NN Regression

JSM 2018: Poster 181 - Classroom Demonstration: Deep Learning for Classification and Regression, Introduction to GPU Computing

Eric A. Suess

Department Of Statistics and Biostatistics

CSU East Bay

eric.suess@csueastbay.edu

Example: Compare Simple Linear Regresion to a single layer NN.

The **cars** dataset in R contains two variables stopping *speed* of cars in mph and *dist* in feet. Using speed to predict stopping distance, two models are fit. See the R code.

a. What function is used to normalize the data?
b. What percentage of the data is used for *training*? What percentage of the data is used for *testing*?
c. What is the fitted linear regression model?
d. What is the correlation between the linear regression predicted values and the values from the test data?
e. Sketch the NN model that is used to model stopping distance.
f. What kind of activation function was used in the ANN? Sketch a picture of what the activation function looks like.
g. What is the correlation between the ANN predicted values and the values from the test data?
h. Examine the scatterplot of speed by distance with the fitted models. Is the NN fitting a near linear function?
i. Which model would you use for prediction? Explain.

Answer:

Read in data and examine structure.

```
suppressMessages(library("tidyverse"))
```

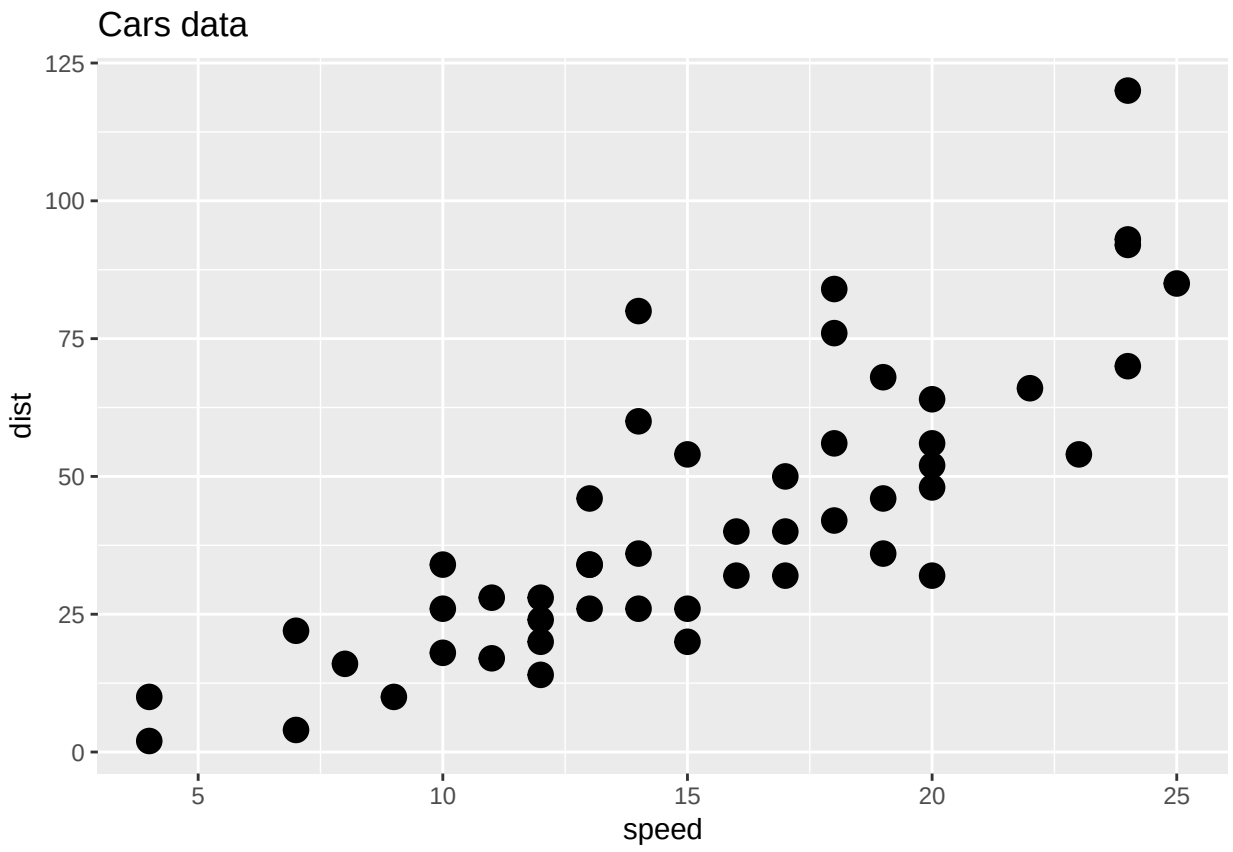
```
cars <- as.tibble(cars)
cars
```

```
## # A tibble: 50 x 2
##    speed  dist
##    <dbl> <dbl>
##  1     4     2
##  2     4    10
##  3     7     4
##  4     7    22
##  5     8    16
##  6     9    10
##  7    10    18
##  8    10    26
##  9    10    34
## 10    11    17
## # ... with 40 more rows
```

```
str(cars)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    50 obs. of  2 variables:
##  $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
##  $ dist : num  2 10 4 22 16 10 18 26 34 17 ...
```

```
cars %>% ggplot(aes(x=speed, y=dist)) +
  geom_point(size = 4) +
  ggtitle("Cars data")
```



Apply scaling to entire data frame.

```
cars_norm <- cars %>% mutate(speed = scale(speed), dist=scale(dist))
cars_norm
```

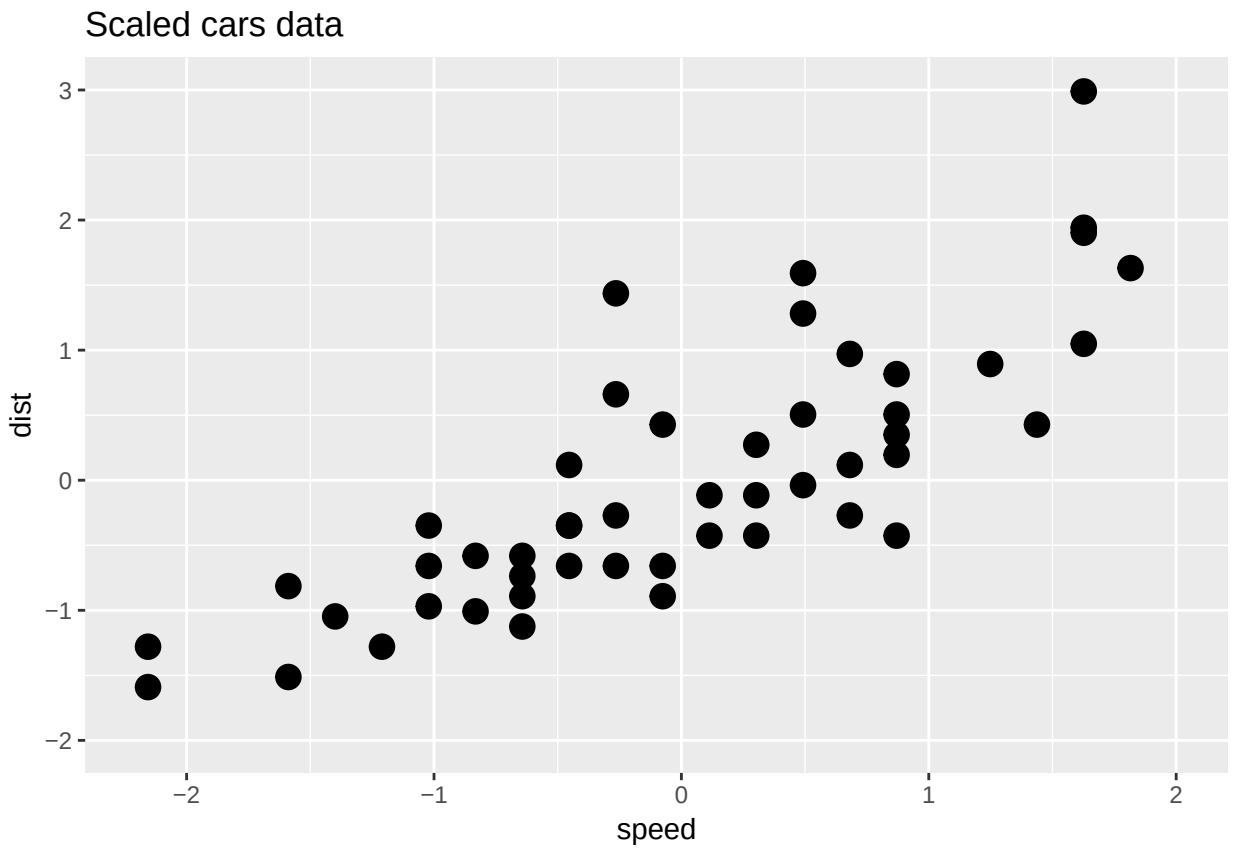
```
## # A tibble: 50 x 2
##     speed   dist
##     <dbl>  <dbl>
##  1 -2.16  -1.59
##  2 -2.16  -1.28
##  3 -1.59  -1.51
##  4 -1.59  -0.814
##  5 -1.40  -1.05
##  6 -1.21  -1.28
##  7 -1.02  -0.969
##  8 -1.02  -0.659
##  9 -1.02  -0.348
## 10 -0.832 -1.01
```

```
## # ... with 40 more rows
```

```
str(cars_norm)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    50 obs. of  2 variables:
##  $ speed: num [1:50, 1] -2.16 -2.16 -1.59 -1.59 -1.4 ...
##   ..- attr(*, "scaled:center")= num 15.4
##   ..- attr(*, "scaled:scale")= num 5.29
##  $ dist : num [1:50, 1] -1.59 -1.28 -1.513 -0.814 -1.047 ...
##   ..- attr(*, "scaled:center")= num 43
##   ..- attr(*, "scaled:scale")= num 25.8
```

```
cars_norm %>% ggplot(aes(x=speed, y=dist)) +
  geom_point(size = 4) +
  ggtitle("Scaled cars data") +
  scale_x_continuous(limits = c(-2.2, 2)) +
  scale_y_continuous(limits = c(-2, 3))
```



Create training and test data.

Side note: This is not done using best practices, the scale() function should only be applied to the training data not the entire dataset. This is a common practice in many machine learning books. This should be corrected.

```
set.seed(12345)

idx <- sample(1:50, 40)

cars_train <- cars_norm[idx, ]
```

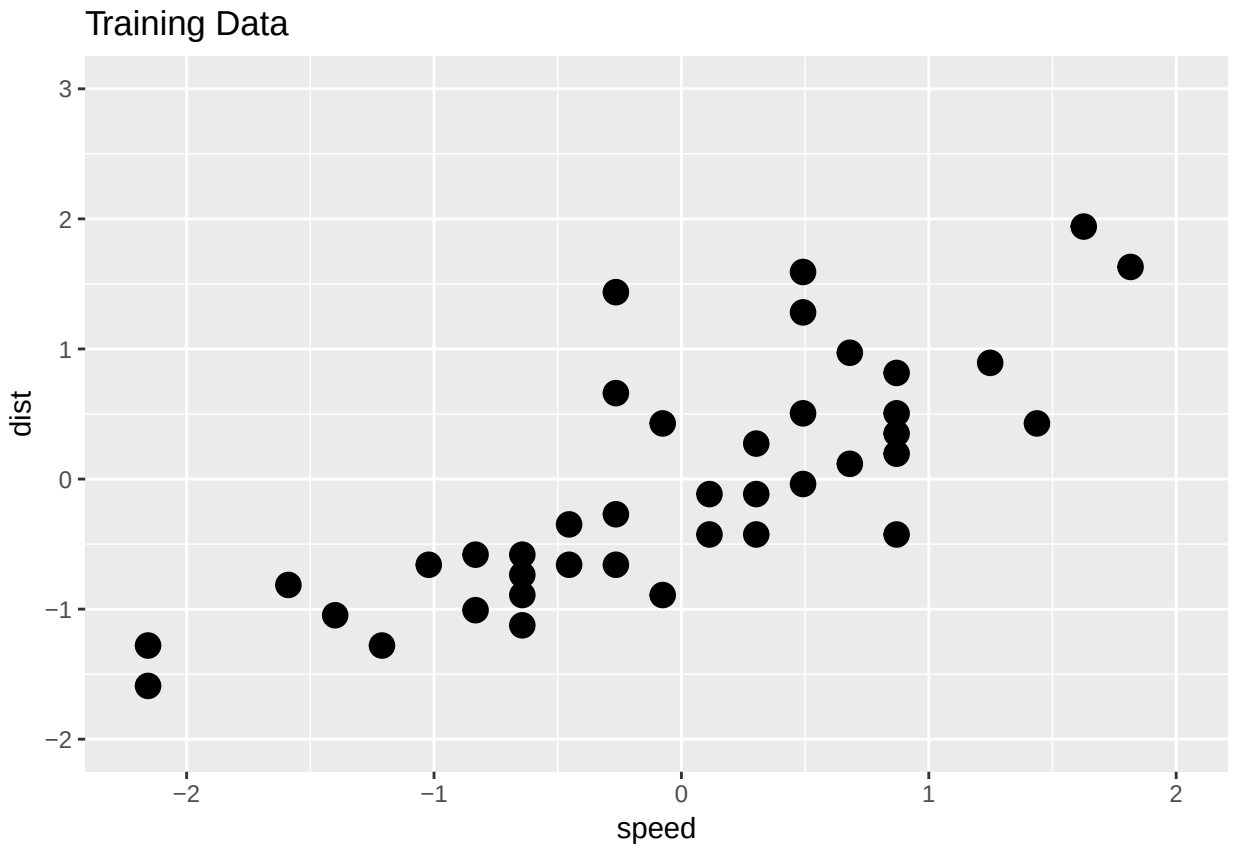
```
str(cars_train)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    40 obs. of  2 variables:
##  $ speed: num  0.681 0.87 1.816 0.87 -0.265 ...
##  $ dist : num  0.117 0.816 1.631 0.505 -0.271 ...
```

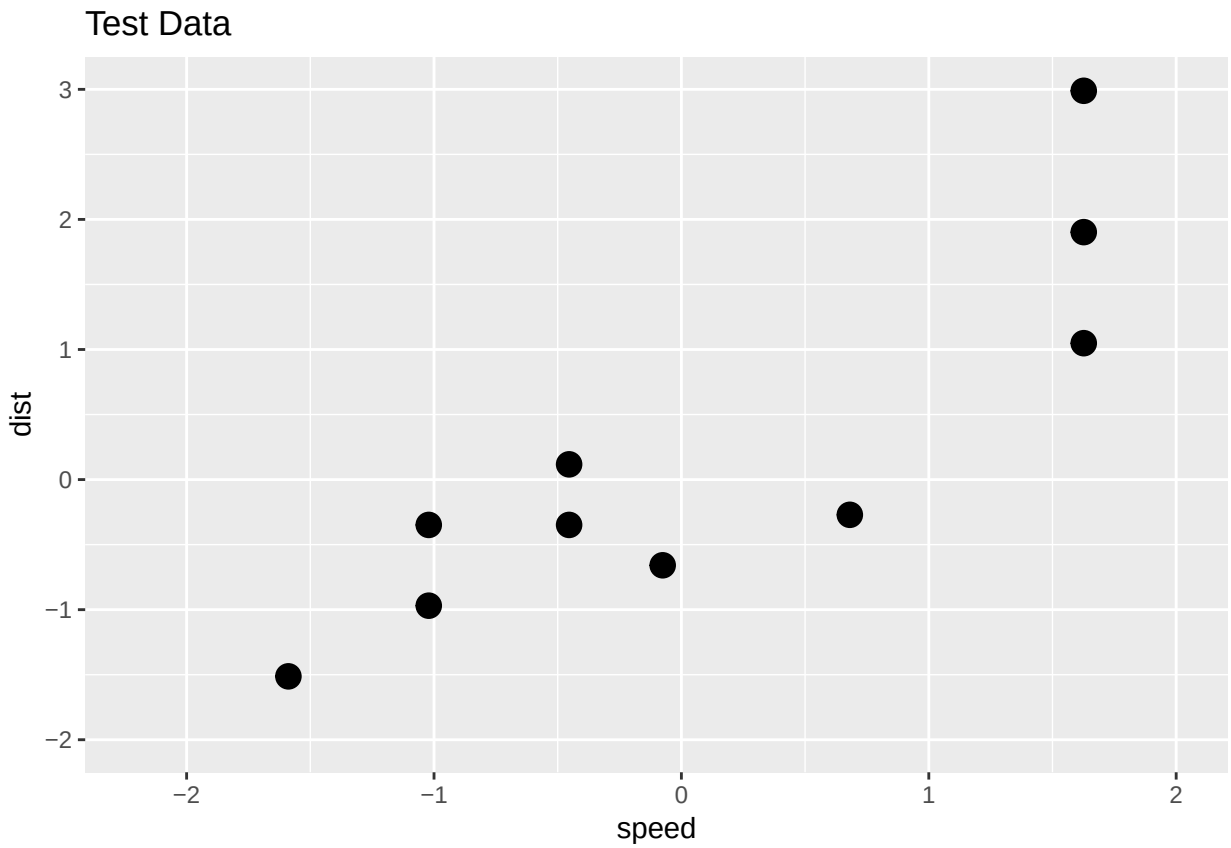
```
cars_test <- cars_norm[-idx, ]
str(cars_test)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    10 obs. of  2 variables:
##  $ speed: num  -1.589 -1.021 -1.021 -0.454 -0.454 ...
##  $ dist : num  -1.513 -0.969 -0.348 -0.348 0.117 ...
```

```
cars_train %>% ggplot(aes(x=speed, y=dist)) +
  geom_point(size = 4) +
  ggtitle("Training Data") +
  scale_x_continuous(limits = c(-2.2, 2)) +
  scale_y_continuous(limits = c(-2, 3))
```




```
cars_test %>% ggplot(aes(x=speed, y=dist)) +
  geom_point(size = 4) +
  ggtitle("Test Data") +
  scale_x_continuous(limits = c(-2.2, 2)) +
  scale_y_continuous(limits = c(-2, 3))
```



Fit a simple linear regression. Train a linear regression model. Predict the Test Data. Compare predicted values with the holdout values.

```
cars_lm <- cars_train %>% lm(dist ~ speed, data = .)

summary(cars_lm)
```

```
##
## Call:
## lm(formula = dist ~ speed, data = .)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.03373 -0.36619 -0.06137  0.23815  1.66240
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.03134    0.08919  -0.351    0.727
## speed        0.73450    0.09457   7.767 2.31e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5639 on 38 degrees of freedom
## Multiple R-squared:  0.6135, Adjusted R-squared:  0.6033
## F-statistic: 60.32 on 1 and 38 DF,  p-value: 2.315e-09
```

```
predicted_lm_dist <- predict(cars_lm, cars_test)
```

```r
# examine the correlation between predicted and actual values
cor(predicted_lm_dist, cars_test$dist)
```

```
## [1] 0.8696118
```

Fit a NN. Train a neural network model. Compare the R code. It is very similar.

```r
library(neuralnet)
```

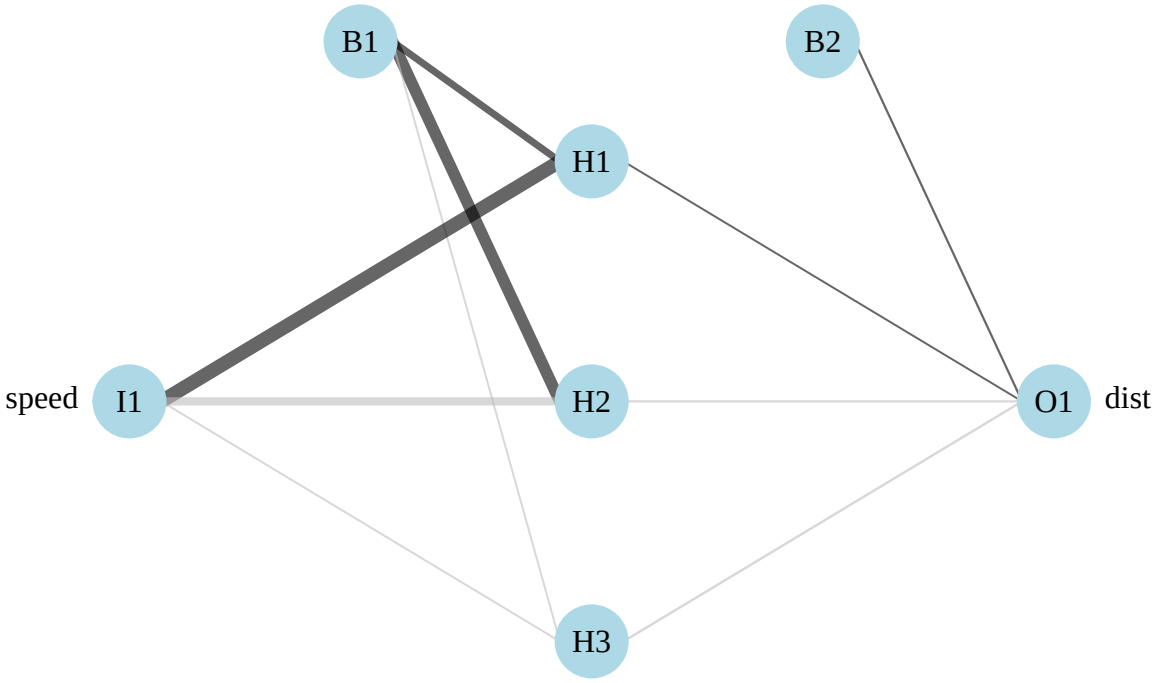
```
##
## Attaching package: 'neuralnet'
```

```
## The following object is masked from 'package:dplyr':
##
##     compute
```

```r
set.seed(12345)

cars_model <- cars_train %>% neuralnet(formula = dist ~ speed,
        act.fct = "logistic", hidden = 3, linear.output=TRUE)

plot(cars_model)
```

Nice plot with the plotnet() function.

```r
library(NeuralNetTools)

par(mar = numeric(4), family = 'serif')
plotnet(cars_model, alpha = 0.6)
```



Predict the Test Data. Compare predicted values with the holdout values.

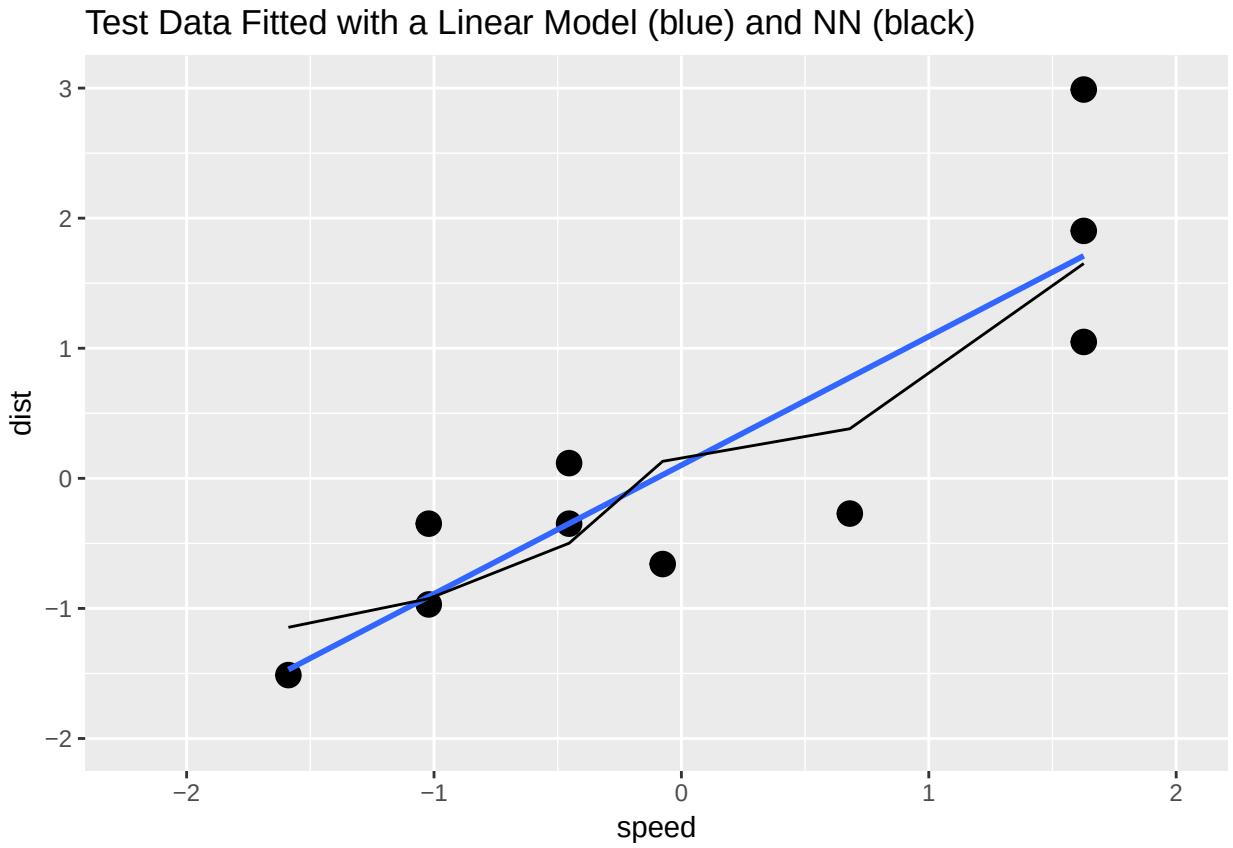
```r
model_results <- compute(cars_model, cars_test[1])

predicted_dist <- model_results$net.result
```

```r
# examine the correlation between predicted and actual values
cor(predicted_dist, cars_test$dist)
```

```
##              [,1]
## [1,] 0.8744086583
```

Plot the fitted models.

```r
ggplot(data=cars_test, aes(x=speed, y=dist)) +
  geom_point(size = 4) +
  geom_smooth(method='lm', formula=y~x, fill=NA) +
  geom_line(aes(y = predicted_dist)) +
  ggtitle("Test Data Fitted with a Linear Model (blue) and NN (black)") +
  scale_x_continuous(limits = c(-2.2, 2)) +
  scale_y_continuous(limits = c(-2, 3))
```



Example: Compare Simple Linear Regression to a Deep Learning, multilayer neural network.

a. Do you think this model will orverfit?

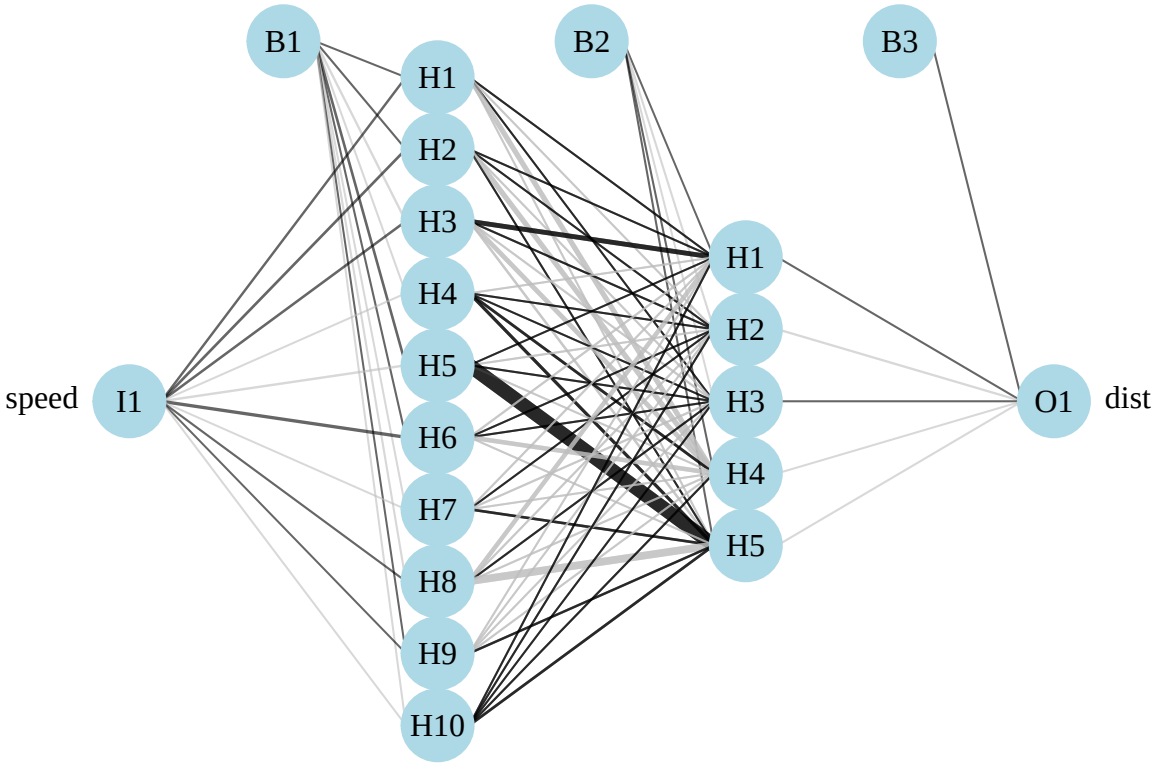
b. What does parsimonious mean?

c. Suggest a better measure for goodness-of-fit.

```
cars_model <- cars_train %>% neuralnet(formula = dist ~ speed,
        act.fct = "logistic", hidden = c(10,5), linear.output=TRUE)

plot(cars_model)
```

Nice plot with the plotnet() function.

```
par(mar = numeric(4), family = 'serif')
plotnet(cars_model, alpha = 0.6)
```



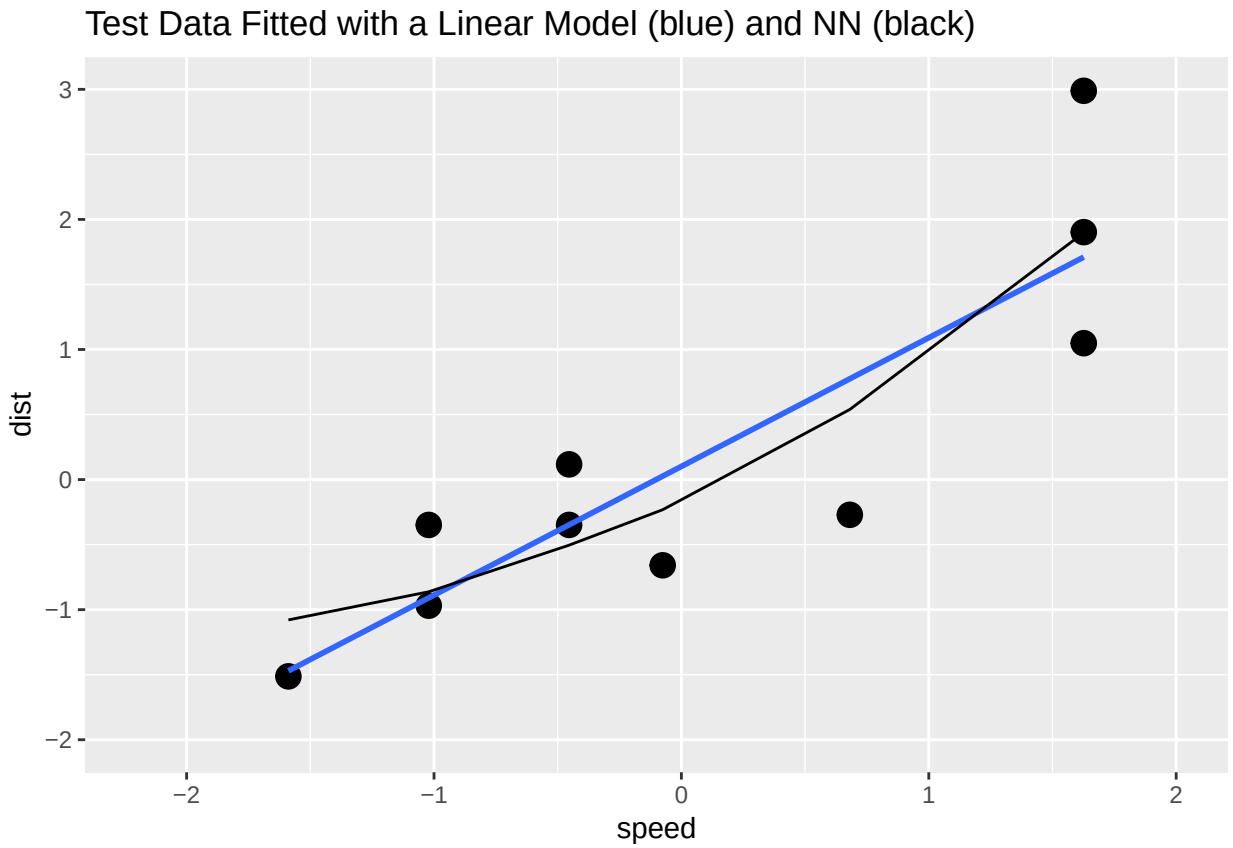
Predict the Test Data. Compare predicted values with the holdout values.

```
model_results <- compute(cars_model, cars_test[1])

predicted_dist <- model_results$net.result

# examine the correlation between predicted and actual values
cor(predicted_dist, cars_test$dist)
```

```
##              [,1]
## [1,] 0.889083417
```

Plot the fitted models.

```
ggplot(data=cars_test, aes(x=speed, y=dist)) +
  geom_point(size = 4) +
  geom_smooth(method='lm', formula=y~x, fill=NA) +
  geom_line(aes(y = predicted_dist)) +
  ggtitle("Test Data Fitted with a Linear Model (blue) and NN (black)") +
  scale_x_continuous(limits = c(-2.2, 2)) +
  scale_y_continuous(limits = c(-2, 3))
```


Test Data Fitted with a Linear Model (blue) and NN (black)
dist
speed
3
2
1
0
−1
−2
−2
−1
0
1
2