Project 2: Machine Learning Mine versus Rock

By Aniket N Prabhu

ASU ID: 1217704488

Results with random seed unspecified:

Maximum Accuracy (%)	Number of components
90	6
90	19
92	19
90	17
90	4
89	8
89	20
90	8
90	10

After conducting multiple tests with random seed unspecified, the maximum accuracy is near 90% which means that there is one out of ten chances in which a mine will be detected as a rock and I will be blown to smithereens or a rock will be detected as a mine. With random seed=0, the accuracy increased to 94% and the number of components needed for maximum accuracy=10. Any other random seed gives accuracy near 90% but not as high as random state=0, the second highest being 92% at random state=8. Thus, it seems better to specify a random seed, use that seed for each number of components and then try a different seed every time rather than leaving it unspecified. Moreover, the confusion matrix tells us how well the neural network has performed. Example: for 94% maximum accuracy, we get the following confusion matrix:

```
[[25  4]
 [ 0 34]]
```

From this confusion matrix, we infer that:

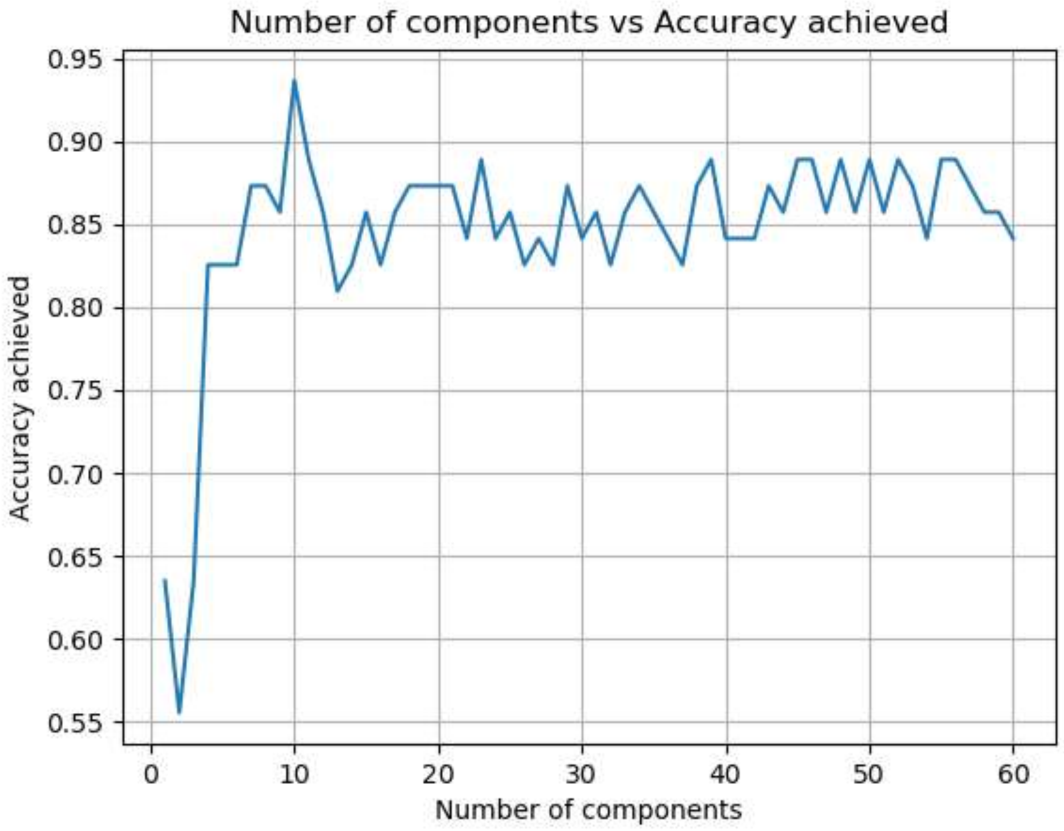
-25 samples were actually rocks as well as predicted as rocks.

-4 samples were actually rocks but predicted as mines.

-0 samples were actually mines but predicted as rocks

-34 samples were actually mines as well as predicted as mines.

This means that I have a better chance of surviving if I specify a random set as even if I mistake a rock for a mine, I would not be mistaking a mine for a rock and ending up dead.



The above plot is taken for the following statement:

mlp = MLPClassifier(hidden_layer_sizes=100, activation='logistic', max_iter=2000, alpha=0.00001, solver='lbfgs', tol=0.0001, random_state=0)

The plot tells us that starting from zero, the number of components indicating underfitting that is, the number of components is insufficient to run the model. Suddenly, we see a steep rise in the accuracy with respect to the number of components. As soon as the number of components hits 10, we get the maximum accuracy of 94%. Taking any more number of components results in an accuracy fall it then hovers around 85% which means that we are taking/considering more data than what is sufficient (10) indicating overfitting. As for the parameters of MLPClassifier, hidden_layer_size=100 gave the best accuracy. ‘logistic’ and ‘tanh’ activation functions gave similar and the best accuracy (94%) although ‘tanh’ required 5 components instead of 10 components to give maximum accuracy. Using ‘adam’ solver instead of ‘lbfgs’ gives a similar result although it runs slower and unlike ‘lbfgs’, the accuracy decreases as the number of components increases. Tolerance is set to 0.0001 or 1e-4 as I believe that increasing it would cause underfitting whereas decreasing it further would result into overfitting.