exam-2018-with-solutions

Maximilian Pfundstein

10/15/2019

Graphical Models

```
set.seed(567)
data("asia")
ind <- sample(1:5000, 4000)
train <- asia[ind,]
test <- asia[-ind,]
```

```
train_10 = train[1:10,]
train_20 = train[1:20,]
train_50 = train[1:50,]
train_100 = train[1:100,]
train_1000 = train[1:1000,]
train_2000 = train[1:2000,]
test_1000 = test[1:1000,]
```

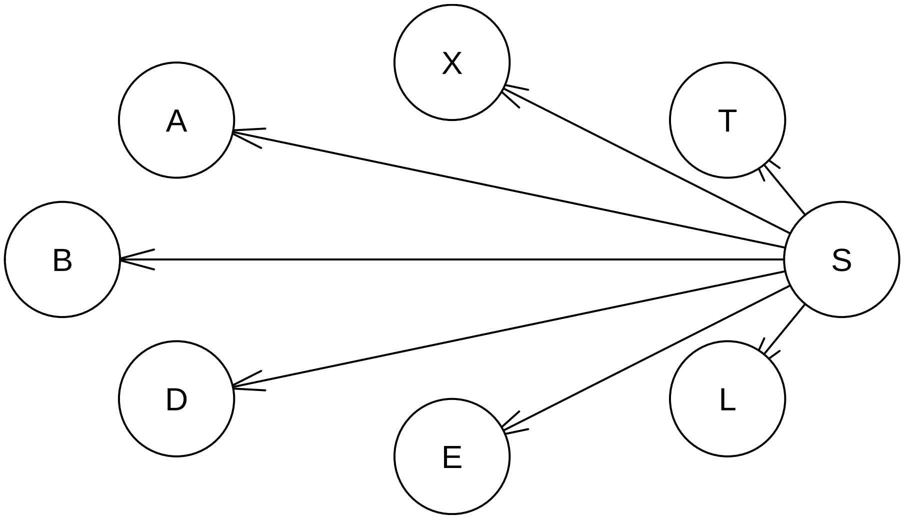
```
## Training the network
train_bayesian_network = function(structure = NULL,
                                  data,
                                  learning_algorithm = iamb,
                                  ...) {

  # Network
  if (is.null(structure)) {
    bayesian_network = learning_algorithm(data, ...)
  }
  else {
    bayesian_network = structure
  }

  # Parameters
  bayesian_network_fit = bn.fit(bayesian_network, data, method="bayes")
  bayesian_network_grain = compile(as.grain(bayesian_network_fit))

  return(list(bn_grain=bayesian_network_grain,
              bn_fit=bayesian_network_fit,
              bn_structure=bayesian_network))
}
```

```
# Structure
bn_naive_bayes = model2network("[S][A|S][T|S][L|S][B|S][E|S][X|S][D|S]")
plot(bn_naive_bayes)
```




```
bn_naive_bayes10 = train_bayesian_network(structure = bn_naive_bayes,
                        data = train_10,
                        learning_algorithm = hc)
```

```
## Warning in check.data(data, allow.missing = TRUE): variable A has levels
## that are not observed in the data.
```

```
## Warning in check.data(data, allow.missing = TRUE): variable T has levels
## that are not observed in the data.
```

```
bn_naive_bayes20 = train_bayesian_network(structure = bn_naive_bayes,
                        data = train_20,
                        learning_algorithm = hc)
```

```
## Warning in check.data(data, allow.missing = TRUE): variable A has levels
## that are not observed in the data.

## Warning in check.data(data, allow.missing = TRUE): variable T has levels
## that are not observed in the data.
```

```
bn_naive_bayes50 = train_bayesian_network(structure = bn_naive_bayes,
                        data = train_50,
                        learning_algorithm = hc)
```

```
## Warning in check.data(data, allow.missing = TRUE): variable T has levels
## that are not observed in the data.
```

```
bn_naive_bayes100 = train_bayesian_network(structure = bn_naive_bayes,
                        data = train_100,
                        learning_algorithm = hc)
```

```
## Warning in check.data(data, allow.missing = TRUE): variable T has levels
## that are not observed in the data.
```

```
bn_naive_bayes1000 = train_bayesian_network(structure = bn_naive_bayes,
                        data = train_1000,
                        learning_algorithm = hc)

bn_naive_bayes2000 = train_bayesian_network(structure = bn_naive_bayes,
                        data = train_2000,
                        learning_algorithm = hc)
```

```r
## Predicting with the network
predict_bayesian_network = function(bayesian_network,
                                    testX_ = testX,
                                    testY_ = testY) {

  res = apply(testX_, 1, FUN = function(x) {
    return(querygrain(setEvidence(bayesian_network$bn_grain, names(x), x))$S)
  })

  # Classify
  pred = apply(res, 2, FUN = function(x) {
    if (x[2] > 0.5) return("yes")
    return("no")
  })

  # Factorise
  testY_factor = testY_
  pred_factor = factor(pred)

  # Call to a library to calculate interesting metrics
  confusion_matrix = table(pred_factor, testY_factor)

  return(list(res=res, pred=pred, cf=confusion_matrix))
}

myResConf10 = predict_bayesian_network(bn_naive_bayes10, test_1000[,-2], test_1000$S)$cf
myResConf20 = predict_bayesian_network(bn_naive_bayes20, test_1000[,-2], test_1000$S)$cf
myResConf50 = predict_bayesian_network(bn_naive_bayes50, test_1000[,-2], test_1000$S)$cf
myResConf100 = predict_bayesian_network(bn_naive_bayes100, test_1000[,-2], test_1000$S)$cf
myResConf1000 = predict_bayesian_network(bn_naive_bayes1000, test_1000[,-2], test_1000$S)$cf
myResConf2000 = predict_bayesian_network(bn_naive_bayes2000, test_1000[,-2], test_1000$S)$cf

acc10 = sum(diag(myResConf10))/sum(myResConf10)
acc20 = sum(diag(myResConf20))/sum(myResConf20)
acc50 = sum(diag(myResConf50))/sum(myResConf50)
acc100 = sum(diag(myResConf100))/sum(myResConf100)
acc1000 = sum(diag(myResConf1000))/sum(myResConf1000)
acc2000 = sum(diag(myResConf2000))/sum(myResConf2000)

print(c(acc10, acc20, acc50, acc100, acc1000, acc2000))
```

```
## [1] 0.672 0.672 0.665 0.665 0.665 0.665
```