Classification

Yuval Benjamini

March 3, 2015

Example of clustering for ps3_realdata matrix

Loading the data into R

```
library('R.matlab')
```

```
## R.matlab v3.1.1 (2014-10-10) successfully loaded. See ?R.matlab for help.
##
## Attaching package: 'R.matlab'
##
## The following objects are masked from 'package:base':
##
##     getOption, isOpen
```

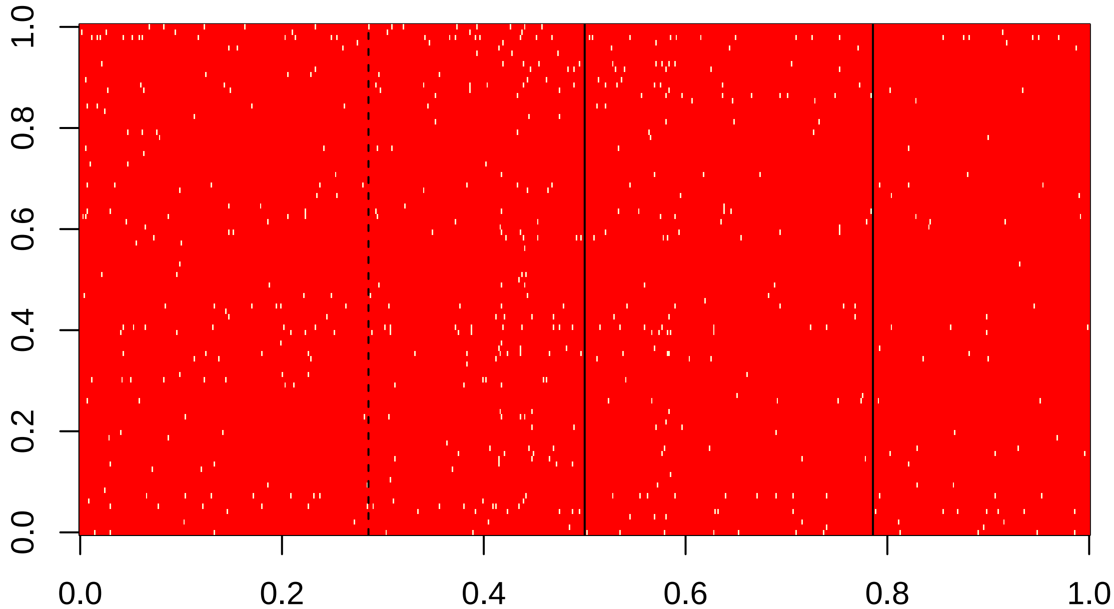
```
# Read data into R.

reread = FALSE
if (reread){
  dat = readMat('ps3_realdata.mat')
} else{
  load('ps3_realdata.RData')
}
# 97 neurons, 91 trials, 8 angles
```

We should sum spikes from 351 to 550

```
dim(dat$train.trial[2][[1]])
```

```
## [1]  97 700
```

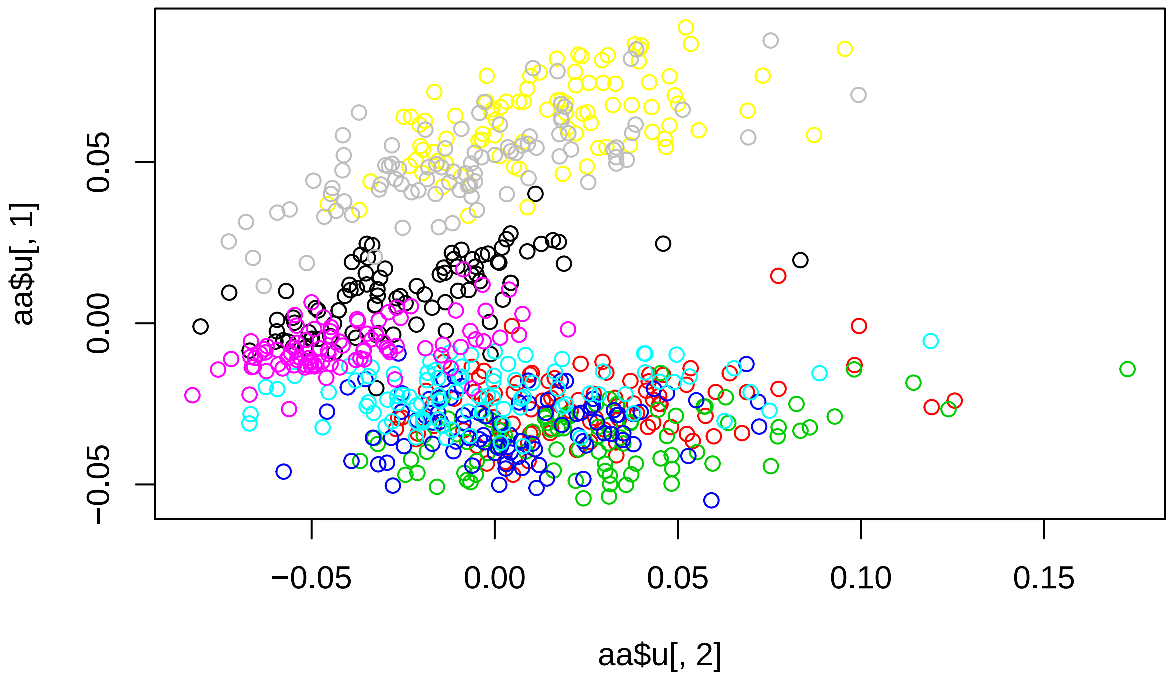
```
image(t(dat$train.trial[100][[1]]))
abline(v = c(200, 350,550)/700, lt =c(2,1,1),  )
```




```
neur = 97
vecs = matrix(nr =length(dat$train.trial)/2, nc = neur)
for (i in seq(2,length(dat$train.trial),2)){
  vecs[i/2,] = rowSums(dat$train.trial[i][[1]][,351:550])
}
test_vecs = matrix(nr =length(dat$test.trial)/2, nc = neur)
for (i in seq(2,length(dat$test.trial),2)){
  test_vecs[i/2,] = rowSums(dat$test.trial[i][[1]][,351:550])
}
```

Form labels, and check they are reasonable

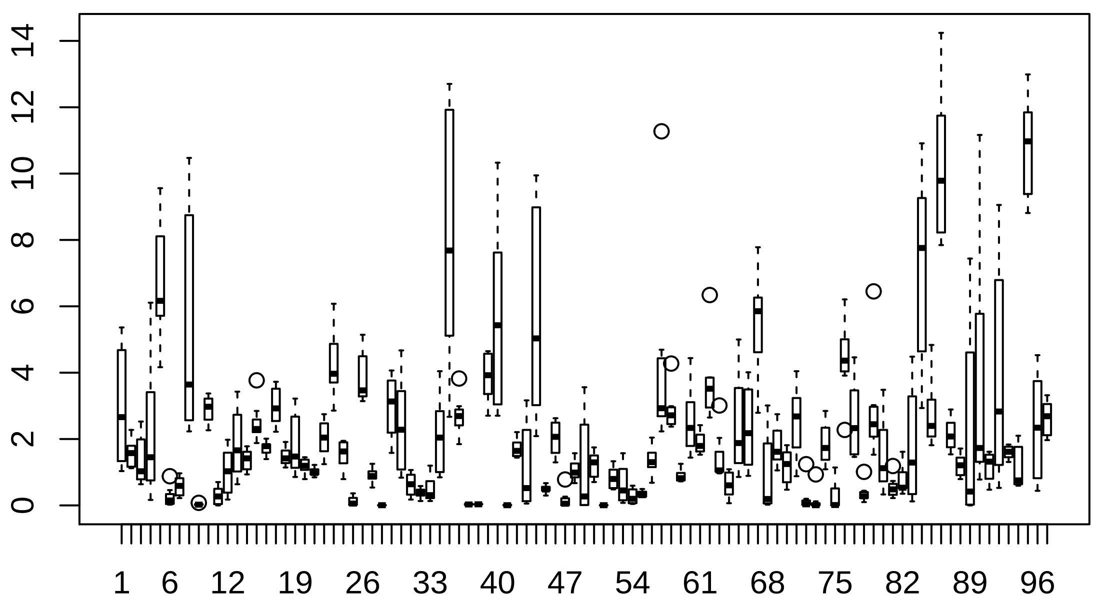
```
labels = rep(1:8, each = 91)
nlab = 8

# dimension reduction to look at clusters
aa = svd(scale(vecs,center = T,scale=F))
plot(aa$u[,2],aa$u[,1],col= labels)
```



Naive Bayes based on Poisson: Learn only the lambda parameter for each neuron

The parameters for each class can be estimated as the mean vector:

```
NB_params = matrix(nr = nlab, nc = neur)
for (l in 1:nlab){
  NB_params[l,] = colMeans(vecs[labels==l,])
}

boxplot(NB_params)
```



Compute the log likelihood. The likelihood in naive Bayes model is the product of the marginals. The log-likelihood is the sum of the log marginals.

```
NB_comp_loglike= function(examp, class,params = NB_params){
  return(sum(dpois(examp,params[class,],log = TRUE)))
}
```

We now compute the likelihood for both the training and the test data.

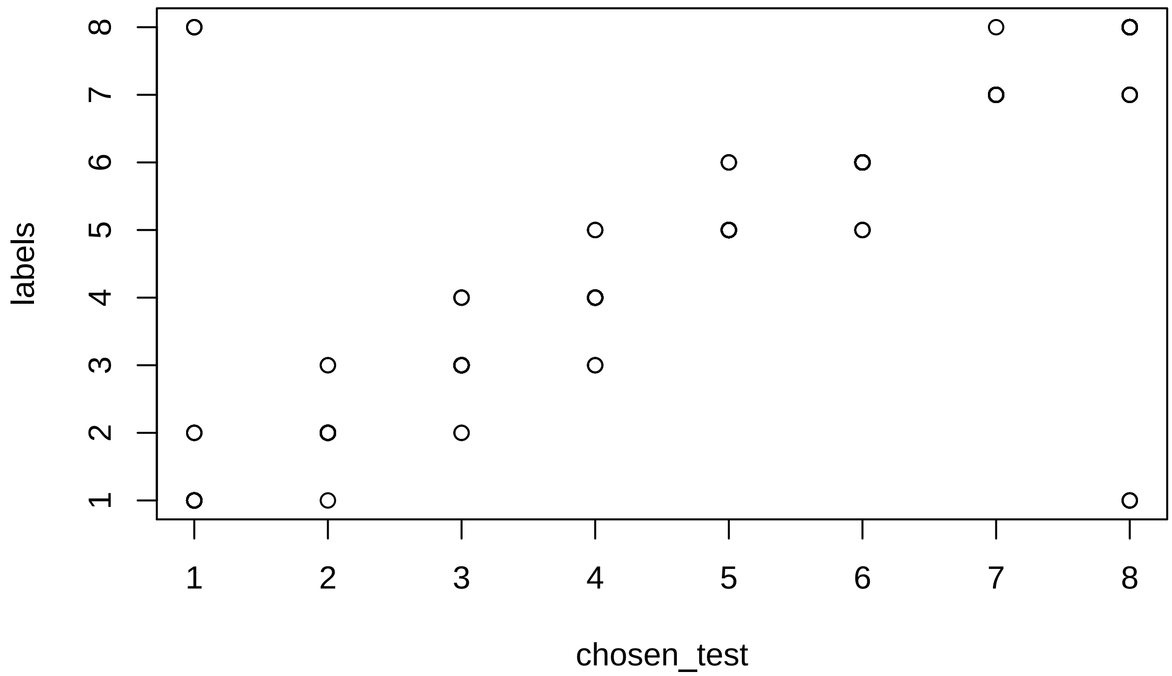
```
likes_for_test = matrix(nr = nrow(test_vecs),nc = nlab)

for (l in 1:nlab){
  for (j in 1:nrow(test_vecs)){
    likes_for_test[j,l] = NB_comp_loglike(examp = test_vecs[j,],NB_params,class = l)
  }
}
```

Warnings are due to 0-means. We set a minimum of 0.01 and try again

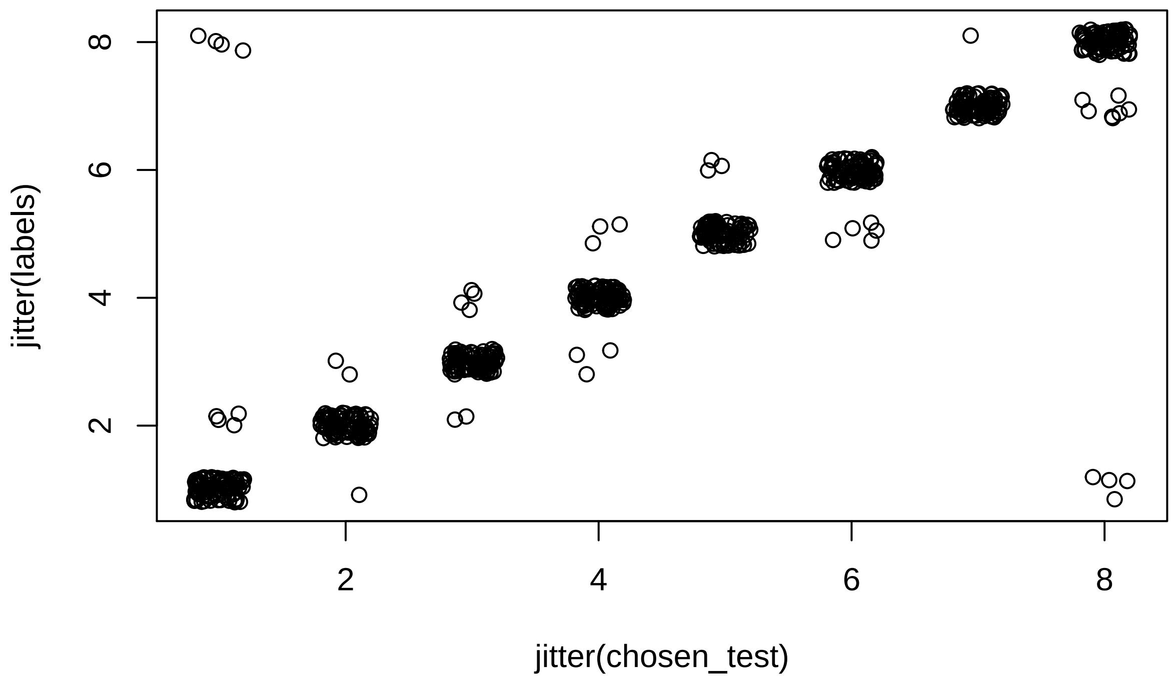
```
# Try again
NB_params_nz = NB_params
for (l in 1:nlab){ NB_params_nz[l,] = pmax(NB_params_nz[l,],0.01)}

for (l in 1:nlab){
  for (j in 1:nrow(test_vecs)){
    likes_for_test[j,l] = NB_comp_loglike(examp = test_vecs[j,],NB_params_nz,class = l)
  }
}

## For comparison
likes_for_train = matrix(nr = nrow(vecs),nc = nlab)
for (l in 1:nlab){
  for (j in 1:nrow(vecs)){
    likes_for_train[j,l] = NB_comp_loglike(examp = vecs[j,],NB_params_nz,class = l)
  }
}
```

Check classification results:

```
chosen_train  = numeric(nrow(test_vecs))
chosen_test= numeric(nrow(test_vecs))
for (j in 1:nrow(test_vecs)){
  chosen_train[j] = which.max(likes_for_train[j,])
  chosen_test[j] = which.max(likes_for_test[j,])
}

plot(chosen_test, labels)
```




```r
plot(jitter(chosen_test), jitter(labels))
```




```r
table(chosen_test, labels)
```

```
##              labels
## chosen_test   1  2  3  4  5  6  7  8
##             1 86  4  0  0  0  0  0  4
##             2  1 85  2  0  0  0  0  0
##             3  0  2 86  4  0  0  0  0
```

```
##              4  0  0  3 87  3  0  0  0
##              5  0  0  0  0 83  3  0  0
##              6  0  0  0  0  5 88  0  0
##              7  0  0  0  0  0  0 84  1
##              8  4  0  0  0  0  0  7 86
```