Week 5 Code Examples, CSC 400, Spring 2024

1. Neural Networks
    a. By purpose
        i. For classification
        ii. For regression
        iii. For time series forecasting
        iv. For anomaly detection
2. Visualization

Neural Networks

For Classification:

We start by installing and loading packages and loading in the data. We split the iris dataset into test and training sets.

```
library(tidyverse)
library(neuralnet)
iris <- iris %>% mutate_if(is.character, as.factor)
summary(iris)

set.seed(245)
data_rows <- floor(0.80 * nrow(iris))
train_indices <- sample(c(1:nrow(iris)), data_rows)
train_data <- iris[train_indices,]
test_data <- iris[-train_indices,]
```

Then we can run the neural network model and plot the best fit.

```
model = neuralnet(
  Species~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width,
  data=train_data,
  hidden=c(4,2),
  linear.output = FALSE
)

plot(model,rep = "best")
```

Now, we should test the model against the test set. Look at the confusion matrix.

```
pred <- predict(model, test_data)
labels <- c("setosa", "versicolor", "virginca")
prediction_label <- data.frame(max.col(pred)) %>%
  mutate(pred=labels[max.col.pred.]) %>%
  select(2) %>%
  unlist()

table(test_data$Species, prediction_label)
```

We can also check the accuracy of the model.

```
check = as.numeric(test_data$Species) == max.col(pred)
accuracy = (sum(check)/nrow(test_data))*100
print(accuracy)
```

Regression

We'll look at the Boston housing data for the regression application. Load the data and split into test and training data. We are also rescaling the variables for this process.

```
set.seed(500)
library(neuralnet)
library(MASS)
data <- Boston
maxs <- apply(data, 2, max)
mins <- apply(data, 2, min)
scaled <- as.data.frame(scale(data, center = mins,
                              scale = maxs - mins))
index <- sample(1:nrow(data), round(0.75 * nrow(data)))
train_ <- scaled[index,]
test_ <- scaled[-index,]
```

Create the model and check the results. We can also plot the model.

```
nn <- neuralnet(medv ~ crim + zn + indus + chas + nox
                + rm + age + dis + rad + tax +
                  ptratio + black + lstat,
                data = train_, hidden = c(5, 3),
                linear.output = TRUE)

pr.nn <- compute(nn, test_[,1:13])
pr.nn_ <- pr.nn$net.result * (max(data$medv) - min(data$medv)) + min(data$medv)
test.r <- (test_$medv) * (max(data$medv) - min(data$medv)) + min(data$medv)
MSE.nn <- sum((test.r - pr.nn_)^2) / nrow(test_)
plot(nn)
```

We can plot the regression line in the form real vs. predicted in the test (or train) set.

```
plot(test_$medv, pr.nn_, col = "red",
     main = 'Real vs Predicted')
scale<-(max(data$medv) - min(data$medv))
abline(min(data$medv),scale, lwd = 2)
```

Time Series Forecasting

We install and load our package, and we'll look at a time series to predict the future. Then we'll look at a plot.

```
library(tsfgrnn)
pred <- grnn_forecasting(UKgas, h = 4)
pred$prediction

plot(pred)
library(ggplot2)
autoplot(pred)

pred <- grnn_forecasting(times = 1:10, h = 2, lags = c(1, 3), msas = "MIMO",
                         transform = "none")
grnn_examples(pred)
```

We can examine other elements of our predictions.

```
grnn_weights(pred)
summary(pred)
plot_example(pred, 1)
plot_example(pred, 4)
```

There are several strategies for forecasting that we can apply.

```
predr <- grnn_forecasting(1:10, h = 2, lags = c(1, 3), msas = "recursive",
                          transform = "none")
predr$prediction
grnn_examples(predr)
plot_example(predr, position = 1, h = 1)
grnn_weights(predr)[[1]]
plot_example(predr, position = 1, h = 2)
grnn_weights(predr)[[2]]
```

Now, we need to assess our models.

```
pred <- grnn_forecasting(ts(1:20), h = 4, lags = 1:2)
ro <- rolling_origin(pred, h = 4)
print(ro$test_sets)
print(ro$predictions)
print(ro$errors)
ro$global_accu
ro$h_accu
plot(ro, h = 4)
ro <- rolling_origin(pred, h = 4, rolling = FALSE)
print(ro$test_sets)
print(ro$predictions)
```

How can we select the best model? Let's look at another short example.

```
pred <- grnn_forecasting(USAccDeaths, h = 12, lags = 1:12, sigma = 100)
plot(pred)
pred <- grnn_forecasting(USAccDeaths, h = 12, lags = 1:12, sigma = 0.05)
plot(pred)

set.seed(5)
times <- ts(1:10 + rnorm(10, 0, .2))
pred <- grnn_forecasting(times, h = 3, transform = "none")
plot(pred)

pred2 <- grnn_forecasting(times, h = 3, transform = "additive")
plot(pred2)
```

Which model best fits your data?

Anomaly Detection

Here's a small example for using a neural network for anomaly detection in the iris dataset.

```
library(ANN2)

random_draw <- sample(1:nrow(iris), size = 100)
X_train     <- iris[random_draw, 1:4]
y_train     <- iris[random_draw, 5]
X_test      <- iris[setdiff(1:nrow(iris), random_draw), 1:4]
y_test      <- iris[setdiff(1:nrow(iris), random_draw), 5]

NN <- neuralnetwork(X = X_train, y = y_train, hidden.layers = c(5, 5),
                    optim.type = 'adam', learn.rates = 0.01, val.prop = 0)

plot(NN)

y_pred <- predict(NN, newdata = X_test)

correct <- (y_test == y_pred$predictions)
plot(X_test, pch = as.numeric(y_test), col = correct + 2)
```

Resources:

1. https://www.datacamp.com/tutorial/neural-network-models-r
2. https://www.geeksforgeeks.org/how-neural-networks-are-used-for-regression-in-r-programming/#
3. https://cran.r-project.org/web/packages/tsfgrnn/vignettes/tsfgrnn.html
4. https://rdrr.io/cran/ANN2/man/neuralnetwork.html