# 1.7.1 自己符号化器

手書き文字(digit)認識をします。データは下記からダウンロードしてください。
http://www.kaggle.com/c/digit-recognizer (http://www.kaggle.com/c/digit-recognizer)

```
setwd("C:/Users/k-harada/Desktop/kdd/digit/ORG")
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 3.1.3
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.1.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:data.table':
##
##     between, last
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
traindata <- fread("train.csv")
```

```
##
Read 47.6% of 42000 rows
Read 95.2% of 42000 rows
Read 42000 rows and 785 (of 785) columns from 0.072 GB file in 00:00:04
```

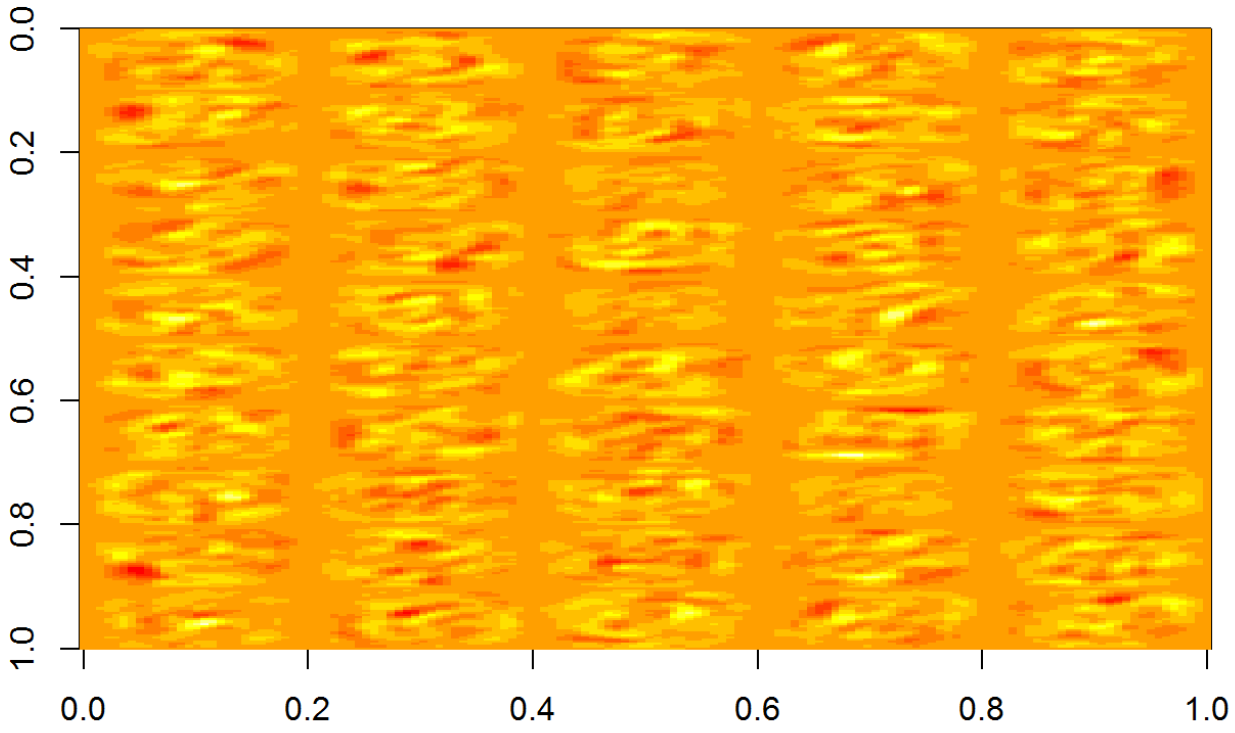
```
testdata <- fread("test.csv")

train_label <- traindata$label[1:28000]
valid_label <- traindata$label[28001:42000]

# do not resize here
# /255 so that range in 0-1
train_mat <- as.matrix(traindata[1:28000, ])[, -1]/255
valid_mat <- as.matrix(traindata[28001:42000, ])[,-1]/255
```

```
# initialize
# auto-encoder
set.seed(0)
W1 <- matrix(rnorm(28*28*50), ncol=28*28)
intercept1 <- rep(-0.5, length = 50)

W2 <- matrix(0, nrow = 28*28, ncol = 50)
intercept2 <- rep(0, length = 28*28)
```

```
# learn rate
eta1 <- 0.01
eta2 <- 0.01

# learn
for (loop in seq(5)){
  for (i in seq(28000)) {
      # feed forward
      output1 <- 1/(1 + exp(-1 * (W1 %*% train_mat[i, ] + intercept1)))
      output2 <- 1/(1 + exp(-1 * (W2 %*% output1 + intercept2)))
      # back propagation
      W2 <- W2 + eta2 * (train_mat[i, ] - output2) %*% t(output1)
      intercept2 <- intercept2 + eta2 * (train_mat[i, ] - output2)
      W1 <- W1 + eta1 * ((output1 * (1 - output1)) * (t(W2) %*% (train_mat[i, ] - output2))) %
*% t(train_mat[i, ])
      intercept1 <- intercept1 + eta1 * (output1 * (1 - output1)) * (t(W2) %*% (train_mat[i, ]
- output2))
  }
}
```

## 中間層を可視化

```
imagemat <- matrix(0, nrow=28*5, ncol=28*10)
for (i in seq(50)){
  imagemat[floor((i-1)/10)*28+1:28,((i-1)%%10)*28+1:28] <- matrix(W2[, i], ncol=28)
}
image(imagemat, ylim=c(1,0))
```




何かではあるが、組み合わせてはじめて意味があるものなのでよくわからない
(スパースにするとそうでもなくなるはず)

## これを初期条件にして学習

```
# 答えの用意
answer_mat <- matrix(0, nrow = 28000, ncol = 10)
for (i in seq(10)) {
    answer_mat[train_label == (i - 1), i] <- 1
}

# learn weights
set.seed(0)

W2 <- matrix(0, nrow = 10, ncol = 50)
intercept2 <- rep(0, length = 10)

# learn rate
eta1 <- 0.01
eta2 <- 0.01
for (loop in seq(5)){
  for (i in seq(28000)) {
      # feed forward
      output1 <- 1/(1 + exp(-1 * (W1 %*% train_mat[i, ] + intercept1)))
      output2 <- 1/(1 + exp(-1 * (W2 %*% output1 + intercept2)))
      # back propagation
      W2 <- W2 + eta2 * (answer_mat[i, ] - output2) %*% t(output1)
      intercept2 <- intercept2 + eta2 * (answer_mat[i, ] - output2)
      W1 <- W1 + eta1 * ((output1 * (1 - output1)) * (t(W2) %*% (answer_mat[i, ] - output2))) %
*% t(train_mat[i, ])
      intercept1 <- intercept1 + eta1 * (output1 * (1 - output1)) * (t(W2) %*% (answer_mat[i, ]
- output2))
  }
}
```

## 学習結果の確認

```
output_mat1 <- 1/(1 + exp(-1 * (train_mat %*% t(W1) + matrix(1, nrow = 28000, ncol = 1) %*% mat
rix(intercept1, ncol = 50))))
output_mat2 <- 1/(1 + exp(-1 * (output_mat1 %*% t(W2) + matrix(1, nrow = 28000, ncol = 1) %*% m
atrix(intercept2, ncol = 10))))

output_mat1_v <- 1/(1 + exp(-1 * (valid_mat %*% t(W1) + matrix(1, nrow = 14000, ncol = 1) %*% m
atrix(intercept1, ncol = 50))))
output_mat2_v <- 1/(1 + exp(-1 * (output_mat1_v %*% t(W2) + matrix(1, nrow = 14000, ncol = 1) %
*% matrix(intercept2, ncol = 10))))

# 出力
trainres <- max.col(as.matrix(output_mat2)) - 1
validres <- max.col(as.matrix(output_mat2_v)) - 1

table(trainres, train_label)
```

```
##         train_label
## trainres    0    1    2    3    4    5    6    7    8    9
##        0 2635    0   10    8    2   25   13    4    5    8
##        1    0 3033   14   13    9    9    5   19   21    5
##        2   10   16 2607   58   15   15   21   38   25   10
##        3    4   13   36 2656    0   85    3   12   61   41
##        4    6    4   29    2 2592   24   15   26   13   52
##        5   14    7    5   50    1 2203   30    3   31    3
##        6   19    3   34   13   16   41 2677    3   17    0
##        7    3    7   36   24    6    8    0 2727    5   40
##        8   20   17   45   66   13   77   13   12 2475   23
##        9   10    7    8   19  102   34    2   80   41 2583
```

```
mean(trainres == train_label)
```

```
## [1] 0.9352857
```

```
table(validres, valid_label)
```

```
##         valid_label
## validres    0    1    2    3    4    5    6    7    8    9
##        0 1369    0    7    7    1   12   10    3    4    6
##        1    0 1541    2    6    8    4    7    6   16    5
##        2    4    9 1247   35    9   12   20   20   12    6
##        3    4    6   13 1297    2   48    0    5   37   19
##        4    3    0   19    1 1220   18   10   10    5   40
##        5    8    2    9   33    0 1086   15    4   19    8
##        6    8    2   16    9   12   17 1282    0    9    1
##        7    1    2   12   10    4    4    0 1386    6   29
##        8   11   12   24   28   14   50   13    7 1237   15
##        9    3    3    4   16   46   23    1   36   24 1294
```

```
mean(validres == valid_label)
```

```
## [1] 0.9256429
```

何もしないよりは精度が上がったようである。