STK2100: Solutions Week 16

Lars H. B. Olsen

28.04.2021

Exam STK2100 2018

See solutions published at https://www.uio.no/studier/emner/matnat/math/STK2100/oppgaver/STK2100_2018_fasit.pdf. They are in Norwegian, but you should be able to translate the text with Google Translate, while the math is universal.

Extra Exercises

Exercise 13

The exercise provides you with all of the needed code, so I do not repeat that here. You will in *(e)* see some variation and that is due to the weights of the neural network are randomly drawn at initiation. Hence, we get different results each time we fit a NN, as the procedures is based on a stochastic initialization phase. That is why one can do ensemble learning in *(f)*.

Exercise 14

Note that we did this exercise in week 13, so I just copy over my solutions from that week.

a)

Bayes classifier is a classifier that minimizes the probability of misclassification (i.e. error rate). By using Bayes' theorem, we have

$$\Pr(Y|X) = \frac{\Pr(Y,X)}{\Pr(X)} = \frac{\Pr(X|Y)\Pr(Y)}{\Pr(X)}.$$

We are given that

$$\Pr(X|Y=k) = \text{Poisson}(\lambda_k) = \frac{(5+5k)^x e^{-(5+5k)}}{x!} = \frac{5^x(1+k)^x e^{-5(1+k)}}{x!}$$

and $\pi_k = \Pr(Y=k) = 1/K$, for $k = 1,2,3$ and where $K = 3$.

We can then compute the marginal distribution of X. We get

$$\begin{aligned}
\Pr(X) &= \sum_{k=1}^{3} \Pr(X|Y=k)\Pr(Y=k) \\
&= \frac{1}{3}\sum_{k=1}^{3} \Pr(X|Y=k) \\
&= \frac{1}{3}\left(\frac{10^x e^{-10}}{x!} + \frac{15^x e^{-15}}{x!} + \frac{20^x e^{-20}}{x!}\right) \\
&= \frac{5^x e^{-10}}{3(x!)}\left(2^x + 3^x e^{-5} + 4^x e^{-10}\right).
\end{aligned}$$

We can know insert all of these values into Bayes' theorem to obtain the following conditional distribution

$$\begin{aligned}
\Pr(Y = k|X) &= \frac{\Pr(X|Y = k)\Pr(Y = k)}{\Pr(X)} \\
&= \frac{\frac{5^x(1+k)^x e^{-5(1+k)}}{x!}\frac{1}{3}}{\frac{5^x e^{-10}}{3(x!)}\left(2^x + 3^x e^{-5} + 4^x e^{-10}\right)} \\
&= \frac{(1+k)^x e^{5(1-k)}}{2^x + 3^x e^{-5} + 4^x e^{-10}}.
\end{aligned}$$

Minimizing the probability of misclassification is equal to maximizing the probability of correct classification. Thus, we get the following Bayes classifier:

$$\text{argmax}_k \Pr(Y = k|X) = \text{argmax}_k \left\{ \frac{(1+k)^x e^{5(1-k)}}{2^x + 3^x e^{-5} + 4^x e^{-10}} \middle| x \right\} = \text{argmax}_k \left\{ (1+k)^x e^{5(1-k)} \middle| x \right\}$$

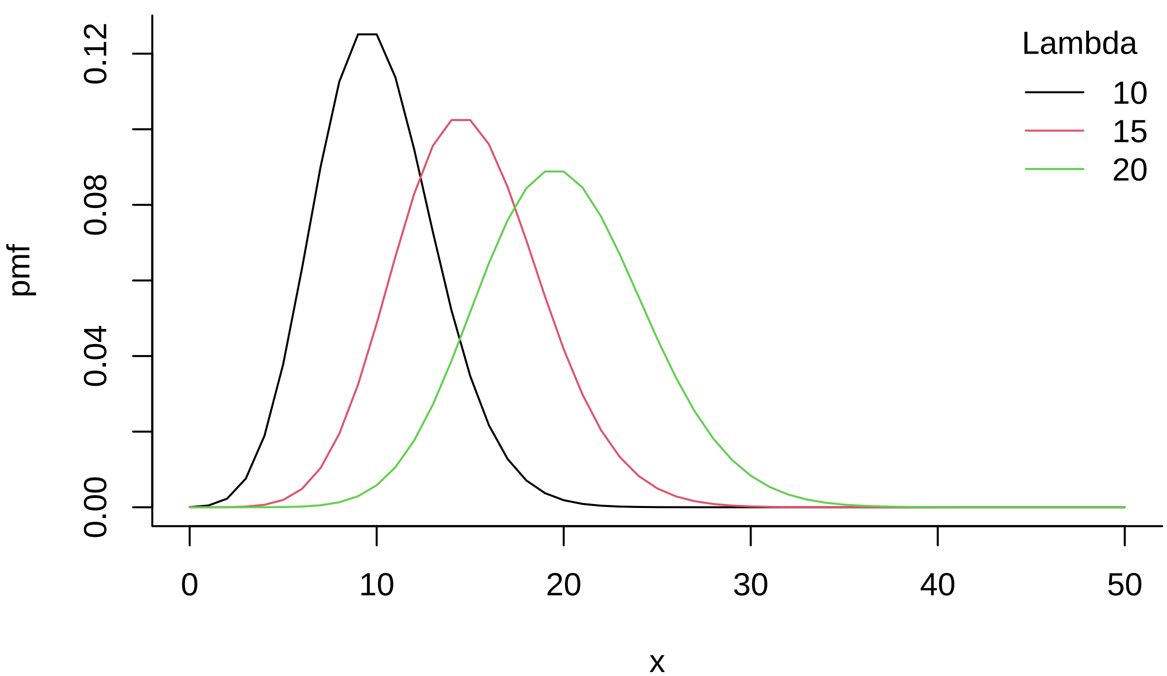
b)

Bayes classifier is a classifier that minimizes the probability of misclassification (i.e. error rate). So, error rate of Bayes classifier:

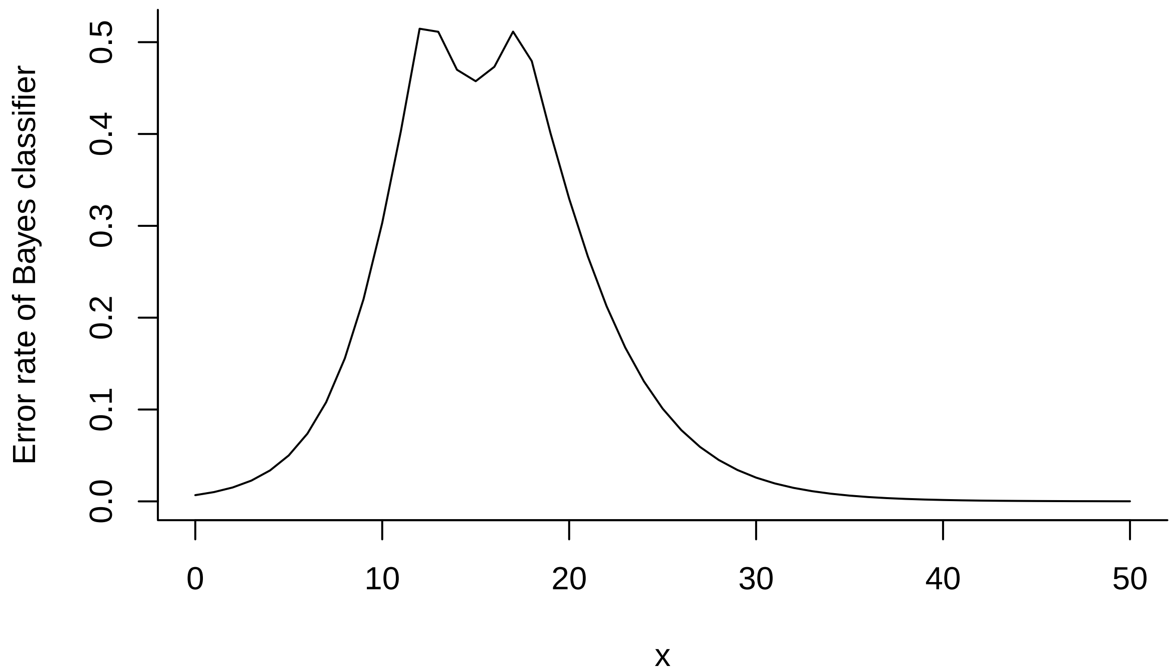
$$\Pr(Y \neq \hat{Y}|X) = 1 - \Pr(Y = \hat{Y}|X) = 1 - \max \Pr(Y = k|X).$$

Start by looking at the three probability mass functions. We see that there is an overlap from around 5 to 25, hence, we expect a higher error rate between these values.

```
# Start by looking at the three distributions
x.grid = 0:50
matplot(x.grid, cbind(dpois(x.grid, lambda = 10),
                      dpois(x.grid, lambda = 15),
                      dpois(x.grid, lambda = 20)),
        type = "l", lty = 1, bty = "l", xlab = "x", ylab = "pmf")
legend("topright", title = "Lambda", legend = c(10, 15, 20),
       lty = 1, col = 1:3, bty = "n")
```



```r
# Theoretical error rate of Bayes classifier
theoretical.Bayes.error.rate = function(x,K) {
  prob.mat = data.frame(k = 1:K, prob = NA)
  for (k in 1:K) {
    prob.mat[k, "prob"] = ((1 + k)^x)*exp(5 - 5*k)/(2^x +3^x*exp(-5) + 4^x*exp(-10))
  }
  theo.error.rate = 1 - prob.mat[which.max(prob.mat[,"prob"]), "prob"]
  return(theo.error.rate)
}

theoretical.Bayes.error.rate.vec =
  Vectorize(theoretical.Bayes.error.rate, vectorize.args = c("x"))

# Plot the theoretical error rate of Bayes classifier.
y.grid = theoretical.Bayes.error.rate.vec(x.grid, 3)
plot(x = x.grid, y = y.grid, type = "l", bty = "l",
     xlab = "x", ylab = "Error rate of Bayes classifier")
```



c)

```r
set.seed(1)

# Simulate y
simulated.data = data.frame(y = sample(x = 1:3, size = 1000, replace = T))

# Simulate X
simulated.data[(simulated.data[,"y"] == 1), "x"] =
  rpois(sum(simulated.data[,"y"] == 1), 10)
simulated.data[(simulated.data[,"y"] == 2), "x"] =
  rpois(sum(simulated.data[,"y"] == 2), 15)
simulated.data[(simulated.data[,"y"] == 3), "x"] =
  rpois(sum(simulated.data[,"y"] == 3), 20)
```

```r
# Bayes classifier
Bayes.classifier = function(x,K) {
  prob.mat = data.frame(k = 1:K, prob = NA)
  for (k in 1:K) {
    prob.mat[k, "prob"] = ((1 + k)^x)*exp(5 - 5*k)/(2^x +3^x*exp(-5) + 4^x*exp(-10))
}
  y.hat = prob.mat[which.max(prob.mat[,"prob"]), "k"]
  return(y.hat)
}

# Compute y.hat based on Bayes classifier.
Bayes.classifier.vec = Vectorize(Bayes.classifier, vectorize.args = c("x"))
simulated.data[,"y.hat.Bayes"] = Bayes.classifier.vec(simulated.data[,"x"], 3)
simulated.data[,"is.pred.correct"] =
  as.numeric(simulated.data[,"y.hat.Bayes"] == simulated.data[,"y"])


# Take a look at the data frame
head(simulated.data, 10)
```

```
##    y  x y.hat.Bayes is.pred.correct
## 1  1  7           1               1
## 2  3 18           3               1
## 3  1  8           1               1
## 4  2 11           1               0
## 5  1  5           1               1
## 6  3 17           2               0
## 7  3 17           2               0
## 8  2 15           2               1
## 9  2 17           2               1
## 10 3 14           2               0
```

```r
# Overall error rate
error.rate = 1 -
  sum(simulated.data[,"y"] == simulated.data[,"y.hat.Bayes"])/nrow(simulated.data)
show(error.rate)
```

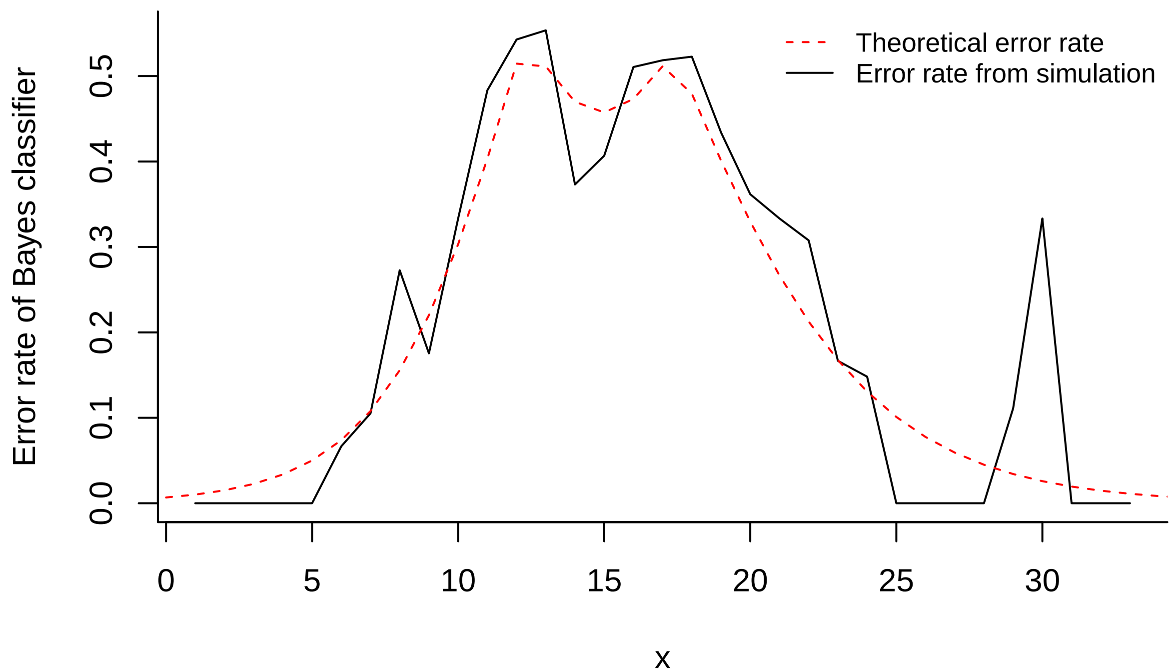
```
## [1] 0.345
```

```r
# Error rate per x value
empirical.error.rate.mat = data.frame(
  x = sort(unique(simulated.data[,"x"])),
  n = as.numeric(table(simulated.data[,"x"])),
  n.correct.pred = NA)

for (i in 1:nrow(empirical.error.rate.mat)) {
  x.target = empirical.error.rate.mat[i, "x"]
  empirical.error.rate.mat[i, "n.correct.pred"] =
    sum(simulated.data[(simulated.data[,"x"] == x.target), "is.pred.correct"])
}

empirical.error.rate.mat[,"error.rate"] =
  1 - empirical.error.rate.mat[,"n.correct.pred"]/empirical.error.rate.mat[,"n"]

# Take a look at the data frame
```

```r
head(empirical.error.rate.mat, 15)
```

```
##      x  n n.correct.pred error.rate
## 1    1  2              2 0.00000000
## 2    2  3              3 0.00000000
## 3    3  4              4 0.00000000
## 4    4  8              8 0.00000000
## 5    5 15             15 0.00000000
## 6    6 30             28 0.06666667
## 7    7 38             34 0.10526316
## 8    8 44             32 0.27272727
## 9    9 57             47 0.17543860
## 10 10 72             48 0.33333333
## 11 11 60             31 0.48333333
## 12 12 70             32 0.54285714
## 13 13 56             25 0.55357143
## 14 14 67             42 0.37313433
## 15 15 59             35 0.40677966
```

```r
# Plot the error rate as a function of x.
plot(x = empirical.error.rate.mat[,"x"], y = empirical.error.rate.mat[,"error.rate"],
     type = "l", xlab = "x", ylab = "Error rate of Bayes classifier", bty = "l")
points(x = x.grid, y = y.grid, type = "l", lty = 2, col = "red")
legend("topright", c("Theoretical error rate","Error rate from simulation"),
       lty = c(2,1), col = c("red","black"), bty = "n", cex = 0.8)
```



Exercise 15

Assume a classification problem where $\Pr(Y = 1) = \Pr(Y = 2) = 0.5$ and $X|Y = k \sim \mathcal{N}(\mu_k, 1)$, with $\mu_1 = -1$ and $\mu_2 = 1$.

a)

Use same approach as in exercise 14 (a)

$$\begin{aligned}
\Pr(X) &= \sum_{k=1}^{2} \Pr(X|Y=k)\Pr(Y=k) \\
&= \frac{1}{2}\sum_{k=1}^{2} \Pr(X|Y=k) \\
&= \frac{1}{2}\left(\frac{1}{\sqrt{2\pi}}e^{-\frac{(x+1)^2}{2}} + \frac{1}{\sqrt{2\pi}}e^{-\frac{(x-1)^2}{2}}\right) \\
&= \frac{1}{2\sqrt{2\pi}}\left(e^{-\frac{(x+1)^2}{2}} + e^{-\frac{(x-1)^2}{2}}\right)
\end{aligned}$$

$$\begin{aligned}
\Pr(Y=k|X) &= \frac{\Pr(X|Y=k)\Pr(Y=k)}{\Pr(X)} \\
&= \frac{\frac{1}{2}\cdot\frac{1}{\sqrt{2\pi}}e^{-\frac{(x-\mu_k)^2}{2}}}{\frac{1}{2\sqrt{2\pi}}\left(e^{-\frac{(x+1)^2}{2}} + e^{-\frac{(x-1)^2}{2}}\right)} \\
&= \frac{e^{-\frac{(x-\mu_k)^2}{2}}}{e^{-\frac{(x+1)^2}{2}} + e^{-\frac{(x-1)^2}{2}}}
\end{aligned}$$

$$\text{Bayes classifier:} \quad \arg\max_k \; \Pr(Y=k|X) = \arg\max_k \left\{ \frac{e^{-\frac{(x-\mu_k)^2}{2}}}{e^{-\frac{(x+1)^2}{2}} + e^{-\frac{(x-1)^2}{2}}} \middle| x \right\}$$

We can simplify this classifier further. We examine the decision boundary

$$\Pr(Y=1|X) > \Pr(Y=2|X) \iff -(x+1)^2 > -(x-1)^2 \iff x < 0.$$

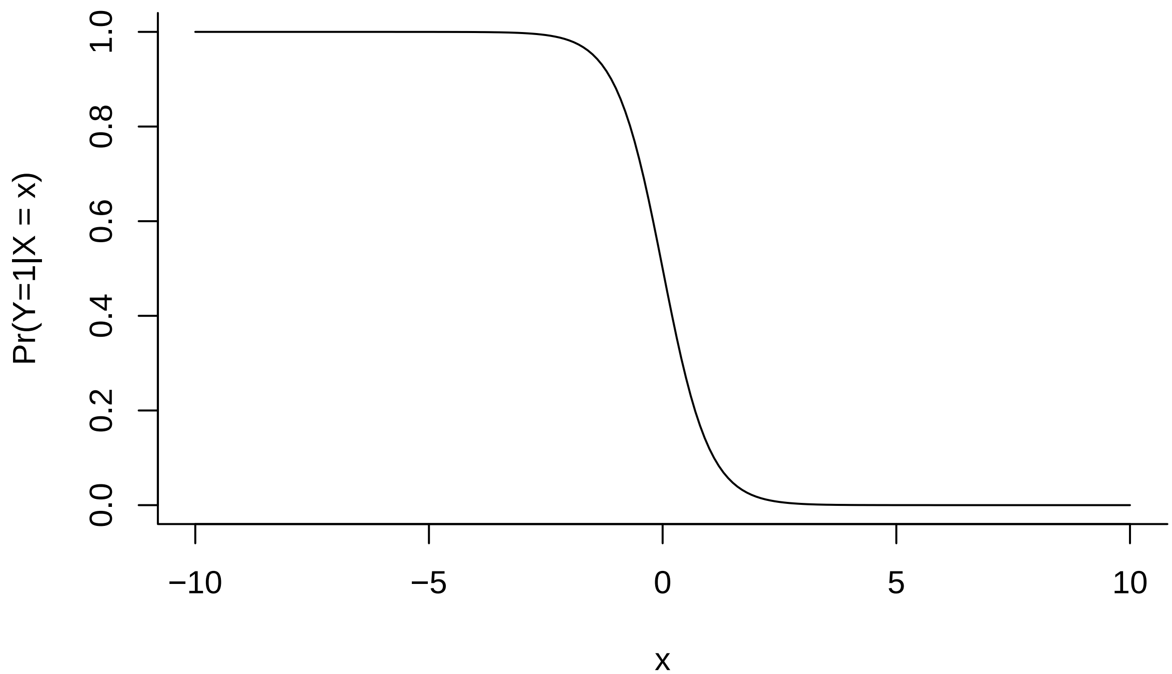
So, we have

$$\begin{aligned}
\text{Bayes classifier:} \quad k^{\text{Bayes}} &= \arg\min_k \{1 - \Pr(Y=k|X)\} \\
&= \arg\max_k \; \Pr(Y=k|X) \\
&= \begin{cases} 1 & \text{if } x < 0 \\ 2 & \text{otherwise} \end{cases}
\end{aligned}$$

b)

We plot $\Pr(Y=1|X=x) = \frac{\exp\left\{-\frac{(x+1)^2}{2}\right\}}{\exp\left\{-\frac{(x+1)^2}{2}\right\}+\exp\left\{-\frac{(x-1)^2}{2}\right\}}$. We see that $\Pr(Y=1|X=x)$ tends to -1 when x tends to $-\infty$, while it tends to 0 when x increase towards ∞.

```r
prob.y1.cond.x.func = function(x) {
  prob.y1.cond.x = exp(-((x+1)^2)/2) / (exp(-((x+1)^2)/2) + exp(-((x-1)^2)/2))
  return(prob.y1.cond.x)
}

x.grid = seq(from = -10, to = 10, by = 0.1)
y.grid = prob.y1.cond.x.func(x.grid)
plot(x = x.grid, y = y.grid, type = "l", xlab = "x", ylab = "Pr(Y=1|X = x)", bty = "l")
```



c)

$$\begin{aligned}
f_X(x) &= \sum_{k=1}^{2} \Pr(X|Y=k)\Pr(Y=k) \\
&= f(x|y=1)f(y=1) + f(x|y=2)f(y=2) \\
&= \frac{1}{2}f(x|y=1) + \frac{1}{2}f(x|y=2) \\
&= \frac{1}{2\sqrt{2\pi}}e^{-\frac{(x+1)^2}{2}} + \frac{1}{2\sqrt{2\pi}}e^{-\frac{(x-1)^2}{2}} \\
&= \frac{1}{2\sqrt{2\pi}}\left(e^{-\frac{(x+1)^2}{2}} + e^{-\frac{(x-1)^2}{2}}\right)
\end{aligned}$$

Null hyphothesis testing:

Reject H_0 if $F_X(x) < \frac{\alpha}{2}$ or $F_X(x) > 1 - \frac{\alpha}{2}$ where

$$F_X(x) = \int_{-\infty}^{x} f_X(u)\mathrm{d}u = \frac{1}{2}\Phi(x+1) + \frac{1}{2}\Phi(x-1).$$

d)

When $\alpha = 0$, the confidence interval is $(-\infty, \infty)$ and we will always accept the null hypothesis. In this case, the given classifier is equal to Bayes classifier.

```
Bayes.classifier = function(x) {
  y.hat = as.numeric(x < 0)*1 + as.numeric(x >= 0)*2
  return(y.hat)
}

custom.classifier = function(x, alpha) { # Test null hypothesis
  if ((1/2*pnorm(x+1) + 1/2*pnorm(x-1) < alpha/2) |
      (1/2*pnorm(x+1) + 1/2*pnorm(x-1) > 1 - alpha/2) ){
    # Null hypothesis is rejected
    y.hat = c("outlier")
  } else {
    # Null hypothesis is accepted
    y.hat = Bayes.classifier(x)
  }
  return(y.hat)
}

custom.classifier.vec = Vectorize(custom.classifier, vectorize.args = c("x"))
```

e)

```
# Set seed for reproducibility.
set.seed(2100)

# Simulate y
simulated.data = data.frame(y = sample(x = 1:2, size = 1000, replace = T))

# Look at the data
head(simulated.data)
```

```
##   y
## 1 2
## 2 2
## 3 1
## 4 1
## 5 1
## 6 1
```

```
# Simulate X based on which class Y the belong to
simulated.data[(simulated.data[,"y"] == 1), "x"] =
  rnorm(sum(simulated.data[,"y"] == 1), mean = -1, sd = 1)
simulated.data[(simulated.data[,"y"] == 2), "x"] =
  rnorm(sum(simulated.data[,"y"] == 2), mean = 1, sd = 1)

# Look at the data
head(simulated.data)
```

```
##   y          x
## 1 2 -0.3241373
## 2 2  0.3088101
## 3 1 -0.6333137
```

```
## 4 1  0.8845868
## 5 1 -0.5908170
## 6 1 -0.5119001
```

```r
## Perform classification
# alpha = 0.05
y.hat.1 = custom.classifier.vec(x = simulated.data[,"x"], alpha = 0.05)

# alpha = 0.01
y.hat.2 = custom.classifier.vec(x = simulated.data[,"x"], alpha = 0.01)

# alpha = 0
y.hat.3 = custom.classifier.vec(x = simulated.data[,"x"], alpha = 0)

# Look at the last 50 y.hat.1 predictions. See some outliers
y.hat.1[(nrow(simulated.data)-50):nrow(simulated.data)]
```

```
##  [1] "1"       "2"       "1"       "2"       "2"       "2"       "2"
##  [8] "1"       "1"       "2"       "1"       "1"       "1"       "outlier"
## [15] "2"       "2"       "2"       "1"       "2"       "1"       "2"
## [22] "2"       "2"       "1"       "1"       "2"       "1"       "2"
## [29] "outlier" "2"       "2"       "2"       "1"       "1"       "1"
## [36] "2"       "1"       "2"       "1"       "outlier" "2"       "1"
## [43] "1"       "2"       "1"       "2"       "2"       "1"       "1"
## [50] "2"       "2"
```

```r
# Get the probability of assigning "outlier"
cat("Prob outlier with alpha = 0.05: ",
    round(mean(y.hat.1 == "outlier"),3), sep = "", "\n")
```

```
## Prob outlier with alpha = 0.05: 0.057
```

```r
cat("Prob outlier with alpha = 0.01: ",
    round(mean(y.hat.2 == "outlier"),3), sep = "", "\n")
```

```
## Prob outlier with alpha = 0.01: 0.015
```

```r
cat("Prob outlier with alpha = 0.00: ",
    round(mean(y.hat.3 == "outlier"),3), sep = "", "\n")
```

```
## Prob outlier with alpha = 0.00: 0
```

```r
# Error rate
error.rate.1 = 1 - sum(simulated.data[,"y"] == y.hat.1)/nrow(simulated.data)
error.rate.2 = 1 - sum(simulated.data[,"y"] == y.hat.2)/nrow(simulated.data)
error.rate.3 = 1 - sum(simulated.data[,"y"] == y.hat.3)/nrow(simulated.data)

cat("Error rate with alpha = 0.05: ", round(error.rate.1,3), sep = "", "\n")
```

```
## Error rate with alpha = 0.05: 0.235
```

```r
cat("Error rate with alpha = 0.01: ", round(error.rate.2,3), sep = "", "\n")
```

```
## Error rate with alpha = 0.01: 0.193
```

```r
cat("Error rate with alpha = 0.00: ", round(error.rate.3,3), sep = "", "\n")
```

```
## Error rate with alpha = 0.00: 0.178
```