Documentation for `cupid_matching`

Table of contents

1 Introduction

`cupid_matching` is a Python-based package that solves, simulates, and estimates separable matching problems with perfectly transferable utility. ***At this stage, it only allows for bipartite, one-to-one matching (e.g., the heterosexual marriage market).***

`cupid_matching` has four central functionalities:

1. to solve for the stable matching using our Iterative Projection Fitting Procedure (IPFP) in variants of the Choo and Siow (2006) model[1]:

with or without singles, homoskedastic or heteroskedastic; and for a class of nested logit models;

2. to estimate the parameters of separable models with linear surplus and entropy using a minimum distance estimator;
3. to estimate the parameters of semilinear Choo and Siow models using a Poisson GLM estimator;
4. to demonstrate solving and estimating the Choo and Siow model with a Streamlit interactive app (see a demo version here).

The package builds on Choo and Siow's pioneering paper (*Journal of Political Economy* 2006) and on my joint work with Alfred Galichon (especially our 2022 *Review of Economic Studies* paper and a forthcoming paper in the *Journal of Applied Econometrics*).

1.1 Citation

If `cupid_matching` is a significant contributor to your research, the following citation is recommended:

- **Salanié, Bernard (2023):** "*cupid_matching:* solving and estimating separable matching models", https://pypi.org/project/cupid_matching.

[1]See also here for a package in R that solves for equilibrium in the basic version of the Choo and Siow model.

1.2 Installation

`cupid_matching` works with Python versions ≥ 3.10. To install [or update] it, run the following code:

```
pip install [-U] cupid_matching
```

The source code is available on GitHub.

1.3 Using teh package

To import functions or classes from `cupid_matching`, use e.g.

```
from cupid_matching.min_distance import estimate_semilinear_mde
```

1.4 Examples

To get a feel for what the package can do, you can download the source here or from Github by

```
git clone https://github.com/bsalanie/cupid_matching.git
```

and run the `example_choo_siow.py` script. It estimates the parameters of the Choo and Siow homoskedastic model using the minimum distance estimator and the Poisson-GLM estimator.

To run the Streamlit app, run `streamlit run cupid_streamlit.py` in the folder where the file `cupid_streamlit.py` is. The app will open in your browser at `http://localhost:8501/`. *Make sure that you are using the version of `streamlit` that you installed with `pip install cupid_matching`*. Alternatively, try the demo version here.

1.5 Features

- Stable matching for bipartite, one-to-one models with perfectly transferable utility.
- Solves for stable matching using the Iterative Projection Fitting Procedure (IPFP) in Choo and Siow (2006) models: with or without singles, homoskedastic or heteroskedastic
- Solves for stable matching using the Iterative Projection Fitting Procedure (IPFP) for a class of nested logit models

- Support for models with different error structures from user-provided code
- Estimates the parameters of separable models using minimum distance
- Estimates parameters of semilinear Choo and Siow models using Poisson-GLM
- Provides a Streamlit interactive app for the homoskedastic Choo and Siow (2006) model with singles.

1.6 Warnings

- Many of these models (including all variants of Choo and Siow) rely heavily on logarithms and exponentials: it is easy to generate examples where numeric instability sets in. As a consequence, the numeric versions of the minimum distance estimator (which use numerical derivatives) are not recommended.
- The bias-corrected minimum distance estimator, 'corrected", may have a larger mean-squared error and/or introduce numerical instabilities.
- The estimated variance of the estimators assumes that the observed matching was sampled at the household level, and that all sampling weights are equal.

2 Background

As `cupid_matching` presently deals with only bipartite, one-to-one models, its functionality will be described using the heterosexual marriage market as an example.

The two sides are men, denoted as m, and women, denoted as w. Each man m has an observed discrete-valued *type or group* x and each woman w has an observed discrete-valued *type/group* y.

For now, we allow for singles: a *match* (or *household*) consists of one man and one woman; a single man; or a single woman. We return to the case when only matches are observed in a later section.

A *matching* specifies who matches whom and who stays single. The data only give us *matching patterns*, the aggregation of the matching by observed types.

The *joint utility* from a match is generated partly by these observed characteristics, and partly by unobserved heterogeneity. The *stable matching* maximizes the total joint utility, summed over all matches.

2.1 Notation

The following notation will be used throughout the description of `cupid_matching`'s functionality.

- m: a man
- w: a woman
- x: a man's observed discrete-valued type
- y a woman's observed discrete-valued type
- Φ: the joint utility matrix
- ε, η: the unobserved heterogeneity vectors
- $\varepsilon_{m0}, \eta_{0w}$ the utility of a single man/woman
- μ_{xy}: the number of matches between men of type x and women of type y
- μ_{x0}: The number of single men of type x
- μ_{0y}: the number of single women of type y
- N_i: the total number of individuals
- N_h: the total number of households
- $\mathcal{E}$: the generalized entropy function
- $\hat{\mu}$: the observed matching patterns
- α: the parameter vector for the generalized entropy function $\mathcal{E}^\alpha$
- β: the parameter vector for the joint utility matrix Φ^β
- $\lambda = (\alpha, \beta)$.
- $(\phi^1, \phi^2, \dots, \phi^K)$: the basis functions for the linear joint utility.

2.2 Primitives

There are n_x men of type $x = 1, \dots, X$ and m_y women of type $y = 1, \dots, Y$. The joint utility created by the match of a man m of type x and a woman w of type y takes the following, *separable* form:

$$\tilde{\Phi}_{mw} \equiv \Phi_{xy} + \varepsilon_{my} + \eta_{xw}.$$

A single man m has utility ε_{m0} and a single woman w has utility η_{0w}. We call $\Phi = (\Phi_{xy})$ the *joint utility matrix.*

The modeler chooses the distributions of the vectors $\varepsilon = (\varepsilon_{m0}, \varepsilon_{m1}, \dots, \varepsilon_{mY})$ and $\eta = (\eta_{0w}, \eta_{1w}, \dots, \eta_{Xw})$.

2.3 The stable matching

We denote μ_{xy} the number of matches between men of type x and women of type y. μ_{x0} represents the number of single men of type x and μ_{0y} represents the number of single women of type y.

Feasibility requires that these numbers be consistent with the *margins* n_x and m_y:

$$\sum_{y=1}^{Y} \mu_{xy} + \mu_{x0} = \; n_x \;\text{ for all } x$$

and

$$\sum_{x=1}^{X} \mu_{xy} + \mu_{0y} = \; m_y \;\text{ for all } y.$$

The total number of individuals is:

$$N_i = \sum_{x=1}^{X} n_x + \sum_{y=1}^{Y} m_y$$

and the total number of households is:

$$N_h \equiv \sum_{x=1}^{X}\sum_{y=1}^{Y} \mu_{xy} + \sum_{x=1}^{X} \mu_{x0} + \sum_{y=1}^{Y} \mu_{0y} = N_i - \sum_{x=1}^{X}\sum_{y=1}^{Y} \mu_{xy}.$$

Galichon-Salanié (*REStud* 2022) shows that in large markets, if the vectors ε and η have full support, the stable matching is the unique solution to the strictly convex problem

$$\max_{\mu} \left(\sum_{x=1}^{X}\sum_{y=1}^{Y} \mu_{xy}\Phi_{xy} + \mathcal{E}(\mu; n, m) \right)$$

under the feasibility constraints.

In this expression, the *generalized entropy function* $\mathcal{E}$ depends on the assumed distributions of the ε and η random vectors. As an example, in the original Choo and Siow (2006) model, the ε and η terms are iid draws from a standard type I extreme value (Gumbel) distribution. Then

$$\mathcal{E}(\mu; n, m) = \sum_{x=1}^{X} \sum_{y=1}^{Y} \mu_{xy} \log \frac{\mu_{xy}^2}{n_x m_y} + \mu_{x0} \log \frac{\mu_{x0}}{n_x} + \mu_{0y} \log \frac{\mu_{0y}}{m_y}.$$

The files `choo_siow.py`, `choo_siow_no_singles.py`,`choo_siow_gender_heteroskedastic.py`, `choo_siow_heteroskedastic.py`, and `nested_logit.py` provide `EntropyFunctions` objects that compute the generalized entropy and at least its first derivative for, respectively:

1. The original Choo and Siow (2006) model;
2. the same model without singles (to be used when only couples are observed);
3. an extension of 1. that allows for a scale parameter τ for the distribution of η;
4. an extension of 1. that has type-dependent scale parameters σ_x and τ_y (with the normalization $\sigma_1 = 1$);
5. A two-layer nested logit model in which singles are in their own nest and the user chooses the structure of the other nests.

Users are welcome to code `EntropyFunctions` objects for different distributions of the unobserved heterogeneity terms.

2.4 Solving for the stable matching

Given any joint surplus matrix Φ_{xy}, margins n_x and m_y, and a generalized entropy function $\mathcal{E}$, one would like to compute the stable matching patterns $(\mu_{xy}, \mu_{x0}, \mu_{0y})$.

For all five classes of models above, this can be done efficiently using the IPFP algorithm in Galichon-Salanié (*REStud* 2022). It is coded in `ipfp_solvers.py` for the four Choo and Siow variants and in `model_classes.py` for the nested logit.

For example, given a Numpy array Φ that is an (X, Y) matrix and a number $\tau > 0$, the following code solves for the stable matching in the gender-heteroskedastic Choo and Siow model with singles (model 3.):

```
import numpy as np
from cupid_matching.ipfp_solvers import ipfp_gender_homoskedastic_solver

solution = ipfp_gender_heteroskedastic_solver(Phi, n, m, tau)
mus, error_x, error_y = solution
muxy, mux0, mu0y = mus.muxy, mus.mux0, mus.mu0y
```

The `mus` above is an instance of a `Matching` object (defined in `matching_utils.py`). The matrix `mus.muxy` has the number of couples in each (x, y) cell at the stable matching; the vectors `mus.mux0` and `mus.mu0y` contain the numbers of single men and women of each type.

The vectors `error_x` and `error_y` are estimates of the precision of the solution (see the code in `ipfp_solvers.py`).

2.5 Estimating the joint utility matrix

Given observed matching patterns μ, a class of generalized entropy functions $\mathcal{E}^\alpha$, and a class of joint surplus functions (Φ^β), one would like to estimate the parameter vector $\lambda = (\alpha, \beta)$.

The package provides two estimators which are described extensively in this paper:

1. The minimum distance estimator in `min_distance.py`
2. The Poisson estimator in `poisson_glm.py`, for the linear-surplus Choo and Siow homoskedastic model only.

At this stage, `cupid_matching` only allows for linear models of the joint surplus:

$$\Phi^\beta_{xy} = \sum_{k=1}^{K} \phi^k_{xy} \beta_k$$

where the *basis functions* $(\phi^1, \dots, \phi^K)$ are chosen by the analyst. It also requires the generalized entropy function to have derivatives with respect to μ that are either parameter-free or linear in a vector of unknown parameters α:

$$\frac{\partial \mathcal{E}}{\partial \mu}(\mu; n, m) = e^0(\mu; n, m) + e(\mu, n, m) \cdot \alpha.$$

2.5.1 The minimum distance estimator

The minimum distance estimator finds the parameter vectors α and β that minimize a well-chosen norm of the vector with $X \times Y$ components

$$\phi_{xy} \cdot \beta + \frac{\partial \mathcal{E}}{\partial \mu}(\mu; n, m).$$

The efficient choice of the norm is the inverse of the variance-covariance matrix of this vector. Under this efficient choice, if the model is correctly specified then the minimized value of the objective function is a χ^2 statistic.

2.5.2 The Poisson-GLM estimator

For the Choo and Siow 2006 specification, an alternative estimator can be used. One can rewrite the estimating equations as the first-order conditions of the maximum-likelihood estimator of a Poisson model with two-way fixed effects. This model has an augmented set of parameters: in addition to β, the parameter vector also contains the expected utilities of the various groups of men and women at the stable matching.

2.5.3 Application

The file `example_choosiow.py` has a demo of the minimum distance and Poisson estimators on the Choo and Siow (2006).

For other models, the minimum distance estimator works as follows. Given

1. An observed matching model stored in a `Matching` object `mus`
2. An `EntropyFunction` object `entropy_model` that allows for p parameters in α
3. An (X, Y, K) Numpy array of basis functions `phi_bases`, the following code estimates the parameters of the model:

```
mde_results = estimate_semilinear_mde(mus, phi_bases, entropy_model)
mde_results.print_results(n_alpha=p)
```

The `mde_results` object contains the estimated α and β, their estimated variance-covariance and standard errors, and the results of the specification test.

2.6 Without singles

Sometimes the data only contain matches (no singles). In this case, the user should use the `MatchingNoSingles` class instead of `Matching`. The adding up constraints are

$$\sum_{y=1}^{Y} \mu_{xy} = n_x \text{ for all } x \; ; \; \sum_{x=1}^{X} \mu_{xy} = m_y \text{ for all } y,$$

and the total numbers of men and women must be equal,

$$\sum_{x=1}^{X} n_x = \sum_{y=1}^{Y} m_y.$$

When singles are not observed, only the double differences of the Φ matrix can be identified. This rules out using any basis function that only depends on x, or only on y (an error will be raised if you try). More precisely, the identifying equation is

$$D_2 \Phi = D_2 \frac{\partial \mathcal{E}}{\partial \mu}(\mu; n, m)$$

where D_2 is the double differencing $(X \times Y, X \times Y)$ matrix; for a vector (u_{xy}),

$$(D_2 u)_{xy} = u_{xy} - \frac{1}{Y} \sum_{t=1}^{Y} u_{xt} - \frac{1}{X} \sum_{z=1}^{X} u_{zy} + \frac{1}{XY} \sum_{z=1}^{X} \sum_{t=1}^{Y} u_{zt}.$$

2.7 minimum distance estimation without singles

The matrix D_2 has rank $r < X \times Y$. To avoid zero eigenvalues in the variance that is used in the optimal weighting matrix, the program premultiplies D_2 by a random (r, XY) matrix A. The identifying equations become

$$A D_2 \Phi = A D_2 \frac{\partial \mathcal{E}}{\partial \mu}(\mu; n, m).$$

The minimum distance estimator must be called with the `no_singles=True` option, and with an `EntropyFunction` object that is set up for the no-singles model. The module `choo_siow_no_singles` provides the appropriate function for the Choo and Siow homoskedastic model.

2.8 Poisson estimation without singles

The Poisson-GLM estimator is easily adapted to the Choo and Siow homoskedastic model without singles. Since only the sums of utilities $u_x + v_y$ are identified in the absence of singles, one of their values must be normalized. The program sets the value u_1 of the expected utility of the first group of men to zero. All other utility estimates must be interpreted accordingly.

---

3 Tutorials

- `example_choosiow.py` shows how to run minimum distance and Poisson estimators on a Choo and Siow homoskedastic model.
- `example_choosiow_no_singles.py` shows how to run minimum distance and Poisson estimators on a Choo and Siow homoskedastic model without singles.
- `example_nestedlogit.py` shows how to run minimum distance estimators on a two-layer nested logit model.

4 Release Notes

4.1 version 1.2 (November 29, 2023)

- incorporates models without singles for both MDE and Poisson; example in `example_choo_siow_no_singles.py`.

4.2 version 1.1.3 (November 8, 2023)

- fixed URL of Streamlit app.

4.3 version 1.1.2 (November 7, 2023)

- improved the Streamlit app, now in two files: `cupid_streamlit.py` and `cupid_streamlit_utils.py`.

4.4 version 1.1.1

- improved documentation
- the package now relies on my utilities package `bs_python_utils`. The `VarianceMatching` class in `matching_utils,py` is new; this should be transparent for the user.

4.5 version 1.0.8

- deleted spurious print statement.

4.6 version 1.0.7

- fixed error in bias-correction term.

4.7 version 1.0.6

- corrected typo.

4.8 version 1.0.5

- simplified the bias-correction for the minimum distance estimator in the Choo and Siow homoskedastic model.

4.9 version 1.0.4

- added an optional bias-correction for the minimum distance estimator in the Choo and Siow homoskedastic model, to help with cases when the matching patterns vary a lot across cells.
- added two complete examples: `example_choosiow.py` and `example_nestedlogit.py`.