CRF Tutorial

Ling-Yun Wu

2019-11-30

Contents

1 Markov Random Field

In this section, we considered a Markov chain example. We represented this Markov chain model by a CRF object and generate the samples by using the sampling functions provided in CRF package. Finally, we learned a new Markov random field (MRF) model from the generated samples.

1.1 Build Markov chain model

First we imported the CRF package:

```
library(CRF)
```

We set the parameters for Markov chain model:

```
n.nodes <- 10
n.states <- 2
prior.prob <- c(0.8, 0.2)
trans.prob <- matrix(0, nrow=2, ncol=2)
trans.prob[1,] <- c(0.95, 0.05)
trans.prob[2,] <- c(0.05, 0.95)
```

The Markov chain consists of 10 nodes and there are 2 states for each node. The prior probability is

```
prior.prob
```

```
## [1] 0.8 0.2
```

and the transition probability is

```
trans.prob
```

```
##      [,1] [,2]
## [1,] 0.95 0.05
## [2,] 0.05 0.95
```

Then we constructed the adjacent matrix of chain:

```
adj <- matrix(0, n.nodes, n.nodes)
for (i in 1:(n.nodes-1))
```

```
{
    adj[i, i+1] <- 1
}
```

Note that the adjacent matrix will be automatically symmetrized when used to build the CRF object, therefore only the upper (or lower) triangular matrix is need here.

```
adj
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    0    1    0    0    0    0    0    0    0     0
##  [2,]    0    0    1    0    0    0    0    0    0     0
##  [3,]    0    0    0    1    0    0    0    0    0     0
##  [4,]    0    0    0    0    1    0    0    0    0     0
##  [5,]    0    0    0    0    0    1    0    0    0     0
##  [6,]    0    0    0    0    0    0    1    0    0     0
##  [7,]    0    0    0    0    0    0    0    1    0     0
##  [8,]    0    0    0    0    0    0    0    0    1     0
##  [9,]    0    0    0    0    0    0    0    0    0     1
## [10,]    0    0    0    0    0    0    0    0    0     0
```

Now we can build the CRF object for Markov chain model:

```
mc <- make.crf(adj, n.states)
```

and set the parameters:

```
mc$node.pot[1,] <- prior.prob
for (i in 1:mc$n.edges)
{
  mc$edge.pot[[i]] <- trans.prob
}
```

1.2 Generate samples

We generated 10000 samples from the Markov chain model and displayed the first 10 samples:

```
mc.samples <- sample.chain(mc, 10000)
mc.samples[1:10, ]
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    1    1    1    1    1    1    1    1    1     1
##  [2,]    1    1    1    1    1    2    2    2    2     2
##  [3,]    1    1    1    1    1    1    1    1    1     1
##  [4,]    1    1    2    1    1    1    1    1    1     1
##  [5,]    1    1    1    1    1    1    1    1    1     1
##  [6,]    1    1    2    2    2    2    2    2    2     2
##  [7,]    1    1    1    1    1    1    1    1    1     1
##  [8,]    1    1    1    1    1    1    1    1    1     1
##  [9,]    1    1    2    2    2    1    1    1    1     1
## [10,]    2    2    2    2    2    2    2    2    2     2
```

1.3 Learn Markov random field model from MC data

In order to learn Markov random field model from generated data, we first built another CRF object:

```
mrf.new <- make.crf(adj, n.states)
```

and created the paramter structure:

```
mrf.new <- make.features(mrf.new)
mrf.new <- make.par(mrf.new, 4)
```

We only need 4 paramters in the MRF model, one for prior probability and three for transition probability, since the probabilities are summed to one.

```
mrf.new$node.par[1,1,1] <- 1
for (i in 1:mrf.new$n.edges)
{
    mrf.new$edge.par[[i]][1,1,1] <- 2
    mrf.new$edge.par[[i]][1,2,1] <- 3
    mrf.new$edge.par[[i]][2,1,1] <- 4
}
```

Then we trained the model using `train.mrf` function:

```
mrf.new <- train.mrf(mrf.new, mc.samples)
```

After training, we can check the parameter values:

```
mrf.new$par
```

```
## [1]  0.08836713  0.76612234 -0.57852938 -0.65817530
```

We normalized the potentials in MRF to make it more like probability:

```
mrf.new$node.pot <- mrf.new$node.pot / rowSums(mrf.new$node.pot)
mrf.new$edge.pot[[1]] <- mrf.new$edge.pot[[1]] / rowSums(mrf.new$edge.pot[[1]])
```

Now we can check the learned prior probability

```
mrf.new$node.pot[1,]
```

```
## [1] 0.5220774 0.4779226
```

and transition probability

```
mrf.new$edge.pot[[1]]
```

```
##           [,1]      [,2]
## [1,] 0.7932539 0.2067461
## [2,] 0.3411496 0.6588504
```

2 Conditional Random Field

In this section, we generated hidden Markov Model (HMM) samples based on the Markov chain samples in previous section. Then we learned a conditional random field (CRF) model from the HMM data.

2.1 Generate samples

Suppose that the Markov chain can not be directly observed. There are 4 observation states and the observation probability (emmision probability) is given as follows:

```
emmis.prob <- matrix(0, nrow=2, ncol=4)
emmis.prob[1,] <- c(0.59, 0.25, 0.15, 0.01)
emmis.prob[2,] <- c(0.01, 0.15, 0.25, 0.59)
emmis.prob
```

```
##      [,1] [,2] [,3] [,4]
## [1,] 0.59 0.25 0.15 0.01
## [2,] 0.01 0.15 0.25 0.59
```

We simulated the observation data from Markov chain samples:

```
hmm.samples <- mc.samples
hmm.samples[mc.samples == 1] <- sample.int(4, sum(mc.samples == 1), replace = TRUE, prob=emmis.prob[1,])
hmm.samples[mc.samples == 2] <- sample.int(4, sum(mc.samples == 2), replace = TRUE, prob=emmis.prob[2,])
hmm.samples[1:10,]
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    1    1    1    1    1    1    1    1    3     1
##  [2,]    3    3    2    1    1    4    4    4    3     2
##  [3,]    2    3    1    1    1    2    1    2    3     1
##  [4,]    2    3    3    1    1    1    3    1    1     1
##  [5,]    1    2    1    1    1    1    1    3    2     1
##  [6,]    3    1    4    4    4    4    2    4    3     4
##  [7,]    1    3    1    2    2    3    1    1    1     2
##  [8,]    3    4    3    1    2    1    1    3    2     1
##  [9,]    1    1    4    4    2    2    1    1    1     1
## [10,]    3    4    4    4    2    4    4    3    3     2
```

2.2 Learn conditional random field model from HMM data

Now we try to learn a CRF model from HMM data. We first built another CRF object:

```
crf.new <- make.crf(adj, n.states)
```

and created the paramter structure:

```
crf.new <- make.features(crf.new, 5, 1)
crf.new <- make.par(crf.new, 8)
```

The major difference between CRF and MRF is that we have 5 node features now, instead of 1 constant feature in MRF model. The first node feature is the constant feature as in MRF model, and the other 4 node features correspond to observation states respectively. The number of edge feature is still one. We now need eight paramters, one for prior probability, three for transition probability, and four for emmision probability.

```
crf.new$node.par[1,1,1] <- 1
for (i in 1:crf.new$n.edges)
{
    crf.new$edge.par[[i]][1,1,] <- 2
    crf.new$edge.par[[i]][1,2,] <- 3
    crf.new$edge.par[[i]][2,1,] <- 4
}
crf.new$node.par[,1,2] <- 5
crf.new$node.par[,1,3] <- 6
crf.new$node.par[,1,4] <- 7
crf.new$node.par[,1,5] <- 8
```

We prepared the node features and the edge features, which are need for training:

```
hmm.nf <- lapply(1:dim(hmm.samples)[1], function(i) matrix(1, crf.new$n.nf, crf.new$n.nodes))
for (i in 1:dim(hmm.samples)[1])
{
    hmm.nf[[i]][2, hmm.samples[i,] != 1] <- 0
    hmm.nf[[i]][3, hmm.samples[i,] != 2] <- 0
```

```
    hmm.nf[[i]][4, hmm.samples[i,] != 3] <- 0
    hmm.nf[[i]][5, hmm.samples[i,] != 4] <- 0
}
hmm.ef <- lapply(1:dim(hmm.samples)[1], function(i) matrix(1, crf.new$n.ef, crf.new$n.edges))
```

Then we trained the model using `train.crf` function:

```
crf.new <- train.crf(crf.new, mc.samples, hmm.nf, hmm.ef)
```

After training, we can check the parameter values:

```
crf.new$par
```

```
## [1]  1.5889883  0.2317296 -2.9349339 -2.7374454  3.8539029  0.2489202 -0.7737314
## [8] -4.2458192
```

With trained CRF model, we can infer the hidden states given the observations:

```
hmm.infer <- matrix(0, nrow=dim(hmm.samples)[1], ncol=dim(hmm.samples)[2])
for (i in 1:dim(hmm.samples)[1])
{
  crf.new <- crf.update(crf.new, hmm.nf[[i]], hmm.ef[[i]])
  hmm.infer[i,] <- decode.chain(crf.new)
}
```

The inferred result was compared with the true hidden states:

```
sum(hmm.infer != mc.samples)
```

```
## [1] 3372
```

2.3 Use other inference methods in the training

The default inference method used in the `train.mrf` and `train.crf` functions is `infer.chain`, which can only handle chain-structured graphs. We can provide the preferred inference method when calling the training functions. For example, use the loopy brief propagation algorithm:

```
crf.new <- train.crf(crf.new, mc.samples, hmm.nf, hmm.ef, infer.method = infer.lbp)
```

In a more complicated way, we can redefine the functions for calculating the negative log-likelihood, i.e., the functions `mrf.nll` and `crf.nll`, respectively.