exam-2018-with-solutions

Maximilian Pfundstein

10/15/2019

Graphical Models

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

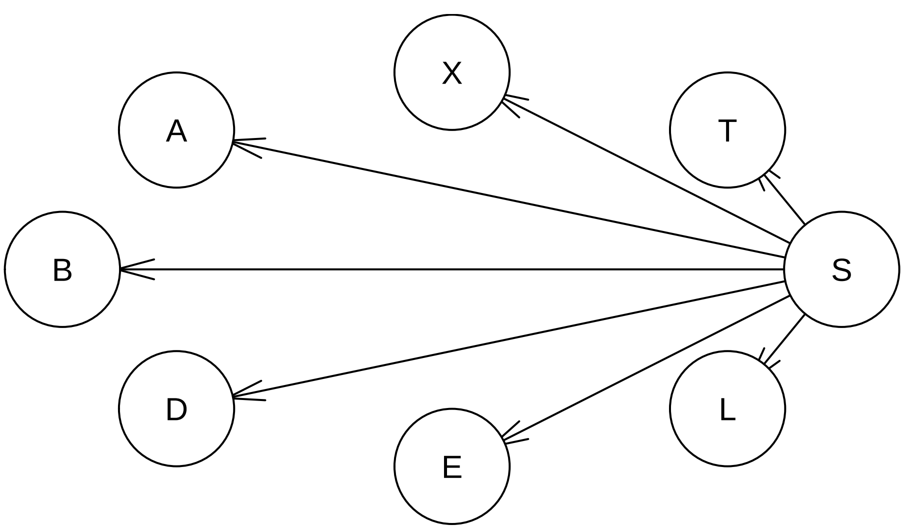
```
test_network = function(network, test_set) {
  grain = compile(as.grain(network))

  predictions = t(apply(test_set, 1, function(x) {
    evidence = setEvidence(grain, names(x[-2]), x[-2])
    query = querygrain(evidence)
    return(query$S)
  }))

  classified = apply(predictions, 1, function(x) {
    if (x[1] > 0.5)
      return("no")
    return("yes")
  })

  accuracy = sum(test_set$S == classified) / length(test_set$S)
  return(accuracy)
}
```

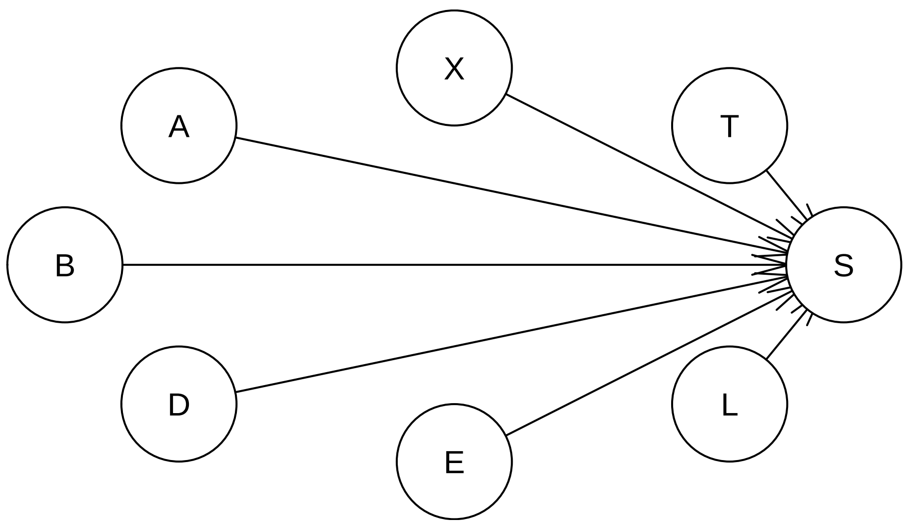
```
bn_naive_bayes = model2network("[S][A|S][T|S][L|S][B|S][E|S][X|S][D|S]")
plot(bn_naive_bayes)
```



```r
bn_net_10 = bn.fit(bn_naive_bayes, tr[1:10,], method = "bayes")
bn_net_20 = bn.fit(bn_naive_bayes, tr[1:20,], method = "bayes")
bn_net_50 = bn.fit(bn_naive_bayes, tr[1:50,], method = "bayes")
bn_net_100 = bn.fit(bn_naive_bayes, tr[1:100,], method = "bayes")
bn_net_1000 = bn.fit(bn_naive_bayes, tr[1:1000,], method = "bayes")
bn_net_2000 = bn.fit(bn_naive_bayes, tr[1:2000,], method = "bayes")

acc_10 = test_network(bn_net_10, te)
acc_20 = test_network(bn_net_20, te)
acc_50 = test_network(bn_net_50, te)
acc_100 = test_network(bn_net_100, te)
acc_1000 = test_network(bn_net_1000, te)
acc_2000 = test_network(bn_net_2000, te)

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




```r
bn_net_10_rev = bn.fit(bn_naive_bayes, tr[1:10,], method = "bayes")
bn_net_20_rev = bn.fit(bn_naive_bayes, tr[1:20,], method = "bayes")
bn_net_50_rev = bn.fit(bn_naive_bayes, tr[1:50,], method = "bayes")
bn_net_100_rev = bn.fit(bn_naive_bayes, tr[1:100,], method = "bayes")
bn_net_1000_rev = bn.fit(bn_naive_bayes, tr[1:1000,], method = "bayes")
bn_net_2000_rev = bn.fit(bn_naive_bayes, tr[1:2000,], method = "bayes")

acc_10_rev = test_network(bn_net_10_rev, te)
acc_20_rev = test_network(bn_net_20_rev, te)
acc_50_rev = test_network(bn_net_50_rev, te)
acc_100_rev = test_network(bn_net_100_rev, te)
acc_1000_rev = test_network(bn_net_1000_rev, te)
acc_2000_rev = test_network(bn_net_2000_rev, te)
```

```r
# Accuracy stays the same and does not increase having more training points
# Even gets slightly worse for more training points
print(c(acc_10, acc_20, acc_50, acc_100, acc_1000, acc_2000))
```

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

```r
# Here we see that the accuracy actually increases with more training points
print(c(acc_10_rev, acc_20_rev, acc_50_rev, acc_100_rev, acc_1000_rev, acc_2000_rev))
```

```
## [1] 0.488 0.488 0.693 0.680 0.698 0.695
```

```r
# Discussion
# The NB classifier only needs to estimate the parameters for distributions of the form
# p(C) and P(A_i|C) where C is the class variable and A_i is a predictive attribute. The
# alternative model needs to estimate p(C) and P(C|A_1,...,A_n). Therefore, it requires
# more data to obtain reliable estimates (e.g. many of the parental combinations may not
# appear in the training data and this is actually why you should use method="bayes", to
# avoid zero relative frequency counters).This may hurt performance when little learning
# data is available. This is actually observed in the experiments above. However, when
# the size of the learning data increases, the alternative model should outperform NB, because
# the latter assumes that the attributes are independent given the class whereas the former
# does not. In other words, note that p(C|A_1,...,A_n) is proportional to P(A_1,...,A_n|C) p(C)
# by Bayes theorem. NB assumes that P(A_1,...,A_n|C) factorizes into a product of factors
# p(A_i|C) whereas the alternative model assumes nothing. The NB's assumption may hurt
# performance. This can be observed in the experiments.
```

Hidden Markov Model

```r
N = 10

# Defining States Z1, Z2, ..., ZN
states = paste(rep("Z", N), 1:N, sep = "")

# Defining Symbols S1, S2, ..., SN
symbols = paste(rep("S", N), 1:N, sep = "")

# Starting Probabilities
startProbs = rep(1/N, N)

# Transition Probabilities
transProbs = matrix(0, ncol = N, nrow = N)
# Staying in the current state with 0.5 probability is just die diagonal
diag(transProbs) = 0.5
# Moving to the next is also 0.5
diag(transProbs[,-1]) = 0.5
transProbs[10, 1] = 0.5

# Emission Probabilities
emissionProbs = matrix(0, ncol = N, nrow = 10)

# 0.2 For i-2 to i+2
for (i in 1:N) {
  for (j in c(3:-1)) {
    emissionProbs[((i-j)%%N)+1,i] = 0.2
  }
}

robot_hmm = initHMM(States = states,
```

```r
                    Symbols = symbols,
                    startProbs = startProbs,
                    transProbs = transProbs,
                    emissionProbs = emissionProbs)

nSim = 100

simulatedStates = simHMM(robot_hmm, nSim)
```

```r
custom_forward = function(hmm, observations) {

  Z = matrix(NA, ncol=length(hmm$States), nrow=length(observations))

  Z[1,] = hmm$emissionProbs[, observations[1]] * hmm$startProbs

  for (t in 2:length(observations)) {
    Z[t, ] = hmm$emissionProbs[, observations[t]] * (Z[t-1,] %*% hmm$transProbs)
  }

  return(t(Z))
}

custom_backward = function(hmm, observations) {

  Z = matrix(NA, ncol=length(hmm$States), nrow=length(observations))

  Z[length(observations),] = 1

  for (t in ((length(observations)-1):0)) {
    for (state in 1:length(hmm$States)) {
     Z[t, state] = sum(Z[t+1,] * hmm$emissionProbs[,observations[t+1]] * transProbs[state,])
    }
  }

  return(t(Z))
}
```

```r
alpha = exp(forward(robot_hmm, simulatedStates$observation))
alpha_custom = custom_forward(robot_hmm, simulatedStates$observation)

alpha[,1:10]
```

```
##         index
## states     1     2     3       4       5       6       7       8        9
##     Z1  0.00 0.000 0e+00 0.00000 4.0e-06 0.0e+00 2.8e-07 6.8e-08 1.08e-08
##     Z2  0.00 0.000 0e+00 0.00000 0.0e+00 0.0e+00 0.0e+00 2.8e-08 9.60e-09
##     Z3  0.00 0.000 0e+00 0.00000 0.0e+00 0.0e+00 0.0e+00 0.0e+00 2.80e-09
##     Z4  0.00 0.000 0e+00 0.00000 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.00e+00
##     Z5  0.00 0.000 0e+00 0.00000 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.00e+00
##     Z6  0.02 0.002 2e-04 0.00000 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.00e+00
##     Z7  0.02 0.004 6e-04 0.00000 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.00e+00
##     Z8  0.02 0.004 8e-04 0.00000 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.00e+00
##     Z9  0.02 0.000 4e-04 0.00012 1.2e-05 1.2e-06 0.0e+00 0.0e+00 0.00e+00
##     Z10 0.02 0.000 0e+00 0.00004 1.6e-05 2.8e-06 4.0e-07 4.0e-08 0.00e+00
```

```
##       index
## states       10
##    Z1  1.08e-09
##    Z2  2.04e-09
##    Z3  1.24e-09
##    Z4  2.80e-10
##    Z5  0.00e+00
##    Z6  0.00e+00
##    Z7  0.00e+00
##    Z8  0.00e+00
##    Z9  0.00e+00
##    Z10 0.00e+00
```

```
alpha_custom[,1:10]
```

```
##        [,1]  [,2]  [,3]    [,4]    [,5]    [,6]    [,7]    [,8]     [,9]
##  [1,] 0.00 0.000 0e+00 0.00000 4.0e-06 0.0e+00 2.8e-07 6.8e-08 1.08e-08
##  [2,] 0.00 0.000 0e+00 0.00000 0.0e+00 0.0e+00 0.0e+00 2.8e-08 9.60e-09
##  [3,] 0.00 0.000 0e+00 0.00000 0.0e+00 0.0e+00 0.0e+00 0.0e+00 2.80e-09
##  [4,] 0.00 0.000 0e+00 0.00000 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.00e+00
##  [5,] 0.00 0.000 0e+00 0.00000 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.00e+00
##  [6,] 0.02 0.002 2e-04 0.00000 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.00e+00
##  [7,] 0.02 0.004 6e-04 0.00000 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.00e+00
##  [8,] 0.02 0.004 8e-04 0.00000 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.00e+00
##  [9,] 0.02 0.000 4e-04 0.00012 1.2e-05 1.2e-06 0.0e+00 0.0e+00 0.00e+00
## [10,] 0.02 0.000 0e+00 0.00004 1.6e-05 2.8e-06 4.0e-07 4.0e-08 0.00e+00
##          [,10]
##  [1,] 1.08e-09
##  [2,] 2.04e-09
##  [3,] 1.24e-09
##  [4,] 2.80e-10
##  [5,] 0.00e+00
##  [6,] 0.00e+00
##  [7,] 0.00e+00
##  [8,] 0.00e+00
##  [9,] 0.00e+00
## [10,] 0.00e+00
```

```
beta = exp(backward(robot_hmm, simulatedStates$observation))
beta_custom = custom_backward(robot_hmm, simulatedStates$observation)

beta[,1:10]
```

```
##       index
## states            1            2            3            4            5
##    Z1  0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##    Z2  0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##    Z3  0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##    Z4  0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##    Z5  0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##    Z6  1.207872e-80 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##    Z7  4.117181e-80 1.207872e-79 0.000000e+00 2.207453e-78 0.000000e+00
##    Z8  2.909308e-80 2.909308e-79 1.207872e-78 9.350542e-78 2.207453e-77
##    Z9  0.000000e+00 2.195000e-79 1.701436e-78 1.207872e-77 7.143089e-77
##    Z10 0.000000e+00 4.935636e-80 4.935636e-79 4.935636e-78 4.935636e-77
```

```
##       index
## states            6            7            8            9           10
##    Z1  3.953100e-76 2.728182e-75 1.503265e-74 7.041884e-74 2.350633e-73
##    Z2  1.224918e-76 1.224918e-75 1.224918e-74 7.990765e-74 4.691251e-73
##    Z3  0.000000e+00 0.000000e+00 5.217307e-75 4.258411e-74 3.299514e-73
##    Z4  0.000000e+00 0.000000e+00 9.588965e-76 9.588965e-75 9.588965e-74
##    Z5  0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##    Z6  0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##    Z7  0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##    Z8  0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##    Z9  2.207453e-76 7.041884e-76 0.000000e+00 0.000000e+00 0.000000e+00
##    Z10 4.935636e-76 2.207453e-75 7.041884e-75 2.350633e-74 0.000000e+00
```

```r
beta_custom[,1:10]
```

```
##               [,1]         [,2]         [,3]         [,4]         [,5]
##  [1,] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [2,] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [3,] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [4,] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [5,] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [6,] 1.207872e-80 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [7,] 4.117181e-80 1.207872e-79 0.000000e+00 2.207453e-78 0.000000e+00
##  [8,] 2.909308e-80 2.909308e-79 1.207872e-78 9.350542e-78 2.207453e-77
##  [9,] 0.000000e+00 2.195000e-79 1.701436e-78 1.207872e-77 7.143089e-77
## [10,] 0.000000e+00 4.935636e-80 4.935636e-79 4.935636e-78 4.935636e-77
##               [,6]         [,7]         [,8]         [,9]        [,10]
##  [1,] 3.953100e-76 2.728182e-75 1.503265e-74 7.041884e-74 2.350633e-73
##  [2,] 1.224918e-76 1.224918e-75 1.224918e-74 7.990765e-74 4.691251e-73
##  [3,] 0.000000e+00 0.000000e+00 5.217307e-75 4.258411e-74 3.299514e-73
##  [4,] 0.000000e+00 0.000000e+00 9.588965e-76 9.588965e-75 9.588965e-74
##  [5,] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [6,] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [7,] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [8,] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##  [9,] 2.207453e-76 7.041884e-76 0.000000e+00 0.000000e+00 0.000000e+00
## [10,] 4.935636e-76 2.207453e-75 7.041884e-75 2.350633e-74 0.000000e+00
```

State Space Models