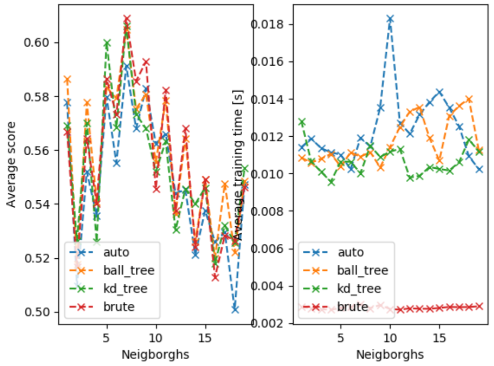


Figure: Average score and training time in relation to the number of neighbors

K-Nearest Neighbor benchmark

Conclusion

0.608 is the best average score we could achieve with a K-Nearest Neighbor Classifier with 7 neighbors and the brute algorithm. This average is computed over 50 iterations. This is the benchmark we need to beat with more advanced methods to justify their use.

Section 4

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):

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

MLP L-BFGS – Activation functions

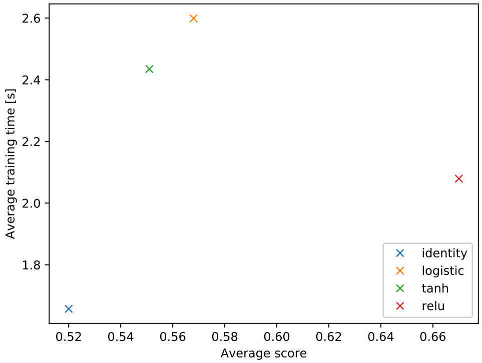


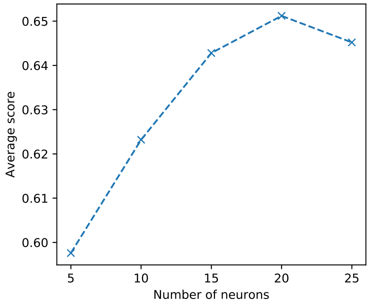
Figure: Average results from 20 iterations with different activation functions.

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)



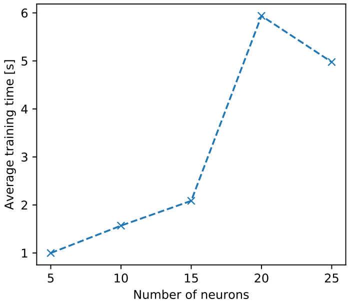


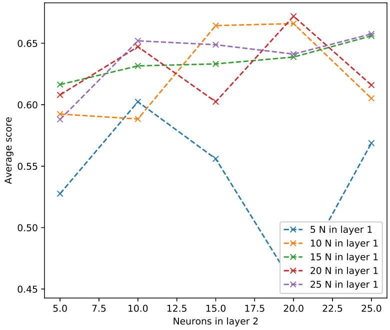
Figure: Average results from 50 iterations with different numbers of neurons in one hidden layer.

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 layes)



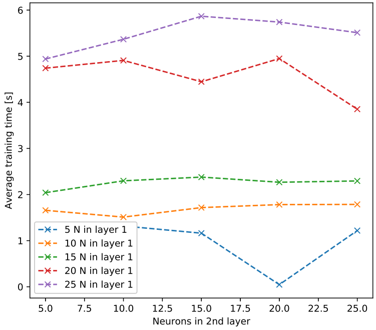


Figure: Average results from 50 iterations with different numbers of neurons in one hidden layer.

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):

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

MLP SGD – Initial learning rate

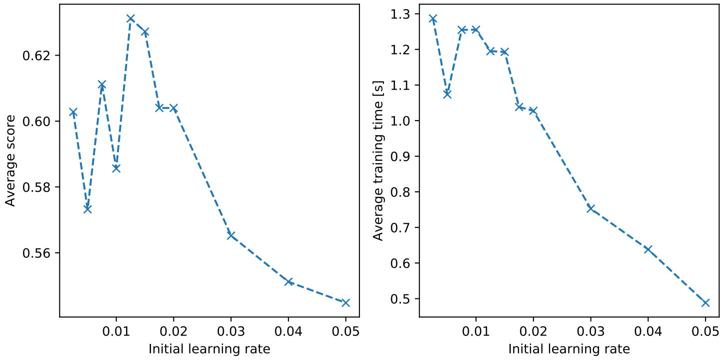


Figure: Average results from 50 iterations with different initial learning rates.

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

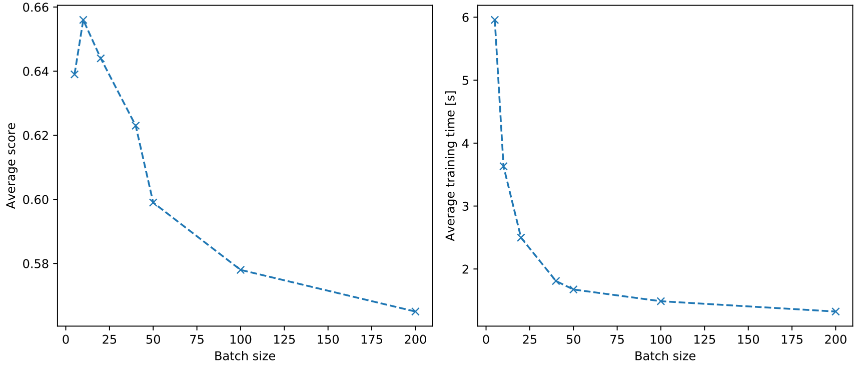


Figure: Average results from 20 iterations with different batch sizes.

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.

MLP results

Solver	L-BFGS
Alpha	1e
Hidden layer sizes	2 layers (20,20)
Random state	fixed seed
Activation function	ReLU
Batch size	25
Initial learning rate	0.001
Result (avg. for 20 iterations)	63%

Table: The parameters for which the MLP Classifier produced the best results in the 2 class case.

PCA dimensionality reduction

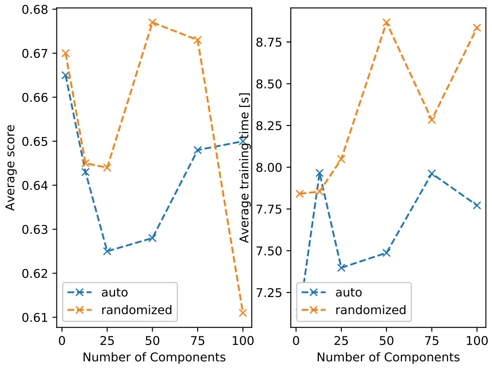


Figure: Comparison of the average score of the classifier for different number of dimensions

PCA

Conclusion

The reduction of dimensions before training did produce improved results when using the randomized solver of the PCA. The average score of 20 iterations with the data points reduced to 50 dimensions was 67.7% on average for 20 iterations.

Data Visualisation

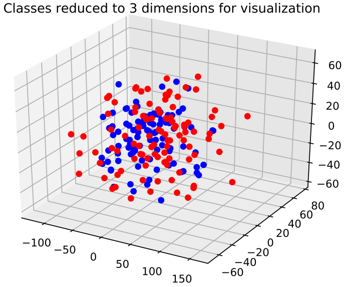


Figure: The data was reduced to 3 dimensions with the PCA algorithm to make visualization possible.

Data Visualisation

Conclusion

The data in these 3 dimensions is very intertwined and not in any obvious way separable. The data in these 3 dimensions only represents 75% of the variance.

KMeans

K-means clustering on the digits dataset (PCA-reduced data)
Centroids are marked with white cross

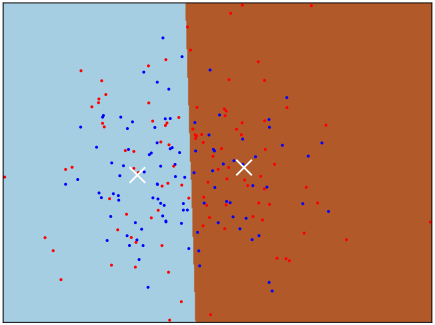


Figure: Kmeans clustering on PCA dimensional reduced data

Conclusion

The data points in these 2 dimensions are not separable, therefore the K-Means Clustering algorithm is only able to classify half of the data correctly. The 2 dimensions of the data only represent 61% of the variance of the data. Using all the dimensions of the data to train and test the K-Means clustering does not produce better results.

Section 7

Multilabel Classification

Data Visualization

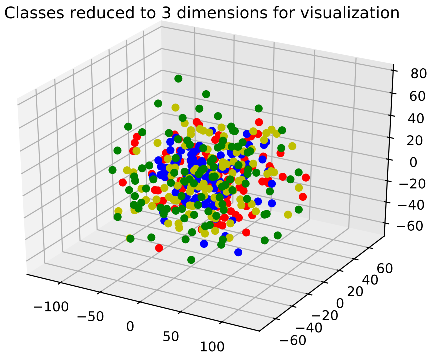


Figure: All data points of the four classes reduced to 3 dimensions with PCA.

Data Visualization

Conclusion

For these 3 dimensions, which represents 63% of the variance of the data set, there is, like in the 2 class case, no clear separation possible.

KMeans

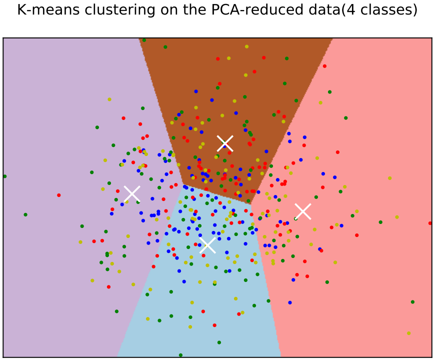


Figure: K-Means clustering on the terrain data set which was reduced to 2 dimensions.

KMeans

Conclusion

Similar to the 2 class case is the K-Means clustering not successful in classifying any class correctly since the data reduced to these dimensions is not meaningful. The data points in these dimensions only describe 50% of the variance. Using all dimensions did not produce a better result.

PCA

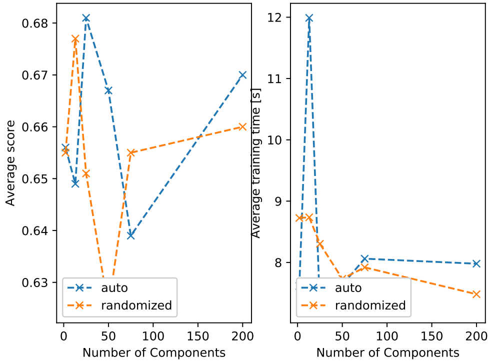


Figure: mlp classifier, with the same parameters as in the 2 class case, tested on differntly through PCA reduced data.

PCA and MLP result

Conclusion

For the same parameters that were used in the 2 class classifying, the Multi Layer Perceptron trained on the PCA reduced data produced an average score of 68% for 25 dimensional data.

Conclusions

Conclusion

None of our approaches were able to achieve a average score of higher than 68%, which still beat the 7-NN benchmark of 60.8%. This should be due to the fact that there are more features that need to be considered for successful classification. The terrain data set of the different terrains is non-linear and -like pictures- has the characteristic of being spacially dependent on the surrounding values. Therefore a Deep Learning approach is the most promising, since through convolutional Layers the spacial aspect of the features is also taken into consideration.