M6L3\_Assignment

Xueyi Fan

June 30, 2016

# Assignment:

1. Go to the UC Irvine Machine Learning Repository and find a dataset for supervised classification. Every student MUST use a different dataset so you MUST get approved for which you can going to use. This can be the same dataset you used for the unsupervised clustering as long as the data has some labeled data.
2. Classify your data using Support Vector Machines. You can use any method/package for SVMs. Answer the following questions:

* How well does the classifier perform?
* Try different kernels. How do they effect its performce?
* What might improve its performce?

library("ggplot2")  
library("e1071")  
library("kernlab")

##   
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':  
##   
## alpha

# Answer:

## Loading the data

Here, I choose the [Breast Cancer Wisconsin (Diagnostic) data set](https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/)

data\_url <- 'https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data'  
  
data <- read.table(url(data\_url), sep = ',')  
  
names(data) <- c('ID number', 'Diagnosis','radius\_mean','texure\_mean','perimeter\_mean','area\_mean','smoothness\_mean','compactness\_mean','concavity\_mean','concave\_points\_mean','symmetry\_mean','fractal\_dimension\_mean', 'radius\_SE','texure\_SE','perimeter\_SE','area\_SE','smoothness\_SE','compactness\_SE','concavity\_SE','concave\_points\_SE','symmetry\_SE','fractal\_dimension\_SE','radius\_worst','texure\_worst','perimeter\_worst','area\_worst','smoothness\_worst','compactness\_worst','concavity\_worst','concave\_points\_worst','symmetry\_worst','fractal\_dimension\_worst')   
  
head(data)

## ID number Diagnosis radius\_mean texure\_mean perimeter\_mean area\_mean  
## 1 842302 M 17.99 10.38 122.80 1001.0  
## 2 842517 M 20.57 17.77 132.90 1326.0  
## 3 84300903 M 19.69 21.25 130.00 1203.0  
## 4 84348301 M 11.42 20.38 77.58 386.1  
## 5 84358402 M 20.29 14.34 135.10 1297.0  
## 6 843786 M 12.45 15.70 82.57 477.1  
## smoothness\_mean compactness\_mean concavity\_mean concave\_points\_mean  
## 1 0.11840 0.27760 0.3001 0.14710  
## 2 0.08474 0.07864 0.0869 0.07017  
## 3 0.10960 0.15990 0.1974 0.12790  
## 4 0.14250 0.28390 0.2414 0.10520  
## 5 0.10030 0.13280 0.1980 0.10430  
## 6 0.12780 0.17000 0.1578 0.08089  
## symmetry\_mean fractal\_dimension\_mean radius\_SE texure\_SE perimeter\_SE  
## 1 0.2419 0.07871 1.0950 0.9053 8.589  
## 2 0.1812 0.05667 0.5435 0.7339 3.398  
## 3 0.2069 0.05999 0.7456 0.7869 4.585  
## 4 0.2597 0.09744 0.4956 1.1560 3.445  
## 5 0.1809 0.05883 0.7572 0.7813 5.438  
## 6 0.2087 0.07613 0.3345 0.8902 2.217  
## area\_SE smoothness\_SE compactness\_SE concavity\_SE concave\_points\_SE  
## 1 153.40 0.006399 0.04904 0.05373 0.01587  
## 2 74.08 0.005225 0.01308 0.01860 0.01340  
## 3 94.03 0.006150 0.04006 0.03832 0.02058  
## 4 27.23 0.009110 0.07458 0.05661 0.01867  
## 5 94.44 0.011490 0.02461 0.05688 0.01885  
## 6 27.19 0.007510 0.03345 0.03672 0.01137  
## symmetry\_SE fractal\_dimension\_SE radius\_worst texure\_worst  
## 1 0.03003 0.006193 25.38 17.33  
## 2 0.01389 0.003532 24.99 23.41  
## 3 0.02250 0.004571 23.57 25.53  
## 4 0.05963 0.009208 14.91 26.50  
## 5 0.01756 0.005115 22.54 16.67  
## 6 0.02165 0.005082 15.47 23.75  
## perimeter\_worst area\_worst smoothness\_worst compactness\_worst  
## 1 184.60 2019.0 0.1622 0.6656  
## 2 158.80 1956.0 0.1238 0.1866  
## 3 152.50 1709.0 0.1444 0.4245  
## 4 98.87 567.7 0.2098 0.8663  
## 5 152.20 1575.0 0.1374 0.2050  
## 6 103.40 741.6 0.1791 0.5249  
## concavity\_worst concave\_points\_worst symmetry\_worst  
## 1 0.7119 0.2654 0.4601  
## 2 0.2416 0.1860 0.2750  
## 3 0.4504 0.2430 0.3613  
## 4 0.6869 0.2575 0.6638  
## 5 0.4000 0.1625 0.2364  
## 6 0.5355 0.1741 0.3985  
## fractal\_dimension\_worst  
## 1 0.11890  
## 2 0.08902  
## 3 0.08758  
## 4 0.17300  
## 5 0.07678  
## 6 0.12440

str(data)

## 'data.frame': 569 obs. of 32 variables:  
## $ ID number : int 842302 842517 84300903 84348301 84358402 843786 844359 84458202 844981 84501001 ...  
## $ Diagnosis : Factor w/ 2 levels "B","M": 2 2 2 2 2 2 2 2 2 2 ...  
## $ radius\_mean : num 18 20.6 19.7 11.4 20.3 ...  
## $ texure\_mean : num 10.4 17.8 21.2 20.4 14.3 ...  
## $ perimeter\_mean : num 122.8 132.9 130 77.6 135.1 ...  
## $ area\_mean : num 1001 1326 1203 386 1297 ...  
## $ smoothness\_mean : num 0.1184 0.0847 0.1096 0.1425 0.1003 ...  
## $ compactness\_mean : num 0.2776 0.0786 0.1599 0.2839 0.1328 ...  
## $ concavity\_mean : num 0.3001 0.0869 0.1974 0.2414 0.198 ...  
## $ concave\_points\_mean : num 0.1471 0.0702 0.1279 0.1052 0.1043 ...  
## $ symmetry\_mean : num 0.242 0.181 0.207 0.26 0.181 ...  
## $ fractal\_dimension\_mean : num 0.0787 0.0567 0.06 0.0974 0.0588 ...  
## $ radius\_SE : num 1.095 0.543 0.746 0.496 0.757 ...  
## $ texure\_SE : num 0.905 0.734 0.787 1.156 0.781 ...  
## $ perimeter\_SE : num 8.59 3.4 4.58 3.44 5.44 ...  
## $ area\_SE : num 153.4 74.1 94 27.2 94.4 ...  
## $ smoothness\_SE : num 0.0064 0.00522 0.00615 0.00911 0.01149 ...  
## $ compactness\_SE : num 0.049 0.0131 0.0401 0.0746 0.0246 ...  
## $ concavity\_SE : num 0.0537 0.0186 0.0383 0.0566 0.0569 ...  
## $ concave\_points\_SE : num 0.0159 0.0134 0.0206 0.0187 0.0188 ...  
## $ symmetry\_SE : num 0.03 0.0139 0.0225 0.0596 0.0176 ...  
## $ fractal\_dimension\_SE : num 0.00619 0.00353 0.00457 0.00921 0.00511 ...  
## $ radius\_worst : num 25.4 25 23.6 14.9 22.5 ...  
## $ texure\_worst : num 17.3 23.4 25.5 26.5 16.7 ...  
## $ perimeter\_worst : num 184.6 158.8 152.5 98.9 152.2 ...  
## $ area\_worst : num 2019 1956 1709 568 1575 ...  
## $ smoothness\_worst : num 0.162 0.124 0.144 0.21 0.137 ...  
## $ compactness\_worst : num 0.666 0.187 0.424 0.866 0.205 ...  
## $ concavity\_worst : num 0.712 0.242 0.45 0.687 0.4 ...  
## $ concave\_points\_worst : num 0.265 0.186 0.243 0.258 0.163 ...  
## $ symmetry\_worst : num 0.46 0.275 0.361 0.664 0.236 ...  
## $ fractal\_dimension\_worst: num 0.1189 0.089 0.0876 0.173 0.0768 ...

summary(data)

## ID number Diagnosis radius\_mean texure\_mean   
## Min. : 8670 B:357 Min. : 6.981 Min. : 9.71   
## 1st Qu.: 869218 M:212 1st Qu.:11.700 1st Qu.:16.17   
## Median : 906024 Median :13.370 Median :18.84   
## Mean : 30371831 Mean :14.127 Mean :19.29   
## 3rd Qu.: 8813129 3rd Qu.:15.780 3rd Qu.:21.80   
## Max. :911320502 Max. :28.110 Max. :39.28   
## perimeter\_mean area\_mean smoothness\_mean compactness\_mean   
## Min. : 43.79 Min. : 143.5 Min. :0.05263 Min. :0.01938   
## 1st Qu.: 75.17 1st Qu.: 420.3 1st Qu.:0.08637 1st Qu.:0.06492   
## Median : 86.24 Median : 551.1 Median :0.09587 Median :0.09263   
## Mean : 91.97 Mean : 654.9 Mean :0.09636 Mean :0.10434   
## 3rd Qu.:104.10 3rd Qu.: 782.7 3rd Qu.:0.10530 3rd Qu.:0.13040   
## Max. :188.50 Max. :2501.0 Max. :0.16340 Max. :0.34540   
## concavity\_mean concave\_points\_mean symmetry\_mean   
## Min. :0.00000 Min. :0.00000 Min. :0.1060   
## 1st Qu.:0.02956 1st Qu.:0.02031 1st Qu.:0.1619   
## Median :0.06154 Median :0.03350 Median :0.1792   
## Mean :0.08880 Mean :0.04892 Mean :0.1812   
## 3rd Qu.:0.13070 3rd Qu.:0.07400 3rd Qu.:0.1957   
## Max. :0.42680 Max. :0.20120 Max. :0.3040   
## fractal\_dimension\_mean radius\_SE texure\_SE perimeter\_SE   
## Min. :0.04996 Min. :0.1115 Min. :0.3602 Min. : 0.757   
## 1st Qu.:0.05770 1st Qu.:0.2324 1st Qu.:0.8339 1st Qu.: 1.606   
## Median :0.06154 Median :0.3242 Median :1.1080 Median : 2.287   
## Mean :0.06280 Mean :0.4052 Mean :1.2169 Mean : 2.866   
## 3rd Qu.:0.06612 3rd Qu.:0.4789 3rd Qu.:1.4740 3rd Qu.: 3.357   
## Max. :0.09744 Max. :2.8730 Max. :4.8850 Max. :21.980   
## area\_SE smoothness\_SE compactness\_SE concavity\_SE   
## Min. : 6.802 Min. :0.001713 Min. :0.002252 Min. :0.00000   
## 1st Qu.: 17.850 1st Qu.:0.005169 1st Qu.:0.013080 1st Qu.:0.01509   
## Median : 24.530 Median :0.006380 Median :0.020450 Median :0.02589   
## Mean : 40.337 Mean :0.007041 Mean :0.025478 Mean :0.03189   
## 3rd Qu.: 45.190 3rd Qu.:0.008146 3rd Qu.:0.032450 3rd Qu.:0.04205   
## Max. :542.200 Max. :0.031130 Max. :0.135400 Max. :0.39600   
## concave\_points\_SE symmetry\_SE fractal\_dimension\_SE  
## Min. :0.000000 Min. :0.007882 Min. :0.0008948   
## 1st Qu.:0.007638 1st Qu.:0.015160 1st Qu.:0.0022480   
## Median :0.010930 Median :0.018730 Median :0.0031870   
## Mean :0.011796 Mean :0.020542 Mean :0.0037949   
## 3rd Qu.:0.014710 3rd Qu.:0.023480 3rd Qu.:0.0045580   
## Max. :0.052790 Max. :0.078950 Max. :0.0298400   
## radius\_worst texure\_worst perimeter\_worst area\_worst   
## Min. : 7.93 Min. :12.02 Min. : 50.41 Min. : 185.2   
## 1st Qu.:13.01 1st Qu.:21.08 1st Qu.: 84.11 1st Qu.: 515.3   
## Median :14.97 Median :25.41 Median : 97.66 Median : 686.5   
## Mean :16.27 Mean :25.68 Mean :107.26 Mean : 880.6   
## 3rd Qu.:18.79 3rd Qu.:29.72 3rd Qu.:125.40 3rd Qu.:1084.0   
## Max. :36.04 Max. :49.54 Max. :251.20 Max. :4254.0   
## smoothness\_worst compactness\_worst concavity\_worst concave\_points\_worst  
## Min. :0.07117 Min. :0.02729 Min. :0.0000 Min. :0.00000   
## 1st Qu.:0.11660 1st Qu.:0.14720 1st Qu.:0.1145 1st Qu.:0.06493   
## Median :0.13130 Median :0.21190 Median :0.2267 Median :0.09993   
## Mean :0.13237 Mean :0.25427 Mean :0.2722 Mean :0.11461   
## 3rd Qu.:0.14600 3rd Qu.:0.33910 3rd Qu.:0.3829 3rd Qu.:0.16140   
## Max. :0.22260 Max. :1.05800 Max. :1.2520 Max. :0.29100   
## symmetry\_worst fractal\_dimension\_worst  
## Min. :0.1565 Min. :0.05504   
## 1st Qu.:0.2504 1st Qu.:0.07146   
## Median :0.2822 Median :0.08004   
## Mean :0.2901 Mean :0.08395   
## 3rd Qu.:0.3179 3rd Qu.:0.09208   
## Max. :0.6638 Max. :0.20750

#shuffle the data   
set.seed(123)  
cancer\_data <- data[order(runif(nrow(data))),]  
cancer\_data[,2]

## [1] B M B B B B B B B M B B B B B B M M B M B B B B B B B M B B B B B M B  
## [36] B M B B B B M B B B B B B B B B B M M B B B B B B B B M B M B B B B B  
## [71] M B B M B B B B M M M B M B B B B M B B B M B B B B B B M B B M B B M  
## [106] M M B B B M B B M B B B B B M M B M M B B B B M B B B M B M B B B B M  
## [141] B B B M B B M B B M B B M B B M B B B M B B B M M M M B B B B B B M B  
## [176] B B B M B M B B M M M B M B B B M M B B M M M B B B M M M B B M B B M  
## [211] B B B M B M M B B B B B B B M B B B B B B B M B B B M B B B M M M M B  
## [246] M M M M M M B M B M B B B M M B B B B B B M B B M M M M M B B B M M B  
## [281] B M B M B B B B M B M B B B B M B B B B B B M M M M B M B B M B M B M  
## [316] B B B B B B M B B B B M B M B B B M M M B M B M B B M B M M M B M M B  
## [351] M M M B M B M B B B B B B B B M M B B B B M M M M M B M M B B B B B B  
## [386] B B M M M M M M B B M M B B M M B B B B M B M M B M M B M B B B M B B  
## [421] B B B B M B B M B B B B M B M B M M M B B M M M B M B M M M M B M M M  
## [456] B M B B B B B M B B B M B B M B M B M B B B B M B B M B B M B B B B B  
## [491] B M B M M M M B B M B M M B B B B M B M M M B B M B M B B B M B M M M  
## [526] M B B B M B M B B M B M M B B M B M M B B B M B M B B M B M M B B B M  
## [561] B B M B B B M M B  
## Levels: B M

## Classifing my data using Support Vector Machines

#to normalize data  
normalize <- function(x){  
 return((x-min(x))/(max(x)-min(x)))  
}  
cancer.normalized <- as.data.frame(lapply(cancer\_data[,3:32],normalize))  
  
#split my data into training part and testing part  
cancer.train <- cancer.normalized[1:500,]  
cancer.test <- cancer.normalized[501:569,]  
cancer.train.target <- cancer\_data[1:500,2]  
cancer.test.target <- cancer\_data[501:569,2]  
  
#using SVM  
  
cancer.svm <- svm(cancer.train.target~., data= cancer.train, kernel = "linear")  
summary(cancer.svm)

##   
## Call:  
## svm(formula = cancer.train.target ~ ., data = cancer.train, kernel = "linear")  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: linear   
## cost: 1   
## gamma: 0.03333333   
##   
## Number of Support Vectors: 39  
##   
## ( 22 17 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## B M

## Answer the following questions:

### How well does the classifier perform?

cancer.svm.pred <- predict(cancer.svm, cancer.test)  
cancer.svm.pred

## 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518   
## B M M B B B B M B M M M B B M B M B   
## 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536   
## B B M B M M M M B B B M B M B B B B   
## 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554   
## M M B B M B M M B B B M B M B B M B   
## 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569   
## M M B B B M B B M B B B M M B   
## Levels: B M

table(cancer.svm.pred, cancer.test.target)

## cancer.test.target  
## cancer.svm.pred B M  
## B 39 1  
## M 0 29

#look at agreement vs. non-agreement  
agreement <- cancer.svm.pred == cancer.test.target  
table(agreement)

## agreement  
## FALSE TRUE   
## 1 68

prop.table(table(agreement))

## agreement  
## FALSE TRUE   
## 0.01449275 0.98550725

Using SVM to classify the data, there is one misclassification. Although it is not the perfect classification comparing with other supervised machine learning methods in previous two lessons, it still has a good performance.

I also compare the prediction error between multiple linear regression and SVM

#create a multiple linear regression model  
  
y <- as.numeric(cancer.train.target == 'B') # y=1 for benign and y=0 for malignant  
  
cancer.lm <- lm(y~., data=cancer.train)  
#make a prediction for each X  
cancer.lm.pred <- predict(cancer.lm, cancer.train)  
  
#function to find the error  
rmse <- function(error){  
 sqrt(mean(error^2))  
}  
  
error <- cancer.lm$residuals  
RMSE.lm.pred <- rmse(error)  
RMSE.lm.pred

## [1] 0.2252355

#create SVM model  
cancer.svm <- svm(y~., data=cancer.train, kernel = "linear")  
cancer.train.pred<- predict(cancer.svm, cancer.train)  
  
#function to find the error  
error.svm <- y - cancer.train.pred  
RMSE.svm.pred <- rmse(error.svm)  
RMSE.svm.pred

## [1] 0.2369482

Comparing the SVM model with multiple linear regression model, the root mean squared error of SVM is 0.2369482, the root mean squared error of linear model is 0.2252355. They have similar performance.

### Try different kernels. How do they effect its performce?

I will try "linear", "radial", "polynomial", and "sigmoid"

#kernel: linear  
svm.linear <- svm(cancer.train.target~., data = cancer.train, kernel = "linear")  
svm.linear.pred <- predict(svm.linear, cancer.test)  
table(svm.linear.pred, cancer.test.target)

## cancer.test.target  
## svm.linear.pred B M  
## B 39 1  
## M 0 29

#kernel: radial basis  
svm.radial <- svm(cancer.train.target~., data = cancer.train, kernel = "radial")  
svm.radial.pred <- predict(svm.radial, cancer.test)  
table(svm.radial.pred, cancer.test.target)

## cancer.test.target  
## svm.radial.pred B M  
## B 38 2  
## M 1 28

#kernel: polynomial  
svm.polynomial <- svm(cancer.train.target~., data = cancer.train, kernel = "polynomial")  
svm.polynomial.pred <- predict(svm.polynomial, cancer.test)  
table(svm.polynomial.pred, cancer.test.target)

## cancer.test.target  
## svm.polynomial.pred B M  
## B 39 8  
## M 0 22

#kernel: sigmoid  
svm.sigmoid <- svm(cancer.train.target~., data = cancer.train, kernel = "sigmoid")  
svm.sigmoid.pred <- predict(svm.sigmoid, cancer.test)  
table(svm.sigmoid.pred, cancer.test.target)

## cancer.test.target  
## svm.sigmoid.pred B M  
## B 39 5  
## M 0 25

When I used kernel linear, there is only one misclassification; when I used kernel radial basis, there are three misclssifications; when I used polynomial, there are eight misclassifications; when I used kernel sigmoid, there are five misclassifications.

So different kernels will effect the performance of the model. For my data, choosing the kenel linear will have the best performance.

### What might improve its performance?

Choosing an appropriate kenel and parameters might improve its performance.

After comparing the different kernels, I decide to use linear which has the best performance with other parameters defaulted(epsilon=0.1, cost=1).

Then I will use cross validation to set best choice of epsilon and cost.

#kernel linear  
linear.tune.out <- tune(svm, cancer.normalized,cancer\_data[,2], kernel="linear", ranges= list(epsilon = c(0,1,0.1), cost=2^(-2:10)))  
summary(linear.tune.out)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## epsilon cost  
## 0 0.5  
##   
## - best performance: 0.0193609   
##   
## - Detailed performance results:  
## epsilon cost error dispersion  
## 1 0.0 0.25 0.01936090 0.01751014  
## 2 1.0 0.25 0.01936090 0.01751014  
## 3 0.1 0.25 0.01936090 0.01751014  
## 4 0.0 0.50 0.01936090 0.01936497  
## 5 1.0 0.50 0.01936090 0.01936497  
## 6 0.1 0.50 0.01936090 0.01936497  
## 7 0.0 1.00 0.02111529 0.01817415  
## 8 1.0 1.00 0.02111529 0.01817415  
## 9 0.1 1.00 0.02111529 0.01817415  
## 10 0.0 2.00 0.02816416 0.01710222  
## 11 1.0 2.00 0.02816416 0.01710222  
## 12 0.1 2.00 0.02816416 0.01710222  
## 13 0.0 4.00 0.02816416 0.01496960  
## 14 1.0 4.00 0.02816416 0.01496960  
## 15 0.1 4.00 0.02816416 0.01496960  
## 16 0.0 8.00 0.03345865 0.02125036  
## 17 1.0 8.00 0.03345865 0.02125036  
## 18 0.1 8.00 0.03345865 0.02125036  
## 19 0.0 16.00 0.03167293 0.02002969  
## 20 1.0 16.00 0.03167293 0.02002969  
## 21 0.1 16.00 0.03167293 0.02002969  
## 22 0.0 32.00 0.03693609 0.01754290  
## 23 1.0 32.00 0.03693609 0.01754290  
## 24 0.1 32.00 0.03693609 0.01754290  
## 25 0.0 64.00 0.04219925 0.01891961  
## 26 1.0 64.00 0.04219925 0.01891961  
## 27 0.1 64.00 0.04219925 0.01891961  
## 28 0.0 128.00 0.04395363 0.01497368  
## 29 1.0 128.00 0.04395363 0.01497368  
## 30 0.1 128.00 0.04395363 0.01497368  
## 31 0.0 256.00 0.04746241 0.01448443  
## 32 1.0 256.00 0.04746241 0.01448443  
## 33 0.1 256.00 0.04746241 0.01448443  
## 34 0.0 512.00 0.05275689 0.01862869  
## 35 1.0 512.00 0.05275689 0.01862869  
## 36 0.1 512.00 0.05275689 0.01862869  
## 37 0.0 1024.00 0.05100251 0.01554031  
## 38 1.0 1024.00 0.05100251 0.01554031  
## 39 0.1 1024.00 0.05100251 0.01554031

By using cross validation to set best parameter, I can get epsilon=0 and cost = 0.5. So I use kernel linear, epsilon=0 and cost=0.5.

#defaulted parameter   
svm.radial.default<- svm(cancer.train.target~., data = cancer.train, kernel = "radial")  
svm.radial.pred <- predict(svm.radial.default, cancer.test)  
table(svm.radial.pred, cancer.test.target)

## cancer.test.target  
## svm.radial.pred B M  
## B 38 2  
## M 1 28

#best performance parameter  
svm.linear.best <- svm(cancer.train.target~., data = cancer.train, kernel = "linear", cost=0.5, epsilon=0)  
svm.linear.best.pred <- predict(svm.linear.best, cancer.test)  
table(svm.linear.best.pred, cancer.test.target)

## cancer.test.target  
## svm.linear.best.pred B M  
## B 39 1  
## M 0 29

When I use kernel radial and cost=1, epsilon=0.1, there are three misclassifications. I change to use linear and cost=0.5, epsilon=0 and there are only one misclassification. Choosing right kernel and parameters improves the performance.