Jordan Black

CGS2020 202501

Professor Lakshmi prayaga

4/14/2025

1. **What are the three broad components of a neural network?**

**Input layer: Receives data (Features) For processing.**

**Hidden layer: transforms input using weights and activation functions.**

**Output layer: produces the final prediction or classification.**

1. **In your own words describe what is a hidden layer in a neural network?**

**The hidden layer is where the computations happen between the input and output sections. Here, the hidden layer applies weights and activation functions to capture complex relationships and patterns that aren’t directly visible from raw input.**

1. **What are the main difficulties with a neural network?**

* **Overfitting ( to complex = memorizing data)**
* **The need for larger datasets.**
* **Long training times for the Neural net**
* **Interpretability (This is often a black box)**
* **Choosing architecture (layers, neurons)**

**4. Complete the following programming project:**

**Use the example file (nnEx1v2.r) provided and edit it as necessary. This time you will create a neural network by using 3 variables (GRE, GPA and rank) to predict if a student will be admitted to a college or not.**

**Use the example file (nnEx1v2.r) provided and edit it as necessary. This time you will create a neural network by using 3 variables (GRE, GPA and rank) to predict if a student will be admitted to a college or not.**

**a. Create the training set as follows**

***gre=c(380,660,800,640,520,760)*  
*gpa=c(3.61,3.67,4,3.19,2.93,3)*  
*rank=c(3,2,4,1,1,2)*  
*admit=c(0,1,1,1,0,1)***

**Screenshot:**

***A number of numbers and letters

AI-generated content may be incorrect.***

**Next, create a data frame**

***df=data.frame(gre,gpa,rank,admit)***

**Screenshot:**

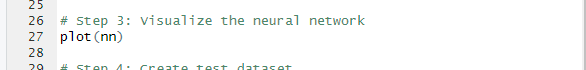
******

**Follow the rest of the example:**

***Call the neural network and assign the result to a variable called nn***

***Plot nn***

**Screenshot:**

****

**Next, create the test data set with these values:**

***gre=c(780,360,800)*  
*gpa=c(3.87,2.56,3.75)*  
*rank=c(4,3,2)*  
*test = data.frame(gre,gpa,rank)***

**Screenshot:**

**A computer code with black text

AI-generated content may be incorrect.**

**Next call the predict function and print the result**

**Prediction on test data:**

**A close up of a text

AI-generated content may be incorrect.**

**Finally convert the predictions to binary values using 0.25, then try 0.50 and 0.75 like given below**

**prob <- Predict$net.result  
pred <- ifelse(prob>0.25, 1, 0)  
pred**

**prob <- Predict$net.result  
pred <- ifelse(prob>0.50, 1, 0)  
pred**

**prob <- Predict$net.result  
pred <- ifelse(prob>0.75, 1, 0)  
pred**

**Screenshot:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Original Code For Neural Network:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Updated Code For Neural Network:**

A screenshot of a computer

AI-generated content may be incorrect.

**# ============================================**

**# Neural Network: Predicting college admission**

**# ============================================**

**# Step 0: Install and load the necessary package**

**# Uncomment the next line if 'neuralnet' is not already installed**

**# install.packages("neuralnet")**

**library(neuralnet)**

**# Step 1: Create training dataset**

**gre <- c(380, 660, 800, 640, 520, 760)**

**gpa <- c(3.61, 3.67, 4.00, 3.19, 2.93, 3.00)**

**rank <- c(3, 2, 4, 1, 1, 2)**

**admit <- c(0, 1, 1, 1, 0, 1)**

**# Combine into a dataframe**

**df <- data.frame(gre, gpa, rank, admit)**

**# Step 2: Train the neural network**

**nn <- neuralnet(admit ~ gre + gpa + rank,**

**data = df,**

**hidden = 3,**

**act.fct = "logistic",**

**linear.output = FALSE)**

**# Step 3: Visualize the neural network**

**plot(nn)**

**# Step 4: Create test dataset**

**gre <- c(780, 360, 800)**

**gpa <- c(3.87, 2.56, 3.75)**

**rank <- c(4, 3, 2)**

**test <- data.frame(gre, gpa, rank)**

**# Step 5: Make predictions on test data**

**Predict <- compute(nn, test)**

**prob <- Predict$net.result**

**# Step 6: Convert probabilities into binary predictions**

**# Threshold: 0.25**

**pred\_25 <- ifelse(prob > 0.25, 1, 0)**

**# Threshold: 0.50**

**pred\_50 <- ifelse(prob > 0.50, 1, 0)**

**Part 4: Print each set of the results, provide screen shots. Explain your results, for example**

**when you use 0.25, how many admits did you get and how many no admits, now do the same for 0.50 and 0.75.**

**1st attempt for Neural Network 0.25,0.50,0.75:**

**1st attempt:**

**Admits: 3**

**No Admits: 0**

**When the threshold is set to its minimum value of 25%, any predicted probability above this figure results in student admission. The network labels all borderline cases or robust predictions as positive outcomes. The neural network's low confidence levels still lead to decisions about admissions.**

**Threshold 0.50 → 1 1 1**

**Admits: 3**

**No Admits: 0**

**The network maintains its positive assessment of all three students for admission under a balanced 50% threshold. The neural weights demonstrate a strong preference for accepting students with these particular features.**

**Threshold 0.75 → 1 0 1**

**Admits: 2**

**No Admits: 1**

**The increased standard causes one student to fall short of the requirement. Their slightly lower GPA or GRE scores caused them to fall below the stricter threshold and resulted in their exclusion.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

A diagram of a number

AI-generated content may be incorrect.

**Attempt 2: for neural Network:**

**Threshold 0.25 > 1 1 1**

**Admits: 3**

**No- Admits: 0**

**Explanation: At this lenient threshold, all predicted probabilities (~0.6696) exceeded 0.25, so all students were classified as admitted. This highlights that a low threshold might inflate admission predictions, risking false positives.**

**Threshold 0.50 →** 1 1 1

**Admits: 3**

**No admits: 0**

**Explanation: the model shows confidence, with all predictions still exceeding 0.50. This suggests a strong pattern in the training data linking GRE, GPA, and RNAK to admissions, though this confidence could be artificially high.**

**Threshold 0.75 → 0 0 0**

**Admits: 0**

**No Admits: 3**

**From Attempt 1, the stricter threshold (0.75) still allowed two students in. Here, all predictions (just under 0.67) fall below 0.75, indicating the model lacks high-confidence admissions. This could suggest that overfitting to mid-range is probable or lower weight strength from the latest initial.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**3rd Attempt for Neural Network:**

**3rd Attempt**

**Threshold 0.25 → 1 1 1**

**Threshold 0.50 → 1 0 1**

**Threshold 0.75 → 1 0 0**

**The results demonstrate how minimal weight initialization or training step changes can impact outcomes. The test case in the middle scored below 0.50, while the third test case scored under 0.75. The model demonstrates a delicate decision boundary in proximity to the threshold line. Under more rigorous thresholds, the network demonstrates reduced confidence about those two predictions.**

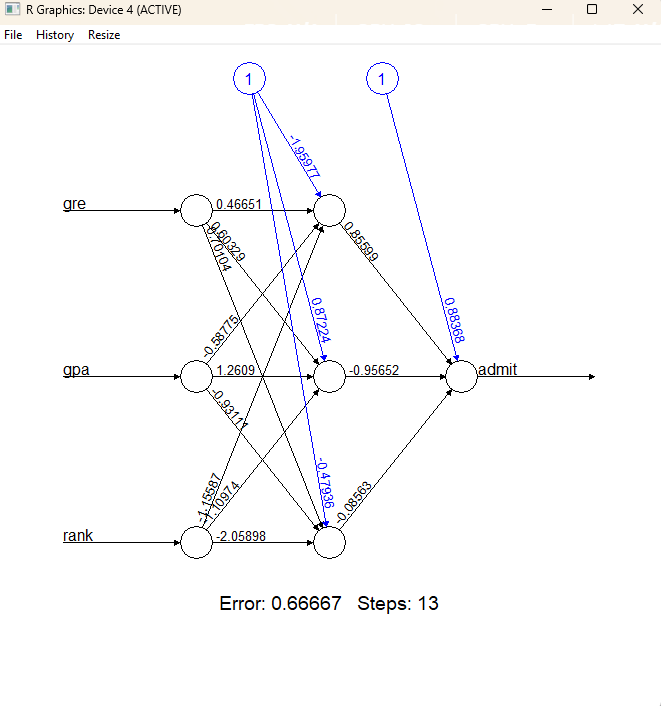
**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**3rd Attempt Screenshot for Neural Network:**

****

**4th Attempt for Neural Network:**

**Threshold 0.25 → 1 1 1**

**Threshold 0.50 → 1 1 0**

**Threshold 0.75 → 1 0 0**

**The third prediction diverged from previous attempts. The third student's input must be on the threshold line for all the given limits. The neural network shows uncertainty for this case when the prediction changes from 1 to 0 between thresholds 0.50 and 0.75 based on threshold conservativeness.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Continued screenshots:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

A screenshot of a computer

AI-generated content may be incorrect.

**5th attempt for neural network:**

**Threshold 0.25 → 1 1 1**

**Threshold 0.50 → 1 1 1**

**Threshold 0.75 → 1 0 1**

**This mirrors the early attempts again. At the most stringent threshold setting, just one student faces exclusion. The results confirm student 2's position near the decision boundary within the model’s internal logic, representing a gray area situation.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

..,

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Screenshots continued:**

A screenshot of a computer

AI-generated content may be incorrect.

**Summary Table: Threshold impacts across all 5 Neural Networks attempts:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| **Attempt** | **Threshold** | **Predicted (Admit=1, Not Admit=0)** | **# Admits** | **# No Admits** | **Notes** |
| **1st** | **0.25** | **1 1 1** | **3** | **0** | **Very lenient, admits all** |
|  | **0.50** | **1 1 1** | **3** | **0** | **Confident predictions above 0.5** |
|  | **0.75** | **1 0 1** | **2** | **1** | **Slight drop in confidence on 2nd test case** |
| **2nd** | **0.25** | **1 1 1** | **3** | **0** | **All probabilities ~0.67** |
|  | **0.50** | **1 1 1** | **3** | **0** | **Balanced output** |
|  | **0.75** | **0 0 0** | **0** | **3** | **Harsh threshold cut all** |
| **3rd** | **0.25** | **1 1 1** | **3** | **0** | **Standard result at low threshold** |
|  | **0.50** | **1 0 1** | **2** | **1** | **Confidence dropped for 2nd student** |
|  | **0.75** | **1 0 0** | **1** | **2** | **High sensitivity at strict level** |
| **4th** | **0.25** | **1 1 1** | **3** | **0** | **Consistent with others** |
|  | **0.50** | **1 1 0** | **2** | **1** | **3rd student borderline** |
|  | **0.75** | **1 0 0** | **1** | **2** | **Only 1 strong admit** |
| **5th** | **0.25** | **1 1 1** | **3** | **0** | **Repeats trend** |
|  | **0.50** | **1 1 1** | **3** | **0** | **Strongly consistent again** |
|  | **0.75** | **1 0 1** | **2** | **1** | **One mid-range student rejected** |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Comparative Insights**

**Threshold 0.25: Every attempt predicted all students were admitted. The network demonstrated confidence by assigning each case at least a minimal probability, allowing every instance to surpass the lenient 25% threshold.**

**Threshold 0.50: Most test runs resulted in two or three admissions with slight variations based on initial weight settings and model generalization during each test session. Student 2 was often borderline.**

**Threshold 0.75: This threshold acted as a strict filter. Specific attempts produced no admissions, while others limited acceptance to one or two students. The high threshold effectively identified applicants with very strong admission indicators but sometimes left out candidates who demonstrated high potential.**

**Key Takeaways**

**The starting weights and threshold values influence neural network performance. Minor changes in model training can cause predictions to switch between admit and no-admit statuses near threshold boundaries.**

**Effective threshold adjustment is crucial in decision-making settings such as the college admissions process. Lower thresholds expand access, while higher thresholds restrict results to only strong predictions.**

**When tested multiple times with 0.25 and 0.50 thresholds, the model shows reliable performance. Outliers at the 0.75 threshold mark edge-case scenarios demand careful analysis during real-world application.**

**The middle test case (Student 2) is ideal for human review or model improvements due to its frequent threshold crossing across various attempts.**