Discretization

Alexis Akira Toda*

First version: June 2, 2019
Updated: September 27, 2023

1 Introduction

This note explains the functionalities in the MATLAB package `discretization`. The user should use these files at their own responsibility. Whenever you use these codes for your research, please cite at least Farmer and Toda (2017), and if possible also Tanaka and Toda (2013, 2015).

In Section 3, I also explain how users may write their own discretization codes using these functionalities.

If you have a stochastic process that you would like to discretize but do not know how to do it yourself, please let me know. Enjoy!

2 Package content

The package includes several directories, namely

- Core functionalities,
- Discretize CIR,
- Discretize Gaussian Mixture AR,
- Discretize nonparametric distribution,
- Discretize SV,
- Discretize VAR,
- Nonparametric Gaussian quadrature,
- Subroutines.

Below, I describe each directory in some detail.

*Department of Economics, University of California San Diego. Email: alexis.akira.toda@gmail.com.

2.1 Core functionalities

This directory contains two files:

- `discreteApproximation.m`
- `entropyObjective.m`

`entropyObjective.m` defines the objective function for solving the maximum entropy problem, which is the dual objective function of the minimization of the Kullback-Leibler divergence. `discreteApproximation.m` solves the actual optimization problem.

These two files are fully tested and should work well. There is no need to modify these files. If you are going to write your own discretization code as in Section 3, you will likely use `discreteApproximation.m`. See the code for usage.

2.2 Subroutines

This directory contains four files, `allcomb2.m`, `fclencurt.m`, `GaussHermite.m`, and `legpts.m`. All of these files are written by third parties. `allcomb2.m` is used to construct the Cartesian product. `fclencurt.m` constructs the Clenshaw-Curtis quadrature. `GaussHermite.m` constructs the Gauss-Hermite quadrature. `legpts.m` constructs the Gauss-Legendre quadrature.

2.3 Discretize nonparametric distribution

This directory contains files to discretize a nonparametric distribution given some centered moments. The usage is

```
[x,p] = discreteNP(N,cMoments,q)
```

where

- `N` is the number of grid points,
- `cMoments` is the vector of centered moments,
- `q` is the initial guess of the distribution (optional),
- `x` is the vector of grid points,
- `p` is the vector of probabilities.

If `q` is uncpecified, it will use a Gaussian distribution.

2.4 Discretize VAR

This directory contains files to discretize the VAR(1) process

$$y_t = b + By_{t-1} + \eta_t,$$

where $\eta_t \sim N(0, \Psi)$. `discreteVAR.m` is the main file for discretization. The usage is

```
[P,X] = discreteVAR(b,B,Psi,Nm,nMoments,method,nSigmas)
```

where

- `b, B, Psi` are as above,
- `Nm` is the number of grid points in each dimension of y_t,
- `nMoments` is the number of moments to try to match (between 1 and 4),
- `method` is the quadrature method, either `'even'` (evenly-spaced grid), `'quadrature'` (Gauss-Hermite quadrature), or `'quantile'` (grid based on quantiles of normal distribution),
- `nSigmas` is the grid spacing parameter (optional),
- `P` is the transition probability matrix,
- `X` is the grid for y_t.

Leland Farmer wrote this code. It is thoroughly tested and should work well. Although the code is written to work for any dimension, I recommend limiting to VARs with up to three variables due to curse of dimensionality. As discussed in Farmer and Toda (2017), the method `'quantile'` is poor and should not be used.[1] If the spectral radius of B is not large (say less than 0.5), then `'quadrature'` is best. Otherwise I recommend using `'even'`.

2.5 Discretize Gaussian mixture AR

This directory contains files to discretize the AR(p) process

$$x_t - \mu = a_1(x_{t-1} - \mu) + \cdots + a_p(x_{t-p} - \mu) + \varepsilon_t,$$

where ε_t is an i.i.d. Gaussian mixture shock. `discreteGMAR.m` is the main file for discretization. The usage is

```
[P,X] = discreteGMAR(mu,A,pC,muC,sigmaC,Nm,nMoments,method,nSigmas)
```

where

- `mu` is μ above,
- `A` is the vector $A = [a_1, \ldots, a_p]$,
- `pC`, `muC`, `sigmaC` are the vectors of proportion, mean, and standard deviation of Gaussian mixture components,
- `Nm` is the number of grid points in each dimension,
- `nMoments` is the number of moments to try to match (between 1 and 4),
- `method` is the quadrature method, either `'even'` (evenly-spaced grid), `'gauss-legendre'` (Gauss-Legendre quadrature), `'clenshaw-curtis'` (Clenshaw-Curtis quadrature), `'gauss-hermite'` (Gauss-Hermite quadrature), or `'GMQ'` (Gaussian quadrature for Gaussian mixture),
- `nSigmas` is the grid spacing parameter (optional).

[1] This option is included only because we did not have a prior on the choice of the grid when we were writing Farmer and Toda (2017) and we have experimented with various grids.

Although the code is written to work for any p, I recommend limiting to $p \leq 3$ due to curse of dimensionality. For the quadrature method, unless you have strong reasons to choose otherwise, I recommend using `'even'`. If the AR(p) process is not so persistent (largest eigenvalue less than 0.5), `'GMQ'` is theoretically most accurate.

`GaussianMixtureQuadrature.m` constructs the nodes and weights of the Gaussian quadrature when the weighting function is a Gaussian mixture, using the Golub-Welsh algorithm. If you specify only one component, then it returns the Gaussian quadrature for the normal distribution.

2.6 Discretize SV

The directory contains files to discretize the AR(1) process with log AR(1) stochastic volatility

$$\begin{aligned} y_t &= \lambda y_{t-1} + u_t, & u_t &\sim N(0, \mathrm{e}^{x_t}), \\ x_t &= (1-\rho)\mu + \rho x_{t-1} + \varepsilon_t, & \varepsilon_t &\sim N(0, \sigma^2). \end{aligned}$$

`discreteSV.m` is the main file for discretization. The usage is

```
[P,yxGrids] = discreteSV(lambda,rho,sigmaU,sigmaE,Ny,Nx,method,nSigmaY)
```

where

- `lambda, rho, sigmaE` are λ, ρ, σ above,
- `sigmaU` is the unconditional standard deviation of $u_t = \mathrm{e}^{x_t}$,
- `Ny, Nx` are number of grid points to discretize the y_t, x_t processes,
- `method` is the same as `discreteVAR.m` (`'even'` is recommended, especially when the persistence ρ is large; never use `'quantile'`),
- `nSigmaY` is the grid spacing parameter (optional).
- `P` is the transition probability matrix,
- `yxGrids` is the grid for y_t, x_t (in this order).

2.7 Discretize CIR

Consider the Cox-Ingersoll-Ross (CIR) model

$$c\,\mathrm{d}r_t = a(b - r_t)\,\mathrm{d}t + \sigma\sqrt{r_t}\,\mathrm{d}W_t.$$

According to the Wikipedia page,[2] the CIR model has the following conditional density:

$$f(r_t; r_0, a, b, \sigma) = c\mathrm{e}^{-u-v}\left(\frac{v}{u}\right)^{q/2} I_q(2\sqrt{uv}),$$

[2] https://en.wikipedia.org/wiki/Cox-Ingersoll-Ross_model

where $q = 2ab/\sigma^2 - 1$, $u = cr_0 e^{-at}$, $v = cr_t$, and I_q is the modified Bessel function of the first kind with order q. Furthermore, the conditional mean and variance of r_t are

$$\begin{aligned} \mathrm{E}\left[r_t \,|\, r_0\right] &= r_0 e^{-at} + b(1 - e^{-at}), \\ \mathrm{Var}\left[r_t \,|\, r_0\right] &= \frac{\sigma^2}{a}(1 - e^{-at})(r_0 e^{-at} + b/2). \end{aligned}$$

Since the conditional density and two moments are known in closed-form, it is straightforward to discretize it using the Farmer and Toda (2017) method. The file `discreteCIR.m` does this. The usage is

```
[P,X] = discreteCIR(a,b,sigma,Delta,N,Coverage,method)
```

where

- `a,b,sigma` are a, b, σ above,
- `Delta` is the length of one period Δt,
- `N` is the number of grid points, and
- `Coverage` is the coverage rate of the grid (optional; default is 0.999),
- `method` is the grid choice, either `'even'` or `'exponential'` (optional; default is `'exponential'`).

2.8 Nonparametric Gaussian quadrature

This directory contains a file to discretize a single nonparametric distribution directly from the data. The main file is `NPGQ.m`. The usage is

```
[x,w] = NPGQ(data,N)
```

where

- `data` is your data,
- `N` is the number of grid points,
- `x` is the vector of grid points,
- `w` is the vector of probabilities assigned to each grid point.

This code is unrelated to the maximum entropy method in Tanaka and Toda (2013, 2015) and Farmer and Toda (2017). Instead, it is based on Toda (2021).

`NPGQ.m` should be useful if you would like to calibrate a distribution from data without imposing any parametric assumptions. One warning is that the method (based on the Golub-Welsh algorithm) requires the population distribution to have moments of order up to $2N$. Therefore if you know data is positive-valued, input `log(data)` instead of `data` and construct your grid by `exp(x)`.

3 Writing your own discretization code

Writing your own discretization code is relatively simple, especially when the stochastic process is one-dimensional. Essentially, all you need to do is to

- define the grid and quadrature formula,
- loop over grid points to match conditional moments using maximum entropy.

As an example, look at `discreteAR.m` in the `Discretize VAR` directory. This code discretizes the AR(1) process with Gaussian shocks

$$x_t = (1-\rho)\mu + \rho x_{t-1} + \varepsilon_t.$$

There are three important parts in the code. The first is lines 40–55, where we define the nodes `X` and weights `W` of the quadrature formula. The simplest way is to use the evenly-spaced grid as in lines 42–43, in which case `W` is just a vector of ones (normalization does not matter). If, for example, for some process you want to use an exponential grid (as in lines 36–41 of `discreteCIR.m`), then because $\mathrm{d}\log x = \mathrm{d}x/x$, you should set `W = 1./X`.

The second important part is lines 63–69, where we define the conditional distribution for each grid point. Note that line 63 computes the conditional mean of the process, and line 68 defines the prior probability q using the normal density and the precomputed weight `W`.

The third part is the actual minimization of the Kullback-Leibler divergence in line 75–113. Here I wrote a code to match up to 4 moments so it is a bit long, but if you are fine with just matching 2 moments, you can write a much shorter code (see, for example, lines 47–66 of `discreteCIR.m`, which matches only 2 moments).

References

Leland E. Farmer and Alexis Akira Toda. Discretizing nonlinear, non-Gaussian Markov processes with exact conditional moments. *Quantitative Economics*, 8(2):651–683, July 2017. doi:10.3982/QE737.

Ken'ichiro Tanaka and Alexis Akira Toda. Discrete approximations of continuous distributions by maximum entropy. *Economics Letters*, 118(3):445–450, March 2013. doi:10.1016/j.econlet.2012.12.020.

Ken'ichiro Tanaka and Alexis Akira Toda. Discretizing distributions with exact moments: Error estimate and convergence analysis. *SIAM Journal on Numerical Analysis*, 53(5):2158–2177, 2015. doi:10.1137/140971269.

Alexis Akira Toda. Data-based automatic discretization of nonparametric distributions. *Computational Economics*, 57:1217–1235, April 2021. doi:10.1007/s10614-020-10012-6.