STAT 882 Final

Katie Haring

Assignment

Build a neural network to classify using the attached dataset. It contains a structured dataset that was once used to train models to detect spam. Grading will be on the quality of your approach and the performance. By quality I mean the structure and clarity of your code. Still, don't over think this one, it isn't intended to be overly difficult. Measure the performance using mean accuracy across a four fold cross-validation. If you can get it to 0.94 you should consider yourself done.

If you can't get the performance that high, just do the best you can. Random seed should be set to 1. The response is in the 'type' column.

Data Import and Setup

```
library(data.table)
library(keras)
library(fastDummies)

setwd('C:/Users/Katie/Documents/Homework/Stat Learning II - Spr 20/Final')
data<-fread('spam.csv')
source('nested_cv.R')

#The keras package needs a data frame to build the neural net
data <- as.data.frame(data)

#Releveling ensures that we're modeling what we think we're modeling
data$type <- relevel(factor(data$type), ref='nonspam')
```

Model

This section contains the code for the final model. It utilizes: 3 layers, 3 sigmoid activation functions, the Adam optimizer, kernel constraints of 1.5 on the last 2 layers. **The model's cross-validated average Accuracy was 94.8%.** A description of how I got to this model can be found in the Methology section.

```
nn_mod = function(dat, response, params) {
  #Categorical variables must be converted to dummies to build a neural net with keras
  dat = dummy_cols(dat, remove_selected_columns = T, remove_first_dummy=T)

    response = paste0(response, '_spam')

    X = as.matrix(dat[,-which(colnames(dat)==response)])
    y = to_categorical(dat[,response])

    mod = keras_model_sequential()

    mod %>%
```

```
      layer_dense(units = 200, input_shape = ncol(X)) %>%
      layer_activation(activation = 'sigmoid') %>%

      #constraining the sum of squares of the weights
      layer_dense(units = 100, kernel_constraint = constraint_maxnorm(1.5)) %>%
      layer_activation(activation = 'sigmoid') %>%

      #constraining the sum of squares of the weights
      layer_dense(units = 2, kernel_constraint = constraint_maxnorm(1.5)) %>%
      layer_activation(activation = 'sigmoid')

    mod %>%
      compile(loss = 'binary_crossentropy',
              optimizer = 'Adam',
              metrics = c('accuracy'))

    mod %>% fit(X, y, epochs = 200, batch_size = 10)

    return(mod)
  }


nn_outer_perf = function(mod, dat, response) {
  dat = dummy_cols(dat, remove_selected_columns = T, remove_first_dummy=T)
  response = paste0(response, '_spam')

  X = as.matrix(dat[,-which(colnames(dat)==response)])
  y = dat[,response]

  preds = predict(mod, X)

  preds = ifelse(preds[,2] > .5, 1, 0)

  return(Accuracy(y, preds))
}


nn_score = function(mod, dat, response) {
  dat = dummy_cols(dat, remove_selected_columns = T, remove_first_dummy=T)
  response = paste0(response, '_spam')

  X = as.matrix(dat[,-which(colnames(dat)==response)])
  y = dat[,response]

  return(predict(mod, X)[,2])
}


res_nn = nested_cv(cv_k1 = 4, seed = 1, model = nn_mod, outer_perf_f = nn_outer_perf,
                   score_f = nn_score, dat = data, response = 'type')

#Mean Accuracy is 94.76%
mean(res_nn$Performance[,1])
```

Methodology

Throughout modeling, I kept epochs = 200 and batch_size = 10, and used binary cross entropy for the loss function. I considered reducing epochs to 100, as the textbook says that a neural network model will typically use between 5 and 100 hidden units, but the accuracy noticeably improved between 100 and 200 for at least some of the model variations that I tried, so I ultimately decided against it.

Just to start, I built two models that contained a single hidden layer and used stochastic gradient descent for optimization- one using the hyperbolic tangent activation function and one using the sigmoid activation function. They both performed poorly (Accuracy roughly equal to 60% and 66%, respectively).

To give my model more flexibility, I switched to using 3 layers. I used the sigmoid activation at all 3 layers, since the sigmoid had outperformed the hyperbolic tangent in the one layer model. This model was actually worse than the one layer sigmoid model (Accuracy around 60%).

Next, I researched previous neural networks that had modeled whether or not an email was spam and found a paper testing several different optimizers in a neural network designed for this purpose. The ones using the Adam optimizer performed best. Researching the Adam optimizer, it claimed to be good for large data sets or data sets with a large number of parameters, as well as noisy data sets. It also claimed to work well with little to no tuning. I switched to using the Adam optimizer (and left all other hyperparameters and model settings the same), and the Accuracy jumped to 93.7%.

Then, I tried a few different methods of improving my model. I tried adding class weights (1 for nonspam, 5 for spam), since the data contained more nonspam than spam and neural networks tend to perform best on the most common class. This was ineffective, with an Accuracy of 92.7% (1% less than the Accuracy of the model that didn't use class weights). Consequently, I removed the class weights and tried using l1 regularization (hyperparameter = 0.003) to zero out the poorest nodes; that bumped the Accuracy to 94.1%.

Finally, I removed the regularizer and tried constraining the sum of squares of the weights for each of the last 2 layers. I tried 3 levels of constraint (1, 1.5, and 2), using the same level of constraint for both layers. All were slightly better than 94.1%, but the model with constraints = 1.5 was the best and, therefore, became my final model. The Accuracy was 94.8%. Since that was over the 94% cut-off and I had tested multiple methods, I stopped there. Given the randomness inherent to artificial neural networks, any one of the last several models I built could probably have been my final model. If I wanted to improve upon this model, I could try combining some of the methods I tested, adding/removing layers, tuning the optimizer, etc.

Code for All Tested Models.

This section contains the model code for every model I tested, in case you want to see them all.

```
#Single layer tanh activation w/ sgd optimizer - Accuracy around 60%
nn_mod = function(dat, response, params) {

    dat = dummy_cols(dat, remove_selected_columns = T, remove_first_dummy=T)
    response = paste0(response, '_spam')

    X = as.matrix(dat[,-which(colnames(dat)==response)])
    y = to_categorical(dat[,response])

    mod = keras_model_sequential()

    mod %>%
      layer_dense(units = 2, input_shape = ncol(X)) %>%
      layer_activation(activation = 'tanh')

    mod %>%
      compile(loss = 'binary_crossentropy',
              optimizer = 'sgd',
              metrics = c('accuracy'))

    mod %>% fit(X, y, epochs = 200, batch_size = 10)

    return(mod)
    }




#Single layer sigmoid activation w/ sgd optimizer - Accuracy around 66%
    nn_mod = function(dat, response, params) {

    dat = dummy_cols(dat, remove_selected_columns = T, remove_first_dummy=T)
    response = paste0(response, '_spam')

    X = as.matrix(dat[,-which(colnames(dat)==response)])
    y = to_categorical(dat[,response])

    mod = keras_model_sequential()

    mod %>%
      layer_dense(units = 2, input_shape = ncol(X)) %>%
      layer_activation(activation = 'sigmoid')

    mod %>%
      compile(loss = 'binary_crossentropy',
              optimizer = 'sgd',
              metrics = c('accuracy'))

    mod %>% fit(X, y, epochs = 200, batch_size = 10)

    return(mod)
```

```
    }



#3 layer w/ triple sigmoid activation and sgd optimizer - Accuracy around 60%
    nn_mod = function(dat, response, params) {

    dat = dummy_cols(dat, remove_selected_columns = T, remove_first_dummy=T)
    response = paste0(response, '_spam')

    X = as.matrix(dat[,-which(colnames(dat)==response)])
    y = to_categorical(dat[,response])

    mod = keras_model_sequential()

    mod %>%
      layer_dense(units = 200, input_shape = ncol(X)) %>%
      layer_activation(activation = 'sigmoid') %>%
      layer_dense(units = 100) %>%
      layer_activation(activation = 'sigmoid') %>%
      layer_dense(units = 2) %>%
      layer_activation(activation = 'sigmoid')

    mod %>%
      compile(loss = 'binary_crossentropy',
              optimizer = 'sgd',
              metrics = c('accuracy'))

    mod %>% fit(X, y, epochs = 200, batch_size = 10)

    return(mod)
    }




#3 layer w/ triple sigmoid activation and Adam optimizer - Accuracy = 93.7%
    nn_mod = function(dat, response, params) {

    dat = dummy_cols(dat, remove_selected_columns = T, remove_first_dummy=T)
    response = paste0(response, '_spam')

    X = as.matrix(dat[,-which(colnames(dat)==response)])
    y = to_categorical(dat[,response])

    mod = keras_model_sequential()

    mod %>%
      layer_dense(units = 200, input_shape = ncol(X)) %>%
      layer_activation(activation = 'sigmoid') %>%
      layer_dense(units = 100,) %>%
      layer_activation(activation = 'sigmoid') %>%
      layer_dense(units = 2,)) %>%
```

```
  layer_activation(activation = 'sigmoid')

mod %>%
  compile(loss = 'binary_crossentropy',
          optimizer = 'Adam',
          metrics = c('accuracy'))

mod %>% fit(X, y, epochs = 200, batch_size = 10)

return(mod)
}
```

```
#3 layer w/ triple sigmoid activation, Adam optimizer, and class weights - Accuracy = 92.7%
nn_mod = function(dat, response, params) {

dat = dummy_cols(dat, remove_selected_columns = T, remove_first_dummy=T)
response = paste0(response, '_spam')

X = as.matrix(dat[,-which(colnames(dat)==response)])
y = to_categorical(dat[,response])

mod = keras_model_sequential()

mod %>%
  layer_dense(units = 200, input_shape = ncol(X)) %>%
  layer_activation(activation = 'sigmoid') %>%
  layer_dense(units = 100) %>%
  layer_activation(activation = 'sigmoid') %>%
  layer_dense(units = 2) %>%
  layer_activation(activation = 'sigmoid')

mod %>%
  compile(loss = 'binary_crossentropy',
          optimizer = 'Adam',
          metrics = c('accuracy'))

class_weight = list('0' = 1., '1' = 5.)

mod %>% fit(X, y, epochs = 200, batch_size = 10, class_weight=class_weight)

return(mod)
}
```

```
#3 layer w/ triple sigmoid activation, Adam optimizer, and l1 regularizer - Accuracy = 94.1%
nn_mod = function(dat, response, params) {

dat = dummy_cols(dat, remove_selected_columns = T, remove_first_dummy=T)
```

```
    response = paste0(response, '_spam')

    X = as.matrix(dat[,-which(colnames(dat)==response)])
    y = to_categorical(dat[,response])

    mod = keras_model_sequential()

    mod %>%
      layer_dense(units = 200, input_shape = ncol(X)) %>%
      layer_activation(activation = 'sigmoid') %>%
      layer_dense(units = 100, kernel_regularizer = regularizer_l1(0.003)) %>%
      layer_activation(activation = 'sigmoid') %>%
      layer_dense(units = 2, kernel_regularizer = regularizer_l1(0.003)) %>%
      layer_activation(activation = 'sigmoid')

    mod %>%
      compile(loss = 'binary_crossentropy',
              optimizer = 'Adam',
              metrics = c('accuracy'))

    mod %>% fit(X, y, epochs = 200, batch_size = 10)

    return(mod)
    }




#3 layer w/ triple sigmoid activation, Adam optimizer, and constrained weights (max = 2) -
  #Accuracy = 94.3%
    nn_mod = function(dat, response, params) {

    dat = dummy_cols(dat, remove_selected_columns = T, remove_first_dummy=T)
    response = paste0(response, '_spam')

    X = as.matrix(dat[,-which(colnames(dat)==response)])
    y = to_categorical(dat[,response])

    mod = keras_model_sequential()

    mod %>%
      layer_dense(units = 200, input_shape = ncol(X)) %>%
      layer_activation(activation = 'sigmoid') %>%
      layer_dense(units = 100, kernel_constraint = constraint_maxnorm(2)) %>%
      layer_activation(activation = 'sigmoid') %>%
      layer_dense(units = 2, kernel_constraint = constraint_maxnorm(2)) %>%
      layer_activation(activation = 'sigmoid')

    mod %>%
      compile(loss = 'binary_crossentropy',
              optimizer = 'Adam',
              metrics = c('accuracy'))
```

```
    mod %>% fit(X, y, epochs = 200, batch_size = 10)

    return(mod)
    }



#3 layer w/ triple sigmoid activation, Adam optimizer, and constrained weights (max = 1.5) -
  #Accuracy = 94.8%
    nn_mod = function(dat, response, params) {

    dat = dummy_cols(dat, remove_selected_columns = T, remove_first_dummy=T)
    response = paste0(response, '_spam')

    X = as.matrix(dat[,-which(colnames(dat)==response)])
    y = to_categorical(dat[,response])

    mod = keras_model_sequential()

    mod %>%
      layer_dense(units = 200, input_shape = ncol(X)) %>%
      layer_activation(activation = 'sigmoid') %>%
      layer_dense(units = 100, kernel_constraint = constraint_maxnorm(1.5)) %>%
      layer_activation(activation = 'sigmoid') %>%
      layer_dense(units = 2, kernel_constraint = constraint_maxnorm(1.5)) %>%
      layer_activation(activation = 'sigmoid')

    mod %>%
      compile(loss = 'binary_crossentropy',
              optimizer = 'Adam',
              metrics = c('accuracy'))

    mod %>% fit(X, y, epochs = 200, batch_size = 10)

    return(mod)
    }



#3 layer w/ triple sigmoid activation, Adam optimizer, and constrained weights (max = 1) -
  #Accuracy = 94.6%
    nn_mod = function(dat, response, params) {

    dat = dummy_cols(dat, remove_selected_columns = T, remove_first_dummy=T)
    response = paste0(response, '_spam')

    X = as.matrix(dat[,-which(colnames(dat)==response)])
    y = to_categorical(dat[,response])

    mod = keras_model_sequential()

    mod %>%
      layer_dense(units = 200, input_shape = ncol(X)) %>%
```

```
  layer_activation(activation = 'sigmoid') %>%
  layer_dense(units = 100, kernel_constraint = constraint_maxnorm(1)) %>%
  layer_activation(activation = 'sigmoid') %>%
  layer_dense(units = 2, kernel_constraint = constraint_maxnorm(1)) %>%
  layer_activation(activation = 'sigmoid')

mod %>%
  compile(loss = 'binary_crossentropy',
          optimizer = 'Adam',
          metrics = c('accuracy'))

mod %>% fit(X, y, epochs = 200, batch_size = 10)

return(mod)
}
```