



WASHINGTON STATE
UNIVERSITY

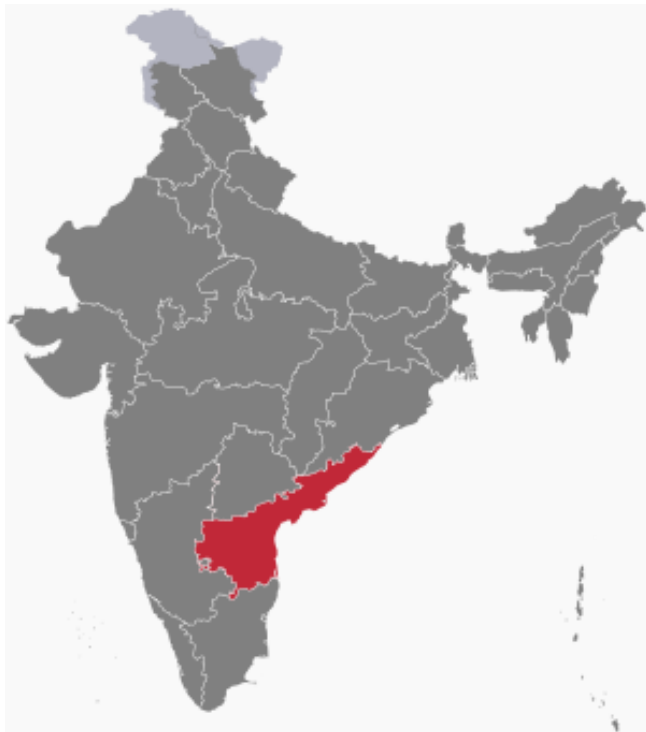
Predicting Liver Disease Using a Feed Forward Neural Network

Aryan Ritwajeet Jha

Currently pursuing PhD ECE (Power Systems)

Problem Statement at a Glance [1, 2]

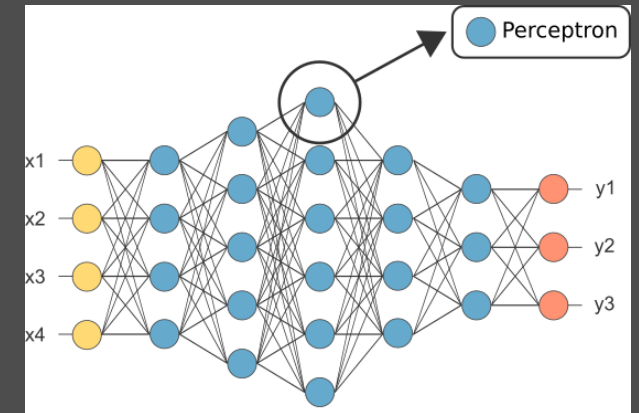
- Scenario: A patient arrives into the hospital with symptoms of liver disease
- Decision: To determine whether the patient has possible liver disease and be admitted for treatment
- Inputs: A set of 10 diagnostic quantities obtained from the patient.
- Additional Inputs for Training: A disease classification label for all arriving patients given by an expert physician.



Diagnostic Quantity	Type
Age	Integer
Gender	Nominal (Binary)
TB	Continuous
DB	Continuous
Alkphos	Integer
Sgpt	Integer
Sgot	Integer
TP	Continuous
ALB	Continuous
A/G Ratio	Continuous

Approach to Solve the Problem

- Train a Feed Forward Neural Type equation here. Network
- For 10 features and 1 classification, we can use a neural network represented by a vector $[10, 10, 10, 1]$.
- For activation function, we use the sigmoid function.



Prediction Function

$$f(x) = \sigma(W_d \sigma(\dots \sigma(W_2 \sigma(W_1 x)) \dots))$$

$$L_1 = \sigma(W_1 x)$$

$$L_2 = \sigma(W_2 L_1)$$

$$\vdots$$

$$L_d = \sigma(W_d L_{d-1})$$

$$F = \frac{1}{2} \|L_d - y\|^2$$

Backpropagation

$$h_d = (L_d - y)L_d(1 - L_d), \quad G_d = h_d L_{d-1}^T$$

$$h_{d-1} = W_d^T h_d L_{d-1}(1 - L_{d-1}), \quad G_{d-1} = h_{d-1} L_{d-2}^T$$

$$\vdots$$

$$h_2 = W_3^T h_3 L_2(1 - L_2), \quad G_2 = h_2 L_1^T$$

$$h_1 = W_2^T h_2 L_1(1 - L_1), \quad G_1 = h_1 x^T$$

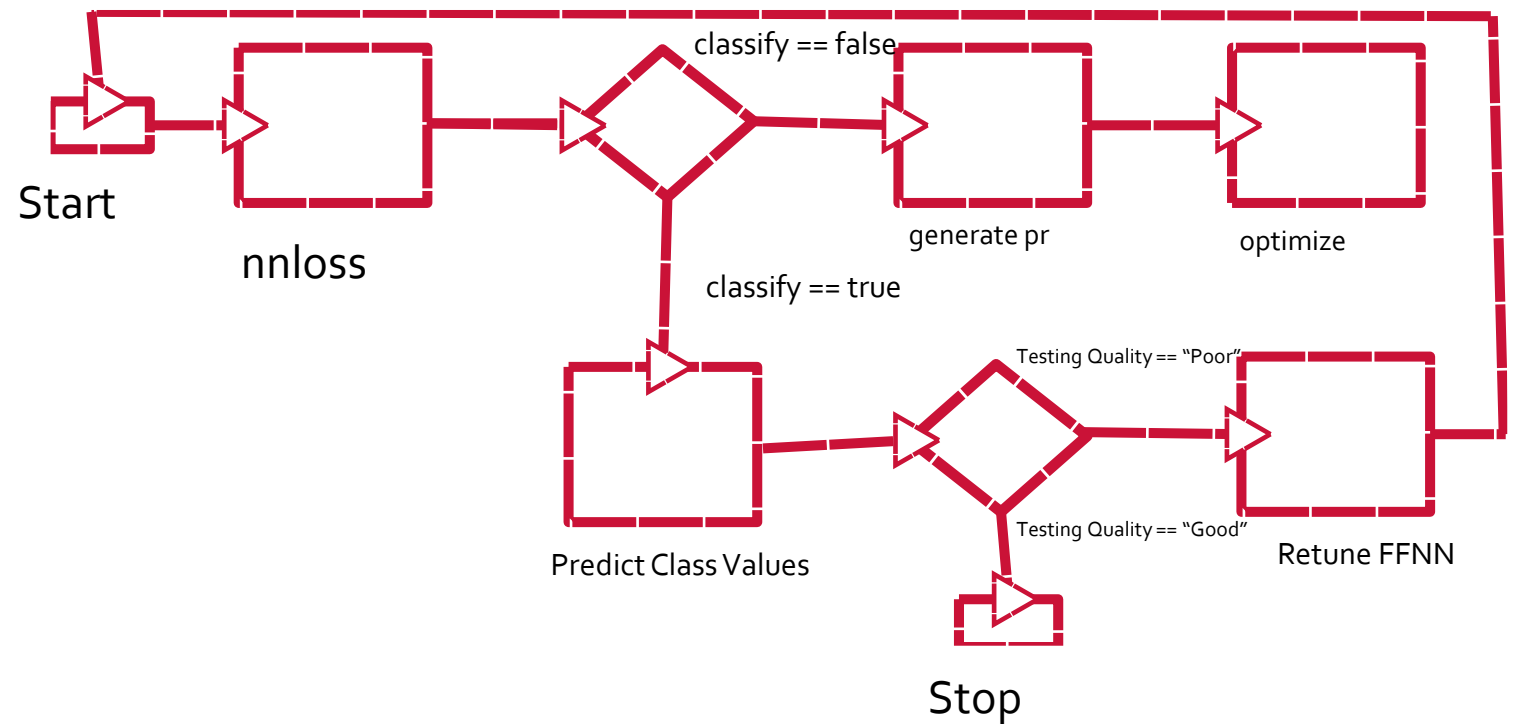
Preprocessing

Given a chosen architecture for the FFNN, we would like to preprocess the dataset for our solver to have an easier time training the Neural Network.

1. Drop Missing Data
2. Convert Nominal String values for Gender to Binary values. $\rightarrow \{0, 1\}$
3. Normalize all row values for a given column against the maximum value in that column $\rightarrow [0, 1]$
4. Choose training and testing data (Random 70 – 30 split)
5. Choose initial weights w_0
6. Map class values to $\{\frac{1}{3}, \frac{2}{3}\}$

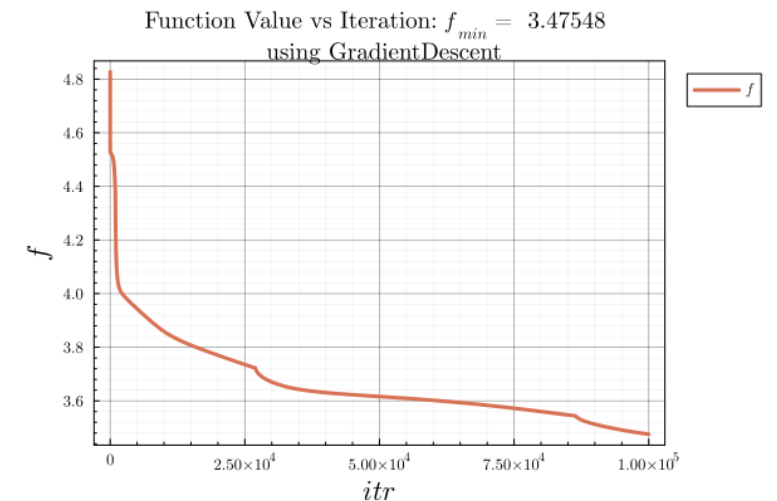
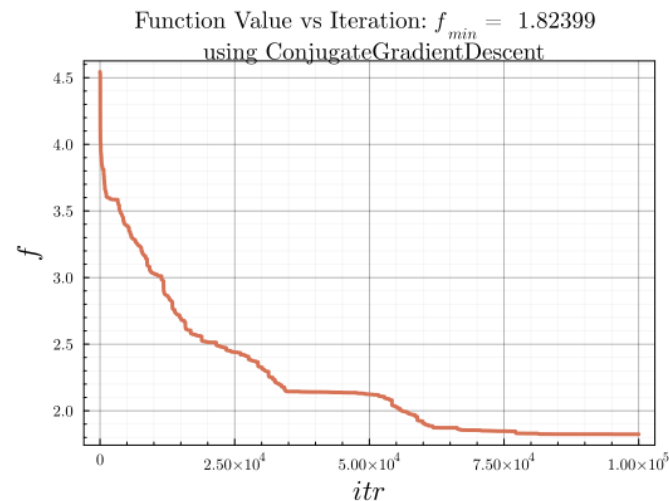
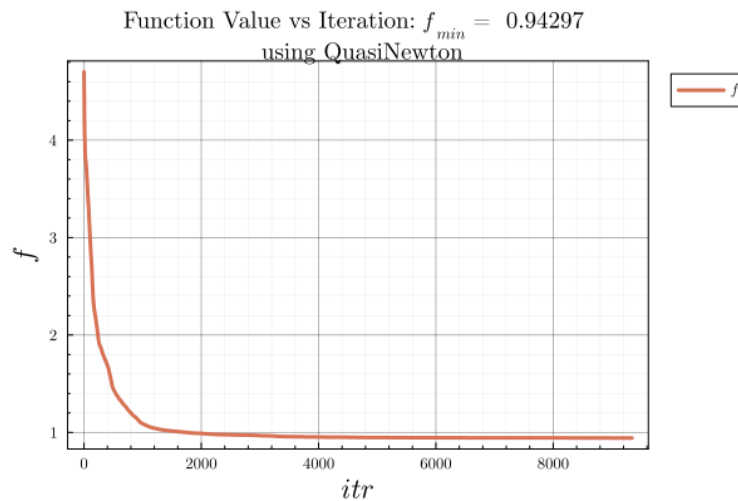
WSU

Code base for Training and Using the FFNN



Some results

- Results varying, some better than others. Possible causes are:
 - Random splitting of training and testing datasets
 - Random initialization of weights
- For the 'base' FFNN of dimensions [10, 10, 10, 1]:



Some results

```
Failed to converge despite 100000 iterations! 😞
*****
Method Used: ConjugateGradientDescent
Linesearch method used: StrongWolfe
*****
Best function value = 1.5190728911905023
*****
Best Known Nonzero Variables:
Only upto 15 values will be printed.
x[1] = -1.3207187616083105
x[2] = -0.4285531460063524
x[3] = -4.360010210198691
x[4] = 0.4751582454508718
x[5] = -4.9709087449052705
x[39] = 2.1851311613594726
x[72] = 5.7125270015950695
x[106] = 1.9931772952961693
x[139] = -0.8364736057833893
x[172] = -8.44485447342595
x[206] = -8.526202685346068
x[207] = -0.10359733774767228
x[208] = 8.826201360892739
x[209] = 7.959290562199788
x[210] = 6.232098675661507
*****
Number of fevals = 100001
Number of gevals = 100000
Number of evals = 21100001
*****
Tolerance criteria used:
dftol = 1.0e-15
length(y_pred) = 405
length(y_true) = 405
Training Data:
Accuracy: 0.9185185185185185
Specificity: 0.958904109589041
Sensitivity: 0.814592920353983
length(y_pred) = 174
length(y_true) = 174
Testing Data:
Accuracy: 0.7068965517241379
Specificity: 0.7786885245901639
Sensitivity: 0.5384615384615384
```

```
*****
Method Used: QuasiNewton
Linesearch method used: StrongWolfe
*****
Optimal function value = 0.6438640832869442
*****
Optimal Nonzero Variables:
Only upto 15 values will be printed.
x[1] = 2.4702411263864477
x[2] = -13.959087385891898
x[3] = -60.92073549610952
x[4] = -9.584867524359968
x[5] = 9.139166904718698
x[39] = -154.0993416606186
x[72] = -10.007516504163736
x[106] = 8486.102907399218
x[139] = -5578.119192645641
x[172] = 74.85080192095211
x[206] = 699.7225571382161
x[207] = 4444.457155000058
x[208] = -2903.7164189432197
x[209] = -701.1544543734368
x[210] = -1.318481099668291
*****
Number of fevals = 61511
Number of gevals = 61510
Number of evals = 12978611
*****
*
Tolerance criteria used:
dftol = 1.0e-15
length(y_pred) = 405
length(y_true) = 405
Training Data:
Accuracy: 0.9679012345679012
Specificity: 0.976027397260274
Sensitivity: 0.9469026548672567
length(y_pred) = 174
length(y_true) = 174
Testing Data:
Accuracy: 0.6781609195402298
Specificity: 0.7868852459016393
Sensitivity: 0.4230769230769231
```

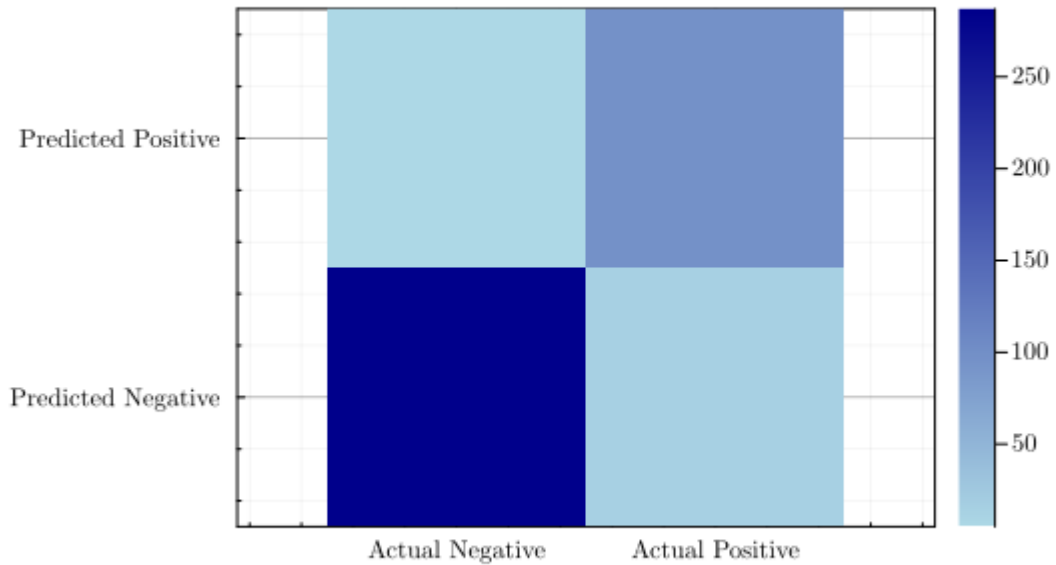
```
Convergence achieved in 81116 iterations 😊
*****
Method Used: QuasiNewton
Linesearch method used: StrongWolfe
*****
Optimal function value = 0.5925323855773336
*****
Optimal Nonzero Variables:
Only upto 15 values will be printed.
x[1] = -7.027301029539397
x[2] = -32.765850366664004
x[3] = -25.632017539628436
x[4] = -27.91163115796061
x[5] = -36.76425923366128
x[39] = -59.72535990966416
x[72] = -89.9468640598543
x[106] = 22.741134468753046
x[139] = 1.6638562817688438
x[172] = 261.43587165831184
x[206] = -3734.7635254797747
x[207] = 1840.0356284281327
x[208] = 2590.8648633719877
x[209] = -711.90339752127
x[210] = -3722.379798733183
*****
Number of fevals = 81116
Number of gevals = 81115
Number of evals = 17115266
*****
*
Tolerance criteria used:
dftol = 1.0e-15
length(y_pred) = 405
length(y_true) = 405
Training Data:
Accuracy: 0.9703703703703703
Specificity: 0.9828767123287672
Sensitivity: 0.9380530973451328
length(y_pred) = 174
length(y_true) = 174
Testing Data:
Accuracy: 0.6494252873563219
Specificity: 0.7049180327868853
Sensitivity: 0.5192307692307693
```

- Results varying, some better than others. Possible causes are:
- Training Accuracy varying between 91 – 97%
- Testing Accuracy varying between 65 – 71%
- For the 'base' FFNN of dimensions [10, 10, 10, 1]:

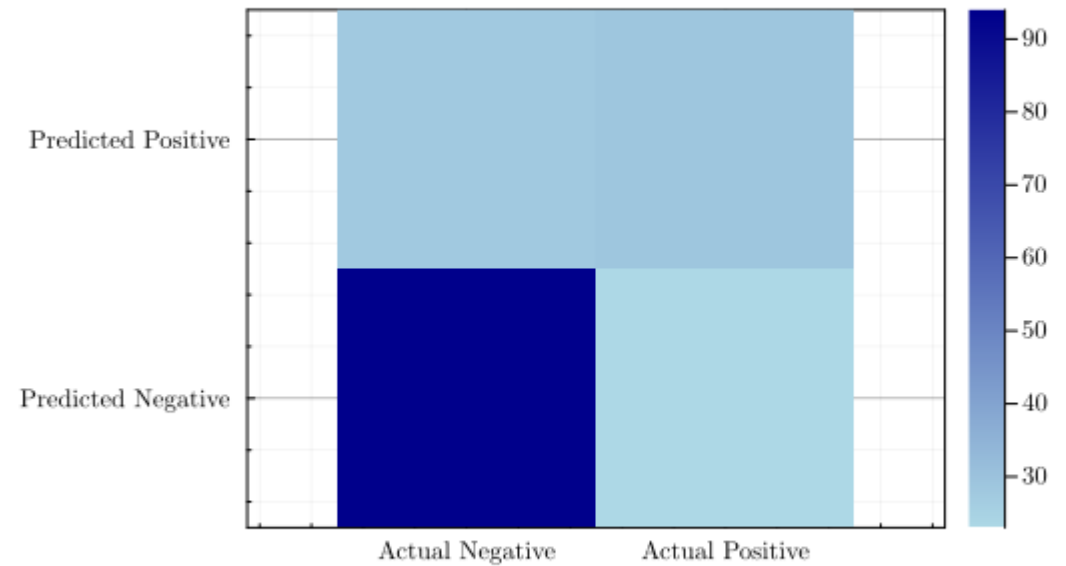
Some results

- Typical Confusion Matrix for the 'base' FFNN of dimensions [10, 10, 10, 1]:

Fit of FFNN on the Training Data



Fit of FFNN on the Testing Data



Other NN architecture experimentation

(all runs using QN-BFGS)

- Using a single hidden layer: [10, 10, 1] : Poorer fit, even on the Training Data (80 – 85%)
- Using less number of hidden nodes: [10, 5, 5, 1]: Less overfitting, gets lower Training Data accuracy (85%) but similar testing accuracy as 'base' architecture (65 – 70%).

WSU

Other untested ideas

- Try dropping some features. This paper [3] talks about how Albumin and A\G Ratio features are not that important when performing a diagnosis.
- Try another activation function, such as $\tanh()$ or *ReLU*.

WSU

Thank You.



WSU

Appendix and References follow



WSU

References

- [1] Ramana,Bendi and Venkateswarlu,N.. (2012). ILPD (Indian Liver Patient Dataset). UCI Machine Learning Repository. <https://doi.org/10.24432/C5Do2C>.
- [2] Class notes and Problem Statements provided by Prof. Tom Asaki as part of the course MATH 564 Convex and Nonlinear Optimization at Washington State University, for the semester of Fall 2023.
- [3] Straw I, Wu | H Investigating for bias in healthcare algorithms: a sex-stratified analysis of supervised machine learning models in liver disease prediction | BMJ Health & Care Informatics 2022;29:e100457. doi: 10.1136/bmjhci-2021-100457

WSU