Mathematical Background: Conditional Moment Conditions

1. Problem Formulation

1.1 Conditional Moment Restrictions

This library addresses estimation problems defined by **conditional moment restrictions (CMR)** of the form:

$$\mathbb{E}[\psi(X, Y; \theta) \mid Z] = 0 \quad P_Z\text{-almost surely}$$

where:

- (X, Y, Z) are random variables observed from an i.i.d. sample $\{(x_i, y_i, z_i)\}_{i=1}^n$
- $X \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$ represents treatments or covariates
- $Y \in \mathcal{Y} \subseteq \mathbb{R}^{d_y}$ represents outcomes or responses
- $Z \in \mathcal{Z} \subseteq \mathbb{R}^{d_z}$ represents instrumental variables
- $\theta \in \Theta$ are the parameters of interest
- $\psi : \mathcal{X} \times \mathcal{Y} \times \Theta \to \mathbb{R}^k$ is a known moment function

1.2 Instrumental Variable Regression

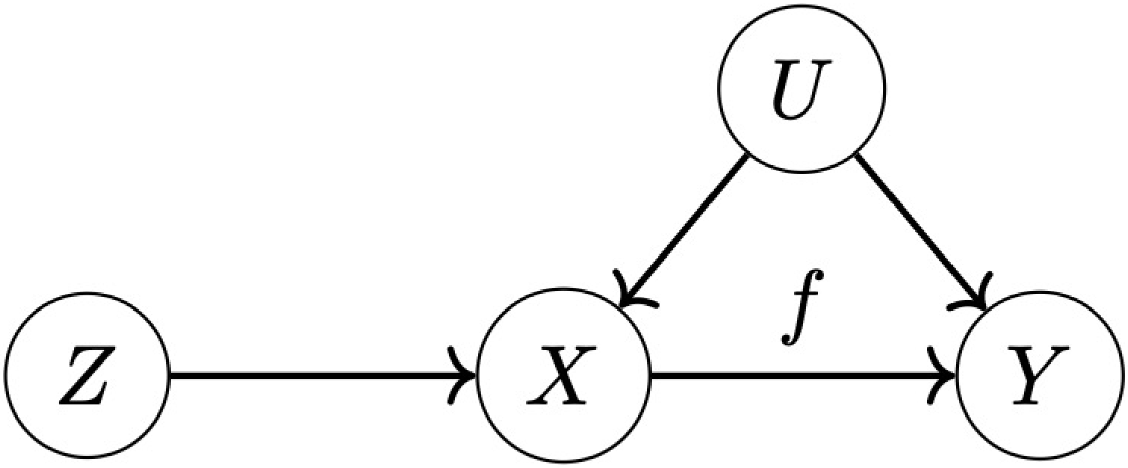
The canonical application is **instrumental variable (IV) regression**, where:

- We seek to estimate a structural function $f_\theta : \mathcal{X} \to \mathcal{Y}$
- The moment function takes the form: $\psi(X, Y; \theta) = f_\theta(X) - Y$
- The CMR becomes: $\mathbb{E}[f_\theta(X) - Y \mid Z] = 0$

This problem arises when:

- There exists unobserved confounding U between X and Y
- An instrument Z satisfies:
    1. **Relevance**: Z is correlated with X
    2. **Exclusion**: Z affects Y only through X
    3. **Exogeneity**: Z is independent of U

The causal structure is



1.3 Unconditional Moment Restrictions

When Z is not specified or the conditional restriction reduces to an unconditional one, we have the **moment restriction (MR)** problem:

$$\mathbb{E}[\psi(X, Y; \theta)] = 0$$

This is a special case where Z is trivial or integrated out.

---

2. Mathematical Framework

2.1 Dual Formulation via f-Divergences

The estimators in this library are best understood through the lens of **Generalized Empirical Likelihood (GEL)**, which formulates estimation as a constrained optimization problem on the space of probability measures.

2.1.1 The Primal Problem: Constrained Optimization Consider the unconditional moment restriction $\mathbb{E}[\psi(X, Y; \theta)] = 0$. The classical **Empirical Likelihood (EL)** estimator seeks a discrete probability distribution $p = (p_1, \ldots, p_n)$ supported on the sample points that maximizes the likelihood (or equivalently minimizes the negative log-likelihood) subject to the moment constraints holding under p:

$$\begin{aligned}
\min_{p \in \mathbb{R}^n} \quad & \sum_{i=1}^{n} -\log(np_i) \\
\text{subject to} \quad & \sum_{i=1}^{n} p_i \psi(x_i, y_i; \theta) = 0 \\
& \sum_{i=1}^{n} p_i = 1, \quad p_i \geq 0
\end{aligned}$$

The Parameter Explosion Problem: The primal problem attempts to estimate n parameters (the weights p_i) plus the parameters of interest θ. As $n \to \infty$, the number of nuisance parameters grows with the sample size, making direct optimization computationally infeasible and statistically challenging ("Neyman-Scott problem").

2.1.2 The Dual Solution: Saddle Point Formulation

To avoid this explosion, we use convex duality. This generalizes beyond EL to the class of **f-divergences**. For a convex function $f : \mathbb{R}_+ \to \mathbb{R}$ with $f(1) = 0$, we minimize the divergence between p and the uniform distribution $u_n = (1/n, \dots, 1/n)$:

$$\min_{p \in \Delta_n} \sum_{i=1}^{n} \frac{1}{n} f\left(np_i\right) \quad \text{s.t.} \quad \mathbb{E}_p[\psi(\theta)] = 0$$

By introducing Lagrange multipliers $\lambda \in \mathbb{R}^k$ for the moment constraints and using the Fenchel conjugate $f^*(t) = \sup_u \{tu - f(u)\}$, the problem converts to a dual formulation that depends only on λ and θ. This reduces the dimensionality from $O(n)$ to $O(k)$, leading to the **Generalized Empirical Likelihood (GEL)** saddle-point objective:

$$\min_{\theta \in \Theta} \sup_{\lambda \in \mathbb{R}^k} \left\{ -\frac{1}{n} \sum_{i=1}^{n} f^*(\lambda^\top \psi(x_i, y_i; \theta)) \right\}$$

This formulation is computationally efficient (k is fixed) and forms the basis for the modern machine learning approaches (VMM, FGEL, KMM) used in this library, which generalize λ from a vector to a function $h(Z)$.

---

3. Estimation Methods

3.1 Unconditional Moment Restrictions

3.1.1 Ordinary Least Squares (OLS)

Objective:

$$\hat{\theta}_{\text{OLS}} = \arg\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \|\psi(x_i, y_i; \theta)\|^2$$

Use case: Simple baseline, assumes no confounding.

3.1.2 Generalized Method of Moments (GMM)

Objective:

$$\hat{\theta}_{\text{GMM}} = \arg\min_{\theta \in \Theta} \bar{\psi}_n(\theta)^\top W_n \bar{\psi}_n(\theta)$$

where:

- $\bar{\psi}_n(\theta) = \frac{1}{n} \sum_{i=1}^{n} \psi(x_i, y_i; \theta)$
- W_n is a weighting matrix (typically $\hat{\Sigma}^{-1}$ where $\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} \psi_i \psi_i^\top$)

Two-step procedure:

1. Obtain initial estimate $\tilde{\theta}$
2. Compute W_n using $\tilde{\theta}$
3. Re-optimize with the optimal weighting matrix

Parameters: `reg_param` for regularization $W_n = (\hat{\Sigma} + \alpha I)^{-1}$

3.1.3 Generalized Empirical Likelihood (GEL) The library implements the dual formulation derived in Section 2.1.

Saddle-Point Objective:

$$\min_{\theta\in\Theta}\max_{\lambda\in\mathbb{R}^k} \mathcal{L}_{GEL}(\theta, \lambda) = \min_{\theta\in\Theta}\max_{\lambda\in\mathbb{R}^k} \left\{ -\frac{1}{n}\sum_{i=1}^{n} f^*(\lambda^\top \psi_i(\theta)) - \alpha\|\lambda\|^2 \right\}$$

Supported Divergences:

1. **Chi-squared (Euclidean EL)**:
   - Primal: Minimize $\sum(np_i - 1)^2$.
   - Dual Conjugate: $f^*(t) = \frac{1}{2}(t+1)^2$.
   - Connection: Closely related to Continuous Updating GMM (CUE).
   - Code: `divergence='chi2'`
2. **Kullback-Leibler (Exponential Tilting)**:
   - Primal: Minimize $\sum p_i \log(np_i)$.
   - Dual Conjugate: $f^*(t) = e^t$.
   - Properties: Often more stable than EL; weights are strictly positive.
   - Code: `divergence='kl'`
3. **Log-Euclidean (Empirical Likelihood)**:
   - Primal: Maximize $\sum \log(np_i)$.
   - Dual Conjugate: $f^*(t) = -\log(1-t)$.
   - Constraint: Requires $\lambda^\top \psi_i < 1$ for all i.
   - Code: `divergence='log'`

Implementation: The `GeneralizedEL` class handles the optimization. The inner maximization over λ is solved via Newton's method (if `optim='lbfgs'`) or gradient ascent, while the outer minimization over θ uses the chosen optimizer (e.g., Adam, OAdam).

3.1.4 Kernel Method of Moments (KMM) **Formulation**: Minimizes an MMD-based objective with entropy regularization.

$$\min_{\theta}\max_{\nu,\eta,h} \left\{ \mathbb{E}_P[h(X,Y,Z)] + c - \gamma\mathbb{E}_Q[f^*(\frac{1}{\gamma}(h(X,Y,Z) + c - \nu^\top\psi(\theta)))] - \frac{\lambda}{2}\|h\|^2_{\mathcal{H}} \right\}$$

where:

- P is the empirical distribution
- Q is a reference distribution (estimated via KDE)
- $h \in \mathcal{H}$ is an RKHS function on the joint space
- $\nu \in \mathbb{R}^k$ are dual variables for moments
- $c \in \mathbb{R}$ is a normalization constant

- $\gamma > 0$ is the entropy regularization parameter
- $\lambda > 0$ is the RKHS regularization parameter

Implementation details:

- Uses random Fourier features for h (dimension controlled by `n_random_features`)
- Reference distribution via KDE with bandwidth `kde_bandwidth`
- Samples from reference distribution: `n_reference_samples`
- Product kernel on (X, Y, Z) space

Parameters:

- `entropy_reg_param`: γ (main regularization)
- `rkhs_reg_param`: λ for RKHS norm
- `n_random_features`: Dimension D of RFF approximation
- `n_reference_samples`: Number of samples from reference distribution
- `kde_bandwidth`: Bandwidth for KDE

3.2 Conditional Moment Restrictions

The modern methods below generalize the GEL principle to **conditional** restrictions $\mathbb{E}[\psi \mid Z] = 0$. Instead of a single vector λ, we now learn a **function** of the instruments, effectively performing an "infinite" number of moment checks or learning the "optimal instrument" in a data-driven way.

3.2.1 Maximum Moment Restrictions (MMR)

Objective: Directly minimizes the MMD between $\psi(\theta)$ and 0 conditional on Z.

$$\hat{\theta}_{\mathrm{MMR}} = \arg\min_{\theta \in \Theta} \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \psi(x_i, y_i; \theta)^\top K_{ij}^Z \psi(x_j, y_j; \theta)$$

where $K_{ij}^Z = k(z_i, z_j)$ is the kernel Gram matrix for instruments.

Advantages:

- Simple, no dual optimization
- Convex in θ for linear models
- Direct interpretation as MMD

Limitations:

- Can be sensitive to kernel choice
- No regularization on moment function

3.2.2 Sieve Minimum Distance (SMD) **Formulation**: Uses sieve basis (polynomials or splines) to approximate the conditional expectation.

Let $\{b_j(z)\}_{j=1}^J$ be a basis for $L^2(P_Z)$. Approximate:

$$\mathbb{E}[\psi(X, Y; \theta) \mid Z = z] \approx \sum_{j=1}^{J} \beta_j b_j(z)$$

Objective:

$$\min_{\theta,\beta} \frac{1}{n} \sum_{i=1}^{n} \left\| \psi(x_i, y_i; \theta) - \sum_{j=1}^{J} \beta_j b_j(z_i) \right\|^2$$

Properties:

- Classical semiparametric approach
- Theoretical guarantees under regularity conditions
- Basis dimension $J = J_n \to \infty$ as $n \to \infty$ (slower than n)

3.2.3 Variational Method of Moments (VMM) VMM generalizes GMM by replacing the fixed weighting matrix with a **learned instrument function** $h(Z)$ in an RKHS.

The Adversarial Game: Instead of minimizing a static quadratic form, VMM solves a minimax game where an "adversary" h tries to maximize the correlation with the moment violations, while the model θ tries to minimize it.

Kernel Version (VMM-kernel): The adversary is a function in a vector-valued RKHS $\mathcal{H}^k$. The objective simplifies analytically to:

$$\min_{\theta} \max_{\boldsymbol{h} \in \mathcal{H}^k} \left\{ \frac{2}{n} \sum_{i=1}^{n} \psi(x_i, y_i; \theta)^\top \boldsymbol{h}(z_i) - \|\boldsymbol{h}\|_{\mathcal{H}^k}^2 \right\}$$

This leads to a closed-form solution for the optimal h, resulting in a generalized quadratic form involving the kernel matrix inverse:

$$\min_{\theta} \psi(\theta)^\top K^Z (K^Z K^Z + \alpha I)^{-1} K^Z \psi(\theta)$$

Neural Version (VMM-neural): Here, the adversary is a neural network $g_\phi(z)$. This recovers the **DeepGMM** method.

$$\min_{\theta} \max_{\phi} \left\{ \frac{2}{n} \sum_{i=1}^{n} \psi(\theta)^\top g_\phi(z) - \mathbb{E}[g_\phi(z)^\top g_\phi(z)] \right\}$$

3.2.4 Functional Generalized Empirical Likelihood (FGEL) FGEL is the direct conditional generalization of GEL. It replaces the vector λ in the GEL objective with a function $h(Z)$.

The Concept: We seek the "worst-case" re-weighting of the data (defined by $h(Z)$) that satisfies the conditional moment restrictions. The f-divergence acts as a regularizer on this re-weighting.

Objective:

$$\min_{\theta} \max_{h \in \mathcal{H}} \left\{ -\frac{1}{n} \sum_{i=1}^{n} f^*(h(z_i)^\top \psi(x_i, y_i; \theta)) - \frac{\alpha}{2} \|h\|_{\mathcal{H}}^2 \right\}$$

Kernel vs. Neural:

- **FGEL-kernel**: h lies in an RKHS. By the representer theorem, $h(z) = \sum \alpha_i k(z, z_i)$, reducing the problem to finding coefficients α_i.
- **FGEL-neural**: h is parametrized by a neural network. This allows for scalable estimation with large datasets where the kernel matrix would be too large to invert.

3.2.5 Kernel Method of Moments - Neural (KMM-neural)

KMM-neural combines the Maximum Mean Discrepancy (MMD) principle with neural network test functions.

The Logic: It minimizes the MMD between the model distribution and a reference distribution, where the "test function" for the MMD is learned adversarially to detect moment violations.

Objective:

$$\min_{\theta} \max_{\nu, c, h} \left\{ \mathbb{E}_P[h] + c - \gamma \mathbb{E}_Q \left[f^* \left(\frac{1}{\gamma} (h + c - \nu_\phi(z)^\top \psi(\theta)) \right) \right] - \text{Reg}(h, \nu) \right\}$$

Key Innovation: Unlike FGEL which effectively reweights the *empirical* distribution, KMM introduces a reference distribution Q (e.g., from a Kernel Density Estimate). This stabilizes training and provides better finite-sample performance by ensuring the solution stays close to the data manifold. It uses **Random Fourier Features (RFF)** to scale the RKHS components to large datasets.

---

4. API Reference

4.1 High-Level Interface

`estimation()` Main entry point for training any estimator with automatic hyperparameter search.

```python
from cmr import estimation

trained_model, stats = estimation(
    model,                      # torch.nn.Module
    train_data,                # dict: {'t': ndarray, 'y': ndarray, 'z': ndarray}
    moment_function,           # callable: (model_output, y) -> torch.Tensor
    estimation_method,         # str: method name
    estimator_kwargs=None,     # dict: method-specific parameters
    hyperparams=None,          # dict: hyperparameter search space
    sweep_hparams=True,        # bool: whether to search hyperparameters
    validation_data=None,      # dict: validation set
    val_loss_func=None,        # callable or str: validation metric
    normalize_moment_function=True,  # bool: normalize moments
    verbose=True               # bool: print progress
)
```

Arguments:

Parameter	Type	Description
`model`	`torch.nn.Module`	Parametric model f_θ containing parameters to estimate
`train_data`	`dict`	Training data with keys `'t'` (treatments), `'y'` (outcomes), `'z'` (instruments)
`moment_function`	`callable`	Function $(f_\theta(x), y) \mapsto \psi(x, y; \theta) \in \mathbb{R}^k$
`estimation_method`	`str`	Method identifier (see table below)
`estimator_kwargs`	`dict`	Method-specific parameters (overrides defaults)
`hyperparams`	`dict`	Hyperparameter search space as `{param: [val1, val2, ...]}`
`sweep_hparams`	`bool`	If `True`, performs grid search over `hyperparams`
`validation_data`	`dict`	Validation set (if `None`, uses `train_data`)
`val_loss_func`	`callable` or `str`	Validation metric: custom function, `'mmr'`, `'hsic'`, or `'moment_violation'`
`normalize_moment_function`	`bool`	If `True`, pretrains and normalizes moment components to unit variance
`verbose`	`bool` or `int`	Verbosity level: `False`/`0`, `True`/`1`, or `2`

Returns:

- `trained_model`: Trained PyTorch model (best from hyperparameter search)
- `stats`: Dictionary containing:
  - `'models'`: List of all trained models
  - `'val_loss'`: Validation losses for each hyperparameter setting
  - `'hyperparam'`: List of hyperparameter configurations
  - `'best_index'`: Index of best model
  - `'train_stats'`: Training statistics for each model

Estimation Methods:

`estimation_method`	Problem Type	Description
`'OLS'`	Unconditional	Ordinary least squares
`'GMM'`	Unconditional	Generalized method of moments
`'GEL'`	Unconditional	Generalized empirical likelihood
`'KMM'`	Unconditional	Kernel method of moments
`'SMD'`	Conditional	Sieve minimum distance
`'MMR'`	Conditional	Maximum moment restrictions
`'VMM-kernel'`	Conditional	Variational MM with kernel
`'VMM-neural'`	Conditional	Variational MM with neural network

estimation_method	Problem Type	Description
'FGEL-kernel'	Conditional	Functional GEL with kernel
'FGEL-neural'	Conditional	Functional GEL with neural network
'KMM-neural'	Conditional	Kernel MM with neural instrument function

4.2 Low-Level Interface

For fine-grained control without hyperparameter search:

```
from cmr.methods.kmm_neural import KMMNeural

estimator = KMMNeural(
    model=model,
    moment_function=moment_function,
    entropy_reg_param=10.0,
    reg_param=1e-2,
    n_random_features=2000,
    verbose=True
)

estimator.train(train_data, validation_data)
trained_model = estimator.model
parameters = estimator.get_trained_parameters()
```

4.3 Data Format

All data must be provided as dictionaries with NumPy arrays:

```
train_data = {
    't': np.array([[...]]),  # shape: (n_samples, d_t)
    'y': np.array([[...]]),  # shape: (n_samples, d_y)
    'z': np.array([[...]])   # shape: (n_samples, d_z) or None for unconditional
}
```

For unconditional moment restrictions, set `'z': None`.

4.4 Moment Function

The moment function must have signature:

```
def moment_function(model_output: torch.Tensor, y: torch.Tensor) -> torch.Tensor:
    """
    Args:
        model_output: Model predictions f_theta(x), shape (n, d_output)
        y: Observed outcomes, shape (n, d_y)
```

```python
    Returns:
        Moment values psi(x, y; theta), shape (n, k)
    """
    return model_output - y  # Example: IV regression
```

Common moment functions:

1. **IV Regression**: `lambda pred, y: pred - y`
2. **Quantile Regression**: `lambda pred, y: (y - pred) * (tau - (y < pred).float())`
3. **Vector-valued**: Return tensor with `shape[1] = k` for k moment conditions

4.5 Validation Metrics

Built-in validation metrics (string identifiers):

- `'moment_violation'`: $\frac{1}{n}\sum_{i=1}^{n}\|\psi(x_i, y_i; \theta)\|^2$ (unconditional)
- `'mmr'`: $\frac{1}{n^2}\sum_{i,j}\psi_i^\top K_{ij}^Z \psi_j$ (conditional, requires Z)
- `'hsic'`: Hilbert-Schmidt Independence Criterion between $\psi(\theta)$ and Z (conditional)

Custom validation functions:

```python
def custom_val_loss(model: torch.nn.Module, val_data: dict) -> float:
    """
    Args:
        model: Trained PyTorch model
        val_data: Validation data dict with keys 't', 'y', 'z'

    Returns:
        Scalar validation loss (lower is better)
    """
    # Your custom validation logic
    return loss_value
```

4.6 Common Workflow

```python
import torch
import numpy as np
from cmr import estimation

# 1. Generate/load data
train_data = {'t': X_train, 'y': Y_train, 'z': Z_train}
val_data = {'t': X_val, 'y': Y_val, 'z': Z_val}
test_data = {'t': X_test, 'y': Y_test, 'z': Z_test}

# 2. Define model architecture
model = torch.nn.Sequential(
    torch.nn.Linear(dim_t, 50),
    torch.nn.LeakyReLU(),
    torch.nn.Linear(50, 20),
    torch.nn.LeakyReLU(),
    torch.nn.Linear(20, dim_y)
```

```
)

# 3. Define moment function
def moment_function(pred, y):
    return pred - y

# 4. Train with hyperparameter search
trained_model, stats = estimation(
    model=model,
    train_data=train_data,
    moment_function=moment_function,
    estimation_method='KMM-neural',
    validation_data=val_data,
    verbose=True
)

# 5. Evaluate
predictions = trained_model(torch.Tensor(test_data['t']))
mse = torch.mean((predictions - torch.Tensor(test_data['y']))**2)
```

---

5. Theoretical Guarantees

5.1 Identification

For the CMR to identify θ, we require:

- **Completeness**: The conditional expectation operator $\mathbb{E}[\cdot \mid Z]$ is injective
- **Global identification**: θ is the unique solution to $\mathbb{E}[\psi(X, Y; \theta) \mid Z] = 0$

Under completeness, the CMR is equivalent to:

$$\int \psi(x, y; \theta) p(x, y \mid z) dx dy = 0 \quad \text{for all } z \in \mathcal{Z}$$

5.2 Consistency and Convergence Rates

Under regularity conditions:

MMR:

- Consistency: $\hat{\theta}_n \xrightarrow{p} \theta_0$
- Rate: $\|\hat{\theta}_n - \theta_0\| = O_p(n^{-r})$ where r depends on smoothness and dimension

FGEL-kernel:

- With RKHS regularization $\alpha_n = O(n^{-\gamma})$ for appropriate γ
- Achieves minimax optimal rates under smoothness assumptions

KMM:

- Combines advantages of MMD testing and moment restriction estimation
- Reference distribution improves finite-sample performance
- Convergence rate depends on RFF approximation quality and reference sample size

5.3 Asymptotic Normality

For semiparametric efficient estimation, under regularity:

$$\sqrt{n}(\hat{\theta}_n - \theta_0) \xrightarrow{d} \mathcal{N}(0, V)$$

where V is the semiparametric efficiency bound. Methods like FGEL and VMM can achieve this bound with appropriate choices of divergence and regularization.

5.4 Computational Complexity

Method	Per-Iteration Complexity	Memory
OLS/GMM	$O(nk^2)$	$O(nk)$
GEL	$O(nk)$	$O(nk)$
MMR	$O(n^2k)$	$O(n^2)$
VMM-kernel	$O(n^3k)$ (matrix inversion)	$O(n^2k)$
FGEL-kernel	$O(n^2k^2)$	$O(nk)$
VMM-neural	$O(nk \cdot \mathrm{NN})$	$O(nk)$
FGEL-neural	$O(nk \cdot \mathrm{NN})$	$O(nk)$
KMM	$O(n^2)$ (full kernel)	$O(n^2)$
KMM-neural	$O(nD + n \cdot \mathrm{NN})$	$O(nD)$

where NN denotes neural network forward/backward pass cost and D is the RFF dimension.

Scalability recommendations:

- $n < 1000$: Any method works well
- $1000 \leq n < 5000$: Use neural versions or KMM with RFF
- $n \geq 5000$: Mandatory to use neural networks or RFF approximations
- Use mini-batch training (`batch_size` parameter) for $n > 10000$

References

1. Kremer, H., Nemmour, Y., Schölkopf, B., & Zhu, J. J. (2023). Estimation Beyond Data Reweighting: Kernel Method of Moments. arXiv:2305.10898
2. Kremer, H., Zhu, J. J., Muandet, K., & Schölkopf, B. (2022). Functional Generalized Empirical Likelihood Estimation for Conditional Moment Restrictions. ICML 2022.
3. Bennett, A., Kallus, N., & Schnabel, T. (2020). Deep Generalized Method of Moments for Instrumental Variable Analysis. NeurIPS 2019.
4. Muandet, K., Fukumizu, K., Sriperumbudur, B., & Schölkopf, B. (2017). Kernel Mean Embedding of Distributions: A Review and Beyond. Foundations and Trends in Machine Learning.