Hybrid-EP + Junction Tree Demo

Eric Chuu

4/27/2022

Computing the log normalizing constant of the G-Wishart density

Consider the G-Wishart density, which has the following form:

$$f\left(\Omega \mid G\right) \propto \left|\Omega\right|^{(\delta-2)/2} \exp\left(-\mathrm{tr}\left(\Omega\Lambda\right)/2\right),$$

Here, Ω, Λ are $p \times p$ non-negative definite matrices, and $\delta > 2$ is the degrees of freedom. The normalizing constant is typically intractable, and the algorithm that we develop in this package targets this quantity.

Case 1: $p = 30$**.**

Define the following problem settings. We first randomly initialize the 30×30 adjacency matrix representation of the graph G.

```
library(graphml)
set.seed(1234)
p = 30
Adj = matrix(rbinom(p^2,1,0.15), p, p)
Adj = Adj + t(Adj)
diag(Adj) = 0
Adj[Adj==1]=0
Adj[Adj==2]=1
diag(Adj) = 1
```

Next, we define the following degrees of freedom and scale matrix.

```
b = 500
Y = matrix(rnorm(p*500), nrow = 500, ncol = p)
Lambda  = t(Y)%*%Y
```

Next, we compute the edge matrix which is needed for the junction tree part of the Hybrid-EP + JT algorithm.

```
# compute the edge matrix
EdgeMat = graphml::getEdgeMat(Adj)
```

Finally, we compute the Hybrid-EP + JT appoximation to the log normalizing constant.

```
start_time <- Sys.time()
hyb_approx = graphml::hybridJT(Adj, EdgeMat, b, Lambda, 1000)
end_time <- Sys.time()
hyb_approx
```

```
## [1] -7450.699
```

```
end_time - start_time
```

```
## Time difference of 0.01939416 secs
```

We compare this to Atay's method.

```
start_time <- Sys.time()
atay_approx = BDgraph::gnorm(Adj, b, Lambda, 1000)
end_time <- Sys.time()
atay_approx
```

```
## [1] -7428.426
```

```
end_time - start_time
```

```
## Time difference of 0.08423519 secs
```

We observe that the two estimates are quite close to each other, and that the Hybrid-EP + JT method is faster.

Case 2: $p = 60$**.**

```
library(graphml)
set.seed(1234)
p = 30
Adj = matrix(rbinom(p^2,1,0.15), p, p)
Adj = Adj + t(Adj)
diag(Adj) = 0
Adj[Adj==1]=0
Adj[Adj==2]=1
diag(Adj) = 1
b = 500
Y = matrix(rnorm(p*500), nrow = 500, ncol = p)
Lambda  = t(Y)%*%Y
# compute the edge matrix
EdgeMat = graphml::getEdgeMat(Adj)
```

Finally, we compute the Hybrid-EP + JT approximation to the log normalizing constant.

```
start_time <- Sys.time()
hyb_approx = graphml::hybridJT(Adj, EdgeMat, b, Lambda, 1000)
end_time <- Sys.time()
hyb_approx
```

```
## [1] -7450.699
```

```
end_time - start_time
```

```
## Time difference of 0.01950097 secs
```

We compare this to Atay's method.

```
start_time <- Sys.time()
atay_approx = BDgraph::gnorm(Adj, b, Lambda, 1000)
end_time <- Sys.time()
atay_approx
```

```
## [1] -7428.426
```

```
end_time - start_time
```

```
## Time difference of 0.08013892 secs
```

We observe that Atay's method fails to give a finite estimate, while the Hybrid-EP + JT quickly produces a sensible approximation.