> # Tugas 1

> # Construct a linearly seperable dataset on 2-D plane

> set.seed(100)

> x=matrix(rnorm(20\*2), ncol=2)

> y=c(rep(-1,10),rep(1,10))

> x[y==1,]=x[y==1,]+1

>

> plot(x, col=(3-y))

>

> dat=data.frame(x=x,y=as.factor(y))

>

> # Load the libsvm R interface

> # Use Liblinear for very large problem

> library('e1071')

>

> svmfit=svm(y ~ ., data=dat, kernel='linear', cost=10, scale=FALSE)

> plot(svmfit,dat)

> svmfit$index

[1] 4 8 13 15 19

> summary(svmfit)

Call:

svm(formula = y ~ ., data = dat, kernel = "linear", cost = 10, scale = FALSE)

Parameters:

SVM-Type: C-classification

SVM-Kernel: linear

cost: 10

gamma: 0.5

Number of Support Vectors: 5

( 2 3 )

Number of Classes: 2

Levels:

-1 1

>

> # Find optimal tuning parameter

> set.seed (1)

> tune.out=tune(svm,y ~ .,data=dat, kernel="linear", ranges = list(cost=c(0.001, 0.01, 0.1, 1,5, 10, 100)))

> bestmod=tune.out$best.model

> summary(bestmod)

Call:

best.tune(method = svm, train.x = y ~ ., data = dat, ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)), kernel = "linear")

Parameters:

SVM-Type: C-classification

SVM-Kernel: linear

cost: 5

gamma: 0.5

Number of Support Vectors: 7

( 3 4 )

Number of Classes: 2

Levels:

-1 1

>

> # Construct the best data

> xtest=matrix(rnorm(20\*2), ncol=2)

> ytest=sample(c(-1,1), 20, rep=TRUE)

> xtest[ytest==1,]=xtest[ytest==1,]+1

> testdat=data.frame(x=xtest, y=as.factor(ytest))

>

> ypred=predict(bestmod, testdat)

> #confusion matrix

> tbl = table(predict=ypred, truth=testdat$y)

>

> true\_negative = as.vector(tbl[1])

> true\_negative

[1] 11

> false\_negative = as.vector(tbl[2])

> false\_negative

[1] 0

> false\_positive = as.vector(tbl[3])

> false\_positive

[1] 1

> true\_positive = as.vector(tbl[4])

> true\_positive

[1] 8

> sum\_all = true\_negative + false\_negative + false\_positive + true\_positive

> sum\_all

[1] 20

>

> accuracy = (true\_positive + true\_negative )/sum\_all

> accuracy

[1] 0.95

> error\_rate = (false\_positive + false\_negative)/sum\_all

> error\_rate

[1] 0.05

> sensitivity = true\_positive/(true\_positive + false\_negative)

> sensitivity

[1] 1

> specificity = true\_negative/(false\_positive + true\_negative)

> specificity

[1] 0.9166667

> precision = true\_positive/(true\_positive + false\_positive)

> precision

[1] 0.8888889

> recall = true\_positive/(true\_positive + false\_negative)

> recall

[1] 1

> f\_measure = (2\*precision\*recall)/(precision+recall)

> f\_measure

[1] 0.9411765