HW13

Zhuo Chen

2018/12/10

Preparation

TRAIN DATASET PART

get the training datasets

```r
if (!exists("mtrain")) {
  mtrain <- read.csv("mnist_train.csv", header=F) %>% as.matrix
  train_classification <- mtrain[,1]  # y value

  mtrain <- mtrain[,-1]/256  # x matrix
  colnames(mtrain) <- 1:(28^2)
  x <- mtrain[1:1000, ]

  rownames(mtrain) <- NULL
}

y <- rep(NA, length(train_classification))
```

Changing all 3s to 1 and all other numbers to 0

```r
for (i in 1:length(train_classification)){
  i_th <- train_classification[i]

  if (i_th==3){
      i_th <- 1
    } else {
      i_th <- 0
    }
  y[i] <- i_th
}

# for caret, y variable should be a factor
# see line 54 in caret_intro_2d.R
y <- factor(y, levels=c(0,1))
y <- y[1:1000]
```

TEST DATASET PART

getting the test data set (Mainly the same as above)

```r
if (!exists("mtrain_t")) {
  mtrain_t <- read.csv("mnist_test.csv", header=F) %>% as.matrix
  train_classification_t <- mtrain_t[,1]  # y value

  mtrain_t <- mtrain_t[,-1]/256  # x matrix
  colnames(mtrain_t) <- 1:(28^2)
  x_t <- mtrain_t[1:1000, ]

  rownames(mtrain_t) <- NULL
}

y_t <- rep(NA, length(train_classification_t))

#Changing all 3s to 1 and all other numbers to 0
for (i in 1:length(train_classification_t)){
  i_th <- train_classification_t[i]

  if (i_th==3){
    i_th <- 1
  } else {
    i_th <- 0
  }
  y_t[i] <- i_th
}

y_t <- factor(y_t, levels=c(0,1))
y_t <- y_t[1:1000]

#Sample
head(y)
```

```
## [1] 0 0 0 0 0 0
## Levels: 0 1
```

```r
head(y_t)
```

```
## [1] 0 0 0 0 0 0
## Levels: 0 1
```

NEURAL NET PART

let's try and fit with a neural net.

With Fixed decay = 0

The process of training

```
#Use size=1 as an example display
tuning_df <- data.frame(size=1, decay=0)
fitControl <- trainControl(## 2-fold CV
  method = "repeatedcv",
  number = 2,
  repeats = 2)
t_out <- caret::train(x=x, y=y, method="nnet",
                      trControl = fitControl,
                      tuneGrid=tuning_df, maxit=1000, MaxNWts=10000)
```

```
## # weights:  787
## initial  value 325.036075
## iter  10 value 79.463635
## iter  20 value 44.677850
## iter  30 value 22.079351
## iter  40 value 18.090139
## iter  50 value 18.062925
## iter  60 value 18.061785
## final  value 18.061783
## converged
## # weights:  787
## initial  value 252.977357
## iter  10 value 58.956603
## iter  20 value 32.308144
## iter  30 value 28.975411
## iter  40 value 17.076083
## iter  50 value 15.401683
## iter  60 value 15.387410
## iter  70 value 15.386215
## iter  80 value 15.385796
## iter  90 value 12.004469
## iter 100 value 11.935708
## iter 110 value 8.336520
## iter 120 value 7.174298
## iter 130 value 7.119810
## iter 140 value 7.119357
## iter 150 value 7.119220
## iter 160 value 7.119206
## final  value 7.119202
## converged
## # weights:  787
## initial  value 432.968325
## final  value 155.946591
## converged
## # weights:  787
## initial  value 404.126660
## iter  10 value 153.468302
## iter  20 value 92.396965
## iter  30 value 63.641888
## iter  40 value 51.870133
## iter  50 value 44.365885
## iter  60 value 44.357740
## iter  70 value 44.357495
## iter  80 value 44.357006
## iter  80 value 44.357006
## iter  80 value 44.357005
## final  value 44.357005
## converged
## # weights:  787
## initial  value 707.400608
## iter  10 value 309.430604
## iter  20 value 309.424337
## final  value 309.424331
## converged
```

Write a predicting-errors function first

```
predicting_errors <- function(y, t_out)
{
  true_y <- y
  predict_y <- predict(t_out, x)
  n_samples <- nrow(x)
  error <- sum(true_y != predict_y)/n_samples
  return(error)
}
```

Predicting the errors

```
predict_error <- predicting_errors(y, t_out)
cat("train prediction error", predict_error, "\n")
```

```
## train prediction error 0.093
```

To find the best size, we can use a for loop

```
error_list <- rep(NA, 5)
for (i in 1:5) {
  tuning_df <- data.frame(size=i, decay=0)
  fitControl <- trainControl(## 2-fold CV
    method = "repeatedcv",
    number = 2,
    repeats = 2)
  t_out <- caret::train(x=x, y=y, method="nnet",
                        trControl = fitControl,
                        tuneGrid=tuning_df, maxit=1000, MaxNWts=10000)
  predict_error <- predicting_errors(y, t_out)
  error_list[i] <- predict_error

}
```

```
## # weights:  787
## initial  value 344.288729
## iter  10 value 58.668609
## iter  20 value 32.158364
## iter  30 value 19.887756
## iter  40 value 12.007103
## iter  50 value 11.960552
## iter  60 value 7.166293
## iter  70 value 7.120259
## iter  80 value 7.119617
## iter  90 value 7.119310
## iter 100 value 7.119227
## iter 110 value 7.119216
## iter 110 value 7.119216
## iter 110 value 7.119216
## final  value 7.119216
## converged
## # weights:  787
## initial  value 281.239448
## iter  10 value 78.547439
## iter  20 value 51.088788
## iter  30 value 40.095125
## iter  40 value 35.600307
## iter  50 value 31.302423
## iter  60 value 25.932931
## iter  70 value 21.110406
## iter  80 value 21.070543
## iter  90 value 21.068712
## iter 100 value 21.068158
## iter 110 value 21.064350
## iter 120 value 17.644018
## iter 130 value 17.641436
## iter 140 value 17.549589
## iter 150 value 11.934443
## iter 160 value 11.932864
## iter 170 value 11.932722
## iter 180 value 11.932653
## iter 190 value 11.932544
## final  value 11.932541
## converged
## # weights:  787
## initial  value 427.690485
## iter  10 value 74.649258
## iter  20 value 38.486143
## iter  30 value 36.276381
## iter  40 value 36.259089
## iter  50 value 36.259002
## final  value 36.258999
## converged
## # weights:  787
## initial  value 360.291604
## iter  10 value 153.473018
## iter  20 value 153.424562
## iter  30 value 153.377108
## final  value 153.182763
## converged
## # weights:  787
## initial  value 608.076112
## iter  10 value 301.983670
## iter  20 value 128.797660
## iter  30 value 69.817190
## iter  40 value 60.061232
## iter  50 value 60.041620
## final  value 60.040425
## converged
## # weights:  1573
## initial  value 231.407035
## iter  10 value 42.721698
## iter  20 value 30.360771
## iter  30 value 26.826463
## iter  40 value 22.988427
## iter  50 value 22.920040
## iter  60 value 22.918623
## iter  70 value 22.918406
## iter  70 value 22.918406
## iter  70 value 22.918406
## final  value 22.918406
## converged
## # weights:  1573
## initial  value 290.163483
## iter  10 value 68.521276
## iter  20 value 31.009758
## iter  30 value 18.231137
## iter  40 value 18.060724
## iter  50 value 17.878081
## iter  60 value 15.871776
## iter  70 value 4.339709
## iter  80 value 0.390550
## iter  90 value 0.075574
## iter 100 value 0.005212
## iter 110 value 0.000353
## final  value 0.000024
## converged
## # weights:  1573
## initial  value 372.545050
## iter  10 value 33.844135
## iter  20 value 17.757348
## iter  30 value 17.681409
## iter  40 value 17.669866
## iter  50 value 13.113782
## iter  60 value 12.851550
## iter  70 value 12.850244
## iter  80 value 12.850075
## iter  90 value 12.850028
## final  value 12.850028
## converged
## # weights:  1573
## initial  value 309.220302
## iter  10 value 86.822076
## iter  20 value 48.310964
## iter  30 value 31.024696
## iter  40 value 22.907855
## iter  50 value 20.275002
## iter  60 value 20.255935
## iter  70 value 20.229949
## iter  80 value 20.229291
## iter  90 value 20.229175
## iter 100 value 14.973988
## iter 110 value 13.641175
## iter 120 value 13.639227
## iter 120 value 13.639227
## iter 120 value 13.639227
## final  value 13.639227
## converged
## # weights:  1573
## initial  value 895.431547
## final  value 309.424310
## converged
## # weights:  2359
## initial  value 269.308281
## iter  10 value 22.725188
## iter  20 value 11.996885
## iter  30 value 7.467137
## iter  40 value 3.398841
## iter  50 value 2.269632
## iter  60 value 2.256688
## iter  70 value 2.250306
## iter  80 value 2.249466
## iter  90 value 2.249404
## iter 100 value 2.249398
## final  value 2.249382
## converged
## # weights:  2359
## initial  value 525.451591
## final  value 155.847978
## converged
## # weights:  2359
## initial  value 539.180267
## iter  10 value 147.015990
## iter  20 value 51.890136
## iter  30 value 30.544427
## iter  40 value 13.999915
## iter  50 value 7.021283
## iter  60 value 6.217522
## iter  70 value 5.401322
## iter  80 value 5.293616
## iter  90 value 5.292835
## iter 100 value 4.190042
## iter 110 value 4.188134
## iter 120 value 4.188087
## iter 130 value 4.188013
## iter 130 value 4.188013
## iter 130 value 4.188013
## final  value 4.188013
## converged
## # weights:  2359
## initial  value 252.142471
## iter  10 value 46.521453
## iter  20 value 23.537784
## iter  30 value 20.968830
## iter  40 value 20.964182
## iter  50 value 20.962259
## iter  60 value 20.961532
## iter  70 value 20.960129
## iter  80 value 20.729080
## iter  90 value 17.391166
## iter 100 value 12.741648
## iter 110 value 3.748975
## iter 120 value 0.014709
## iter 130 value 0.000202
## final  value 0.000088
## converged
## # weights:  2359
## initial  value 526.755065
## iter  10 value 290.618022
## iter  20 value 208.945010
## iter  30 value 82.904421
## iter  40 value 48.671897
## iter  50 value 38.500292
## iter  60 value 28.855870
## iter  70 value 25.447153
## iter  80 value 23.485542
## iter  90 value 21.945189
## iter 100 value 20.111701
## iter 110 value 17.839214
## iter 120 value 17.837294
## iter 130 value 17.836863
## iter 140 value 17.836644
## iter 150 value 17.649933
## iter 160 value 17.386816
## iter 170 value 13.300628
## iter 180 value 10.955319
## iter 190 value 10.950103
## iter 200 value 10.949387
## iter 210 value 10.949250
## iter 220 value 10.949181
## iter 230 value 10.949135
## final  value 10.949134
## converged
## # weights:  3145
## initial  value 325.468729
## iter  10 value 76.189229
## iter  20 value 25.374124
## iter  30 value 5.236931
## iter  40 value 3.442890
## iter  50 value 0.739080
## iter  60 value 0.013992
## iter  70 value 0.005237
## iter  80 value 0.000846
## final  value 0.000019
## converged
## # weights:  3145
## initial  value 316.666621
## iter  10 value 69.665886
## iter  20 value 29.282398
## iter  30 value 28.358849
## iter  40 value 23.109364
## iter  50 value 22.834195
## iter  60 value 18.167104
## iter  70 value 18.068905
## iter  80 value 18.068431
## final  value 18.068357
## converged
## # weights:  3145
## initial  value 223.747707
## iter  10 value 25.072500
## iter  20 value 6.456358
## iter  30 value 0.193436
## iter  40 value 0.002463
## iter  50 value 0.000124
## final  value 0.000099
## converged
## # weights:  3145
## initial  value 465.227673
## iter  10 value 71.889837
## iter  20 value 32.203960
## iter  30 value 31.998146
## iter  40 value 26.774419
## iter  50 value 0.122664
## iter  60 value 0.001474
## final  value 0.000089
## converged
## # weights:  3145
## initial  value 1042.172751
## iter  10 value 287.259185
## iter  20 value 92.970909
## iter  30 value 34.869441
## iter  40 value 33.486423
## iter  50 value 29.901555
## iter  60 value 29.414761
## iter  70 value 20.247637
## iter  80 value 14.521960
## iter  90 value 13.337685
## iter 100 value 11.520981
## iter 110 value 11.303628
## iter 120 value 10.898729
## iter 130 value 10.830086
## iter 140 value 10.761817
## iter 150 value 10.643480
## iter 160 value 10.559498
## iter 170 value 10.511234
## iter 180 value 10.367839
## iter 190 value 10.363858
## iter 200 value 10.363682
## iter 210 value 10.362978
## iter 220 value 10.339376
## iter 230 value 10.308067
## iter 240 value 10.307549
## iter 250 value 10.173770
## iter 260 value 9.971684
## iter 270 value 9.971052
## iter 280 value 9.963826
## iter 290 value 9.912498
## iter 300 value 9.912083
## final  value 9.912055
## converged
## # weights:  3931
## initial  value 329.284030
## iter  10 value 58.695245
## iter  20 value 36.785849
## iter  30 value 32.247597
## iter  40 value 31.979858
## iter  50 value 23.185447
## iter  60 value 11.360953
## iter  70 value 3.224530
## iter  80 value 0.198306
## iter  90 value 0.049792
## iter 100 value 0.008134
## iter 110 value 0.001202
## iter 120 value 0.000184
## final  value 0.000081
## converged
## # weights:  3931
## initial  value 364.690776
## iter  10 value 54.394760
## iter  20 value 27.056071
## iter  30 value 18.110220
## iter  40 value 17.902572
## iter  50 value 7.125687
## iter  60 value 4.821581
## iter  70 value 3.414842
## iter  80 value 2.293540
## iter  90 value 2.092588
## iter 100 value 0.017565
## iter 110 value 0.007439
## iter 120 value 0.003817
## iter 130 value 0.002456
## iter 140 value 0.000426
## iter 150 value 0.000426
## iter 160 value 0.000425
## final  value 0.000425
## converged
## # weights:  3931
## initial  value 214.442567
## iter  10 value 45.289557
## iter  20 value 11.761583
## iter  30 value 3.851215
## iter  40 value 0.155748
## iter  50 value 0.002312
## iter  60 value 0.000130
## final  value 0.000089
## converged
## # weights:  3931
## initial  value 628.712535
## iter  10 value 89.111406
## iter  20 value 28.557186
## iter  30 value 23.355668
## iter  40 value 22.916259
## iter  50 value 22.486649
## iter  60 value 19.320538
## iter  70 value 18.796103
## iter  80 value 9.889086
## iter  90 value 2.089615
## iter 100 value 1.658374
## iter 110 value 1.402228
## iter 120 value 0.046647
## iter 130 value 0.002558
## iter 140 value 0.001341
## iter 150 value 0.000288
## iter 160 value 0.000231
## iter 170 value 0.000230
## final  value 0.000230
## converged
## # weights:  3931
## initial  value 433.531379
## iter  10 value 29.326560
## iter  20 value 2.689516
## iter  30 value 0.041825
## iter  40 value 0.003259
## iter  50 value 0.000426
## iter  60 value 0.000269
## final  value 0.000095
## converged
```

```
error_list
```

```
## [1] 0.011 0.093 0.002 0.004 0.000
```

We decide to use size=5, under which error=0, a good enough nod

With Varied decay

The process of training

```
#Use size=1, decay=0.1 as an example display
tuning_df <- data.frame(size=1, decay=.1)
fitControl <- trainControl(method="none")
fitControl <- trainControl(## 2-fold CV
  method = "repeatedcv",
  number = 2,
  repeats = 2)
t_out_1 <- caret::train(x=x, y=y, method="nnet",
                      trControl = fitControl,
                      tuneGrid=tuning_df, maxit=1000, MaxNWts=10000)
```

```
## # weights:  787
## initial  value 501.740245
## iter  10 value 112.865157
## iter  20 value 49.496558
## iter  30 value 21.368470
## iter  40 value 16.530902
## iter  50 value 15.900225
## iter  60 value 15.890798
## final  value 15.890752
## converged
## # weights:  787
## initial  value 420.319758
## iter  10 value 70.940478
## iter  20 value 37.311641
## iter  30 value 21.710421
## iter  40 value 15.748341
## iter  50 value 15.576996
## iter  60 value 15.529127
## iter  70 value 15.523372
## final  value 15.523368
## converged
## # weights:  787
## initial  value 337.264411
## iter  10 value 81.816715
## iter  20 value 44.897805
## iter  30 value 29.523125
## iter  40 value 26.138524
## iter  50 value 21.446464
## iter  60 value 20.928412
## iter  70 value 20.914206
## final  value 20.914015
## converged
## # weights:  787
## initial  value 276.023772
## iter  10 value 68.030811
## iter  20 value 31.388996
## iter  30 value 21.883363
## iter  40 value 16.853761
## iter  50 value 15.913223
## iter  60 value 15.886407
## iter  70 value 15.885990
## final  value 15.885979
## converged
## # weights:  787
## initial  value 559.722633
## iter  10 value 143.769934
## iter  20 value 76.268611
## iter  30 value 46.364927
## iter  40 value 31.472039
## iter  50 value 24.910279
## iter  60 value 21.928808
## iter  70 value 21.386366
## iter  80 value 21.357804
## iter  90 value 21.356035
## final  value 21.356003
## converged
```

Predicting the errors

```
predict_error <- predicting_errors(y, t_out_1)
cat("train1 prediction error", predict_error, "\n")
```

```
## train1 prediction error 0
```

To find the best decay, we can use a for loop

```
error_list_d <- rep(NA, 4)
for (i in 0:3) {
  tuning_df <- data.frame(size=5, decay=(i/10))
  fitControl <- trainControl(## 2-fold CV
    method = "repeatedcv",
    number = 2,
    repeats = 2)
  t_out_1 <- caret::train(x=x, y=y, method="nnet",
                        trControl = fitControl,
                        tuneGrid=tuning_df, maxit=1000, MaxNWts=10000)
  predict_error <- predicting_errors(y, t_out_1)
  error_list_d[i] <- predict_error

}
```

```
## # weights:  3931
## initial  value 783.190969
## iter  10 value 153.568607
## iter  20 value 153.567367
## iter  30 value 153.556891
## iter  40 value 135.881770
## iter  50 value 39.765514
## iter  60 value 36.331269
## iter  70 value 36.259217
## iter  80 value 36.259002
## iter  90 value 36.255713
## iter 100 value 36.239471
## iter 110 value 36.209020
## iter 120 value 35.938259
## iter 130 value 12.638611
## iter 140 value 0.026823
## iter 150 value 0.000706
## final  value 0.000088
## converged
## # weights:  3931
## initial  value 566.783101
## final  value 155.847979
## converged
## # weights:  3931
## initial  value 332.482088
## iter  10 value 43.697032
## iter  20 value 0.827813
## iter  30 value 0.011781
## iter  40 value 0.000710
## iter  50 value 0.000352
## final  value 0.000085
## converged
## # weights:  3931
## initial  value 209.393969
## iter  10 value 23.398724
## iter  20 value 1.756719
## iter  30 value 0.123019
## iter  40 value 0.009959
## iter  50 value 0.003714
## iter  60 value 0.001606
## iter  70 value 0.000199
## iter  80 value 0.000112
## final  value 0.000095
## converged
## # weights:  3931
## initial  value 1190.674253
## iter  10 value 121.981462
## iter  20 value 52.568181
## iter  30 value 31.435832
## iter  40 value 30.863472
## iter  50 value 29.187279
## iter  60 value 25.092583
## iter  70 value 21.427636
## iter  80 value 5.426596
## iter  90 value 3.406150
## iter 100 value 3.369140
## iter 110 value 3.365241
## iter 120 value 3.365191
## iter 120 value 3.365191
## final  value 3.365177
## converged
## # weights:  3931
## initial  value 476.510985
## iter  10 value 78.323339
## iter  20 value 22.253374
## iter  30 value 14.527723
## iter  40 value 10.533222
## iter  50 value 9.490200
## iter  60 value 9.065541
## iter  70 value 8.837522
## iter  80 value 8.808602
## iter  90 value 8.805559
## iter 100 value 8.805538
## final  value 8.805537
## converged
## # weights:  3931
## initial  value 335.495076
## iter  10 value 101.852907
## iter  20 value 36.701195
## iter  30 value 17.668228
## iter  40 value 15.193691
## iter  50 value 13.186560
## iter  60 value 11.404954
## iter  70 value 10.377162
## iter  80 value 9.878115
## iter  90 value 9.665955
## iter 100 value 9.454303
## iter 110 value 9.374300
## iter 120 value 9.362739
## iter 130 value 9.362233
## iter 140 value 9.362215
## iter 140 value 9.362215
## iter 140 value 9.362215
## final  value 9.362215
## converged
## # weights:  3931
## initial  value 275.409078
## iter  10 value 73.551355
## iter  20 value 23.333369
## iter  30 value 12.466308
## iter  40 value 11.113370
## iter  50 value 10.358839
## iter  60 value 10.053357
## iter  70 value 9.945929
## iter  80 value 9.939486
## iter  90 value 9.939299
## iter 100 value 9.939228
## final  value 9.939227
## converged
## # weights:  3931
## initial  value 430.983514
## iter  10 value 80.701663
## iter  20 value 28.235654
## iter  30 value 11.892024
## iter  40 value 10.254132
## iter  50 value 9.341492
## iter  60 value 8.979601
## iter  70 value 8.831144
## iter  80 value 8.794909
## iter  90 value 8.770714
## iter 100 value 8.742131
## iter 110 value 8.737534
## iter 120 value 8.737186
## iter 130 value 8.737175
## final  value 8.737174
## converged
## # weights:  3931
## initial  value 919.404129
## iter  10 value 199.392295
## iter  20 value 73.252878
## iter  30 value 32.960847
## iter  40 value 18.796988
## iter  50 value 17.311835
## iter  60 value 14.774143
## iter  70 value 14.164380
## iter  80 value 13.890693
## iter  90 value 13.812358
## iter 100 value 13.781810
## iter 110 value 13.764234
## iter 120 value 13.755719
## iter 130 value 13.752611
## iter 140 value 13.745857
## iter 150 value 13.742403
## iter 160 value 13.741188
## iter 170 value 13.741121
## iter 180 value 13.741109
## iter 190 value 13.741098
## final  value 13.741097
## converged
## # weights:  3931
## initial  value 723.880471
## iter  10 value 168.358563
## iter  20 value 55.353849
## iter  30 value 32.534084
## iter  40 value 20.415248
## iter  50 value 19.419294
## iter  60 value 18.762155
## iter  70 value 17.785080
## iter  80 value 17.209568
## iter  90 value 16.966916
## iter 100 value 16.899920
## iter 110 value 16.878153
## iter 120 value 16.822495
## iter 130 value 16.808384
## iter 140 value 16.807905
## iter 150 value 16.806965
## final  value 16.806902
## converged
## # weights:  3931
## initial  value 410.674094
## iter  10 value 109.615255
## iter  20 value 47.767189
## iter  30 value 26.264848
## iter  40 value 20.171176
## iter  50 value 17.556892
## iter  60 value 16.585171
## iter  70 value 16.309539
## iter  80 value 16.057205
## iter  90 value 15.739536
## iter 100 value 15.418192
## iter 110 value 15.283997
## iter 120 value 15.268914
## iter 130 value 15.268724
## iter 140 value 15.268711
## final  value 15.268711
## converged
## # weights:  3931
## initial  value 625.546123
## iter  10 value 94.574036
## iter  20 value 36.659690
## iter  30 value 24.958212
## iter  40 value 22.279676
## iter  50 value 20.479492
## iter  60 value 19.064189
## iter  70 value 18.041842
## iter  80 value 16.676321
## iter  90 value 16.202710
## iter 100 value 16.169194
## iter 110 value 16.157543
## iter 120 value 16.156839
## final  value 16.156820
## converged
## # weights:  3931
## initial  value 656.623541
## iter  10 value 116.512853
## iter  20 value 49.836097
## iter  30 value 29.030354
## iter  40 value 22.400069
## iter  50 value 20.950432
## iter  60 value 20.387594
## iter  70 value 20.189890
## iter  80 value 20.026838
## iter  90 value 19.101307
## iter 100 value 17.376442
## iter 110 value 16.426446
## iter 120 value 16.393253
## iter 130 value 16.392864
## final  value 16.392843
## converged
## # weights:  3931
## initial  value 1286.688523
## iter  10 value 340.153512
## iter  20 value 170.231001
## iter  30 value 62.030012
## iter  40 value 34.821820
## iter  50 value 28.160470
## iter  60 value 25.473052
## iter  70 value 23.960297
## iter  80 value 23.737127
## iter  90 value 23.691625
## iter 100 value 23.584979
## iter 110 value 23.428008
## iter 120 value 23.319439
## iter 130 value 23.137319
## iter 140 value 23.029768
## iter 150 value 23.014746
## iter 160 value 22.984066
## iter 170 value 22.974133
## iter 180 value 22.973591
## final  value 22.973588
## converged
## # weights:  3931
## initial  value 804.356698
## iter  10 value 98.910403
## iter  20 value 48.552588
## iter  30 value 35.523920
## iter  40 value 31.580741
## iter  50 value 29.608909
## iter  60 value 28.162085
## iter  70 value 26.515802
## iter  80 value 23.729115
## iter  90 value 23.385873
## iter 100 value 23.324675
## iter 110 value 23.304880
## iter 120 value 23.301682
## iter 130 value 23.301371
## final  value 23.301362
## converged
## # weights:  3931
## initial  value 477.776355
## iter  10 value 89.262557
## iter  20 value 30.772751
## iter  30 value 24.091576
## iter  40 value 20.446431
## iter  50 value 19.806014
## iter  60 value 19.724726
## iter  70 value 19.689276
## iter  80 value 19.674504
## iter  90 value 19.674122
## final  value 19.674121
## converged
## # weights:  3931
## initial  value 552.013104
## iter  10 value 91.604751
## iter  20 value 32.903220
## iter  30 value 25.430658
## iter  40 value 22.721401
## iter  50 value 22.076443
## iter  60 value 21.817810
## iter  70 value 21.687572
## iter  80 value 21.671074
## iter  90 value 21.664870
## iter 100 value 21.664262
## final  value 21.664245
## converged
## # weights:  3931
## initial  value 630.220527
## iter  10 value 90.010905
## iter  20 value 42.081050
## iter  30 value 31.737054
## iter  40 value 27.692109
## iter  50 value 23.943390
## iter  60 value 22.938523
## iter  70 value 22.147010
## iter  80 value 21.978509
## iter  90 value 21.924451
## iter 100 value 21.909198
## iter 110 value 21.908267
## final  value 21.908253
## converged
## # weights:  3931
## initial  value 822.582192
## iter  10 value 146.136458
## iter  20 value 57.971309
## iter  30 value 43.665130
## iter  40 value 39.996640
## iter  50 value 35.383308
## iter  60 value 34.134492
## iter  70 value 33.823483
## iter  80 value 33.206957
## iter  90 value 31.498437
## iter 100 value 31.132702
## iter 110 value 31.071717
## iter 120 value 31.060945
## iter 130 value 31.053958
## iter 140 value 31.049828
## iter 150 value 31.049052
## iter 160 value 31.047566
## final  value 31.047504
## converged
```

```
error_list_d
```

```
## [1]  0  0  0 NA
```

From the result we choose decay=0.1

We decide Size=5 decay=0.1 gives the best result

CROSS VALIDATION TEST

```
tuning_df <- data.frame(size=5, decay=.1)
fitControl <- trainControl(method="none")
fitControl <- trainControl(## 2-fold CV
  method = "repeatedcv",
  number = 2,
  repeats = 2)
t_out_f <- caret::train(x=x, y=y, method="nnet",
                        trControl = fitControl,
                        tuneGrid=tuning_df, maxit=1000, MaxNWts=10000)
```

```
## # weights:  3931
## initial  value 523.833497
## iter  10 value 107.384095
## iter  20 value 38.072743
## iter  30 value 23.808853
## iter  40 value 15.953220
## iter  50 value 11.406636
## iter  60 value 10.493137
## iter  70 value 9.933248
## iter  80 value 9.607673
## iter  90 value 9.214714
## iter 100 value 9.150555
## iter 110 value 9.128104
## iter 120 value 9.126676
## iter 130 value 9.126647
## final  value 9.126646
## converged
## # weights:  3931
## initial  value 558.488929
## iter  10 value 127.334621
## iter  20 value 71.810877
## iter  30 value 31.730191
## iter  40 value 18.272184
## iter  50 value 12.405121
## iter  60 value 10.620145
## iter  70 value 10.294825
## iter  80 value 10.128441
## iter  90 value 9.842073
## iter 100 value 9.727993
## iter 110 value 9.701540
## iter 120 value 9.666817
## iter 130 value 9.664135
## iter 140 value 9.664068
## final  value 9.664067
## converged
## # weights:  3931
## initial  value 354.438200
## iter  10 value 114.557180
## iter  20 value 37.335855
## iter  30 value 16.552523
## iter  40 value 12.006037
## iter  50 value 10.678454
## iter  60 value 9.836547
## iter  70 value 9.550460
## iter  80 value 9.465457
## iter  90 value 9.442467
## iter 100 value 9.436178
## iter 110 value 9.435370
## iter 120 value 9.435292
## iter 130 value 9.435234
## final  value 9.435231
## converged
## # weights:  3931
## initial  value 354.678955
## iter  10 value 111.363921
## iter  20 value 50.476310
## iter  30 value 21.170893
## iter  40 value 13.441307
## iter  50 value 10.838101
## iter  60 value 9.913820
## iter  70 value 9.619420
## iter  80 value 9.533238
## iter  90 value 9.508490
## iter 100 value 9.505123
## iter 110 value 9.505004
## final  value 9.505001
## converged
## # weights:  3931
## initial  value 733.266251
## iter  10 value 149.593050
## iter  20 value 55.411011
## iter  30 value 24.031033
## iter  40 value 18.217152
## iter  50 value 16.230307
## iter  60 value 15.364645
## iter  70 value 15.038584
## iter  80 value 14.705268
## iter  90 value 14.507869
## iter 100 value 14.404951
## iter 110 value 14.305019
## iter 120 value 14.108671
## iter 130 value 14.034201
## iter 140 value 14.027048
## iter 150 value 14.026497
## iter 160 value 14.026458
## final  value 14.026457
## converged
```

Compare the error to the test set error

```r
#Predicting the errors
predict_error <- predicting_errors(y, t_out_f)
cat("train1.1 prediction error", predict_error, "\n")
```

```
## train1.1 prediction error 0
```

```r
#Compare to the test data errors
pred_error_t <- predicting_errors(y_t, t_out_f)
cat("test prediction error", pred_error_t, "\n")
```

```
## test prediction error 0.186
```

So we have size=5, decay=0.1 as an acceptable combination.