Learning Conditional Expectation Operators via a Functional Newton Method

Thiago R. Ramos[1] Alek Fröhlich[2,3] Daniel Perazzo[2,3] Massimiliano Pontil[2,4]

CBJM 2026

[1]Federal University of São Carlos

[2]CSML, Istituto Italiano di Tecnologia

[3]University of Genoa

[4]University College London

Motivation: dependence beyond marginals

We want to capture the *joint behaviour* of (X, Y) beyond marginals.

Assuming $P_{X,Y} \ll P_X \otimes P_Y$, the Radon–Nikodym derivative

$$\kappa(x, y) \;=\; \frac{dP_{X,Y}}{d(P_X \otimes P_Y)}(x, y)$$

encodes *all* information about dependence.

It admits an SVD-like decomposition

$$\kappa(x, y) \;=\; 1 + \sum_{i=1}^{\infty} \sigma_i \, \phi_i^\star(x) \, \psi_i^\star(y),$$

with $\{\phi_i^\star\} \subset L^2(P_X)$, $\{\psi_i^\star\} \subset L^2(P_Y)$ orthonormal.

Why κ matters: one object, many tasks

Conditional expectation operator

$$\mathsf{E}(g)(x) \;=\; \mathbb{E}\left[g(Y) \mid X = x\right] \;=\; \int g(y)\,\kappa(x,y)\,p(y)\,dy.$$

Knowing κ gives us E, and thus:

- **Regression:** $g(y) = y \;\Rightarrow\; \mathsf{E}(g)(x) = \mathbb{E}\left[Y \mid X = x\right]$
- **Conditional CDF:** $g_t(y) = \mathbb{1}\{y \le t\} \;\Rightarrow\; \mathsf{E}(g_t)(x) = \mathbb{P}\left(Y \le t \mid X = x\right)$
- **Independence testing:** $X \perp Y \iff \kappa \equiv 1$

Goal: learn a truncated rank-d decomposition of κ.

The learning problem

Look for a rank-d model

$$\kappa_{\phi,\psi}(x,y) \;=\; 1 + \phi(x)^\top \psi(y), \qquad \phi : \mathcal{X} \to \mathbb{R}^d,\ \psi : \mathcal{Y} \to \mathbb{R}^d,$$

minimising the $L^2(P_X \otimes P_Y)$ error

$$\min_{\phi,\psi} \ \|\kappa - \kappa_{\phi,\psi}\|^2_{P_X \otimes P_Y}.$$

Key simplification. Expanding the square,

$$\|\kappa - \kappa_{\phi,\psi}\|^2 = \underbrace{\|\kappa\|^2}_{\text{const}} + \|\kappa_{\phi,\psi}\|^2_{P_X \otimes P_Y} - 2\,\mathbb{E}_{P_X \otimes P_Y}\left[\kappa(X,Y)\,\kappa_{\phi,\psi}(X,Y)\right].$$

The cross term collapses because $\kappa = dP_{X,Y}/d(P_X \otimes P_Y)$:

$$\mathbb{E}_{P_X \otimes P_Y}\left[\kappa(X,Y)\,\kappa_{\phi,\psi}(X,Y)\right] = \mathbb{E}_{P_{X,Y}}\left[\kappa_{\phi,\psi}(X,Y)\right].$$

Hence (up to a constant)

$$L(\phi,\psi) = \mathbb{E}_{P_X \otimes P_Y}\left[\kappa_{\phi,\psi}(X,Y)^2\right] - 2\,\mathbb{E}_{P_{X,Y}}\left[\kappa_{\phi,\psi}(X,Y)\right].$$

We never need to know κ — only samples from $P_{X,Y}$ and the marginals.

The idea: functional gradient descent

We minimise $L(\phi, \psi)$ *directly over functions* $\phi \in L^2(P_X)^d$, $\psi \in L^2(P_Y)^d$.

Inspired by gradient boosting (Friedman, 2001): instead of fitting parameters of a fixed model, build ϕ, ψ *iteratively as functions*, each step moving along the functional gradient of L.

Our twist. L turns out to be *quadratic in each block*:

- the pointwise Hessians w.r.t. $\phi(x)$ and $\psi(y)$ are *constant* matrices $(\Sigma_\psi, \Sigma_\phi)$;
- so we can do functional Newton instead of plain functional gradient steps;
- Newton's step has a closed form $\Rightarrow$ no inner line search, no learning rate to tune.

Gâteaux derivative + Riesz: what is a "functional gradient"?

The objective $L : L^2(P_X)^d \times L^2(P_Y)^d \to \mathbb{R}$ is a function on a Hilbert space, not on $\mathbb{R}^n$.

Gâteaux derivative. The directional derivative of L at (ϕ, ψ) in direction $h \in L^2(P_X)^d$:

$$D_\phi L(\phi, \psi)[h] \;=\; \lim_{\varepsilon \to 0^+} \frac{L(\phi + \varepsilon h, \psi) - L(\phi, \psi)}{\varepsilon}.$$

For our L, the map $h \mapsto D_\phi L(\phi, \psi)[h]$ is *linear and continuous*.

Riesz representation. Every bounded linear functional on a Hilbert space $\mathcal{H}$ is an inner product with a unique vector. So there exists a unique $\nabla_\phi L(\phi, \psi) \in L^2(P_X)^d$ with

$$D_\phi L(\phi, \psi)[h] \;=\; \langle h,\, \nabla_\phi L(\phi, \psi) \rangle_{L^2(P_X)^d} \;=\; \mathbb{E}_{P_X}\left[h(X)^\top \nabla_\phi L(\phi, \psi)(X) \right].$$

$\Rightarrow$ $\nabla_\phi L$ is a function of x, the right object to subtract from ϕ in a (functional) Newton/gradient step.

Functional calculus: gradients and Hessians

Work in $\phi \in L^2(P_X)^d$, $\psi \in L^2(P_Y)^d$. Define

$$m_\psi(x) = \mathbb{E}\left[\psi(Y) - \mathbb{E}\left[\psi(Y)\right] \mid X = x\right], \quad \Sigma_\psi = \mathbb{E}_{P_Y}\left[\psi(Y)\psi(Y)^\top\right],$$

and symmetrically m_ϕ, Σ_ϕ.

Gâteaux gradients

$$\nabla_\phi L(\phi, \psi)(x) = 2\,\Sigma_\psi\,\phi(x) - 2\,m_\psi(x), \quad \nabla_\psi L(\phi, \psi)(y) = 2\,\Sigma_\phi\,\psi(y) - 2\,m_\phi(y).$$

Diagonal Hessians are *constant multiplication operators*:

$$\nabla^2_{\phi\phi} L = 2\,\Sigma_\psi, \qquad \nabla^2_{\psi\psi} L = 2\,\Sigma_\phi.$$

$\Rightarrow$ Newton steps are available in closed form.

Estimating m_ϕ, m_ψ from samples

The Newton step

$$\psi^\star(\phi)(y) = \Sigma_\phi^{-1}\, m_\phi(y), \qquad m_\phi(y) = \mathbb{E}\left[\phi(X) - \mathbb{E}\left[\phi(X)\right] \mid Y = y\right]$$

needs the conditional expectation m_ϕ — an unknown function.

Plug-in via regression. Treating it as a multi-output supervised problem:

- targets $t_i = \phi(X_i) - \frac{1}{n}\sum_j \phi(X_j) \in \mathbb{R}^d$
- fit any regressor $r : Y \to \mathbb{R}^d$ on $\{(Y_i, t_i)\}_{i=1}^n$
- $\widehat{m}_\phi(y) \coloneqq r(y)$

Boosting analogy. Refit a fresh regressor each iteration on updated ϕ-residuals — exactly the gradient-boosting recipe, but in the functional Newton direction.

Our implementation: `sklearn` base learner (default: shallow decision tree). Linear, kernel, or neural regressors plug in without changing the algorithm.

The FSNM algorithm

Algorithm 1 Functional Spectral-Newton Method (FSNM)

Require: initial ϕ_0, ψ_0; step sizes $\eta_\phi, \eta_\psi \in (0, 1]$; iterations T

1: **for** $t = 0, \ldots, T-1$ **do**

2: $\quad \psi_{t+1}(y) \leftarrow (1 - \eta_\psi)\, \psi_t(y) + \eta_\psi\, \Sigma_{\phi,t}^{-1}\, m_{\phi,t}(y)$

3: $\quad \phi_{t+1}(x) \leftarrow (1 - \eta_\phi)\, \phi_t(x) + \eta_\phi\, \Sigma_{\psi,t+1}^{-1}\, m_{\psi,t+1}(x)$

4: **end for**

Ensure: (ϕ_T, ψ_T)

- Block coordinate *Newton*: invert the (small $d \times d$) Hessian blocks Σ_ϕ, Σ_ψ.
- $\eta = 1 \Rightarrow$ pure Newton; $\eta < 1 \Rightarrow$ relaxed / boosting-style.

FSNM as relaxed best-response & convergence

Best response. Stationarity gives the exact block minimisers

$$\psi^\star(\phi) = \Sigma_\phi^{-1} m_\phi, \qquad \phi^\star(\psi) = \Sigma_\psi^{-1} m_\psi,$$

so each FSNM update is a convex combination towards them: $\psi_{t+1} - \psi_t = \eta_\psi(\psi^\star(\phi_t) - \psi_t)$.

Contraction. Block-quadratic loss $\Rightarrow$ exact per-step decrease

$$L(\phi_t, \psi_t) - L(\phi_{t+1}, \psi_{t+1}) = \eta_\psi(2 - \eta_\psi)\, G_t^\psi + \eta_\phi(2 - \eta_\phi)\, G_t^\phi,$$

where $G_t = \frac{1}{4}\|\Sigma^{-1/2}\nabla L\|^2$ are Mahalanobis gaps to the best response.

Theorem (asymptotic stationarity)
If $\lambda I \preceq \Sigma_{\phi,t}, \Sigma_{\psi,t} \preceq \Lambda I$ uniformly, then for any $\eta \in (0, 1]$

$$G_t^\phi, G_t^\psi \to 0, \quad \|\nabla L\|_{L^2} \to 0, \quad \|\psi_{t+1} - \psi_t\|, \|\phi_{t+1} - \phi_t\| \to 0.$$

Experiments: setup

Four benchmark distributions (analytical $\mathbb{E}[Y \mid X]$ and $\mathbb{P}(Y \leq t \mid X)$):

- **LinearStudentT**: heteroscedastic Student-t with X-dependent location/scale/df.
- **EconDensity**: $Y = X^2 + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \alpha^2(1+X)^2)$.
- **GaussianMixture**: 5-component bivariate mixture (multimodal $Y \mid X$).
- **SkewNormal**: bivariate skew-normal via a shared half-normal factor.

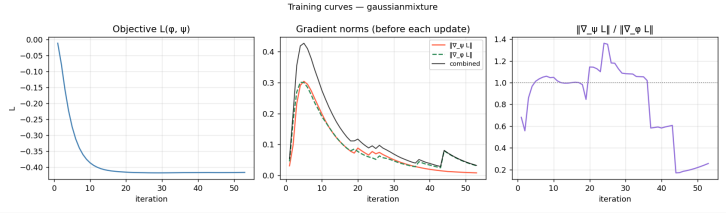
FSNM configuration. $N = 10000$, $d = 3$, $\eta_\phi = \eta_\psi = 0.1$, decision-tree base learners (`max_depth`$=\infty$, `min_samples_leaf`$=200$), canonicalisation each step, early stopping on gradient plateau.

Downstream metrics (lower is better):

- **Regression MSE**: $\mathbb{E}[Y \mid X]$ from $\kappa_{\phi,\psi}$ vs. analytical truth.
- **CDF MSE**: $\mathbb{P}(Y \leq 0 \mid X)$ from $\kappa_{\phi,\psi}$ vs. analytical truth.

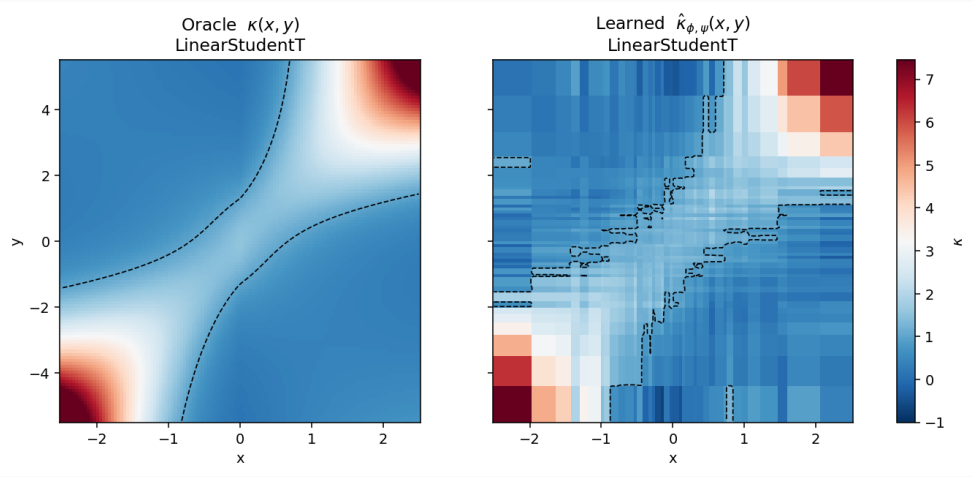
Experiments: quantitative results

Distribution	Regression MSE	CDF MSE (t=0)
LinearStudentT	0.025	0.0012
EconDensity	0.667	0.0009
GaussianMixture	0.012	0.0014
SkewNormal	0.020	0.0012



Loss decreases monotonically (contraction lemma).

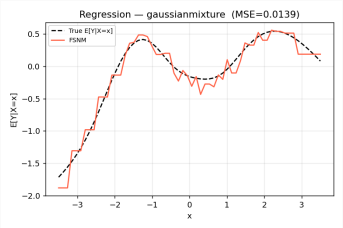
Experiments: oracle vs learned $\kappa(x, y)$

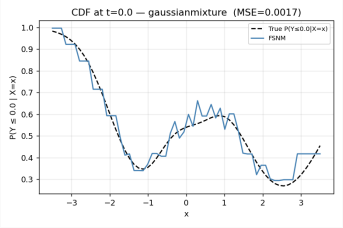


LinearStudentT: oracle $\kappa(x, y) = p(x, y)/(p(x)p(y))$ vs. learned $\hat{\kappa}_{\phi,\psi}$. FSNM captures the saddle structure (high κ at $(\pm X, \pm Y)$ corners, $\kappa \approx 1$ along the dashed independence boundary).

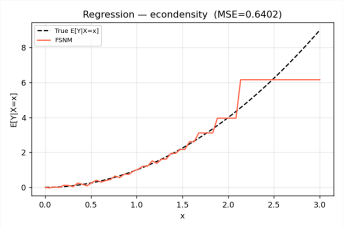
Experiments: qualitative recovery

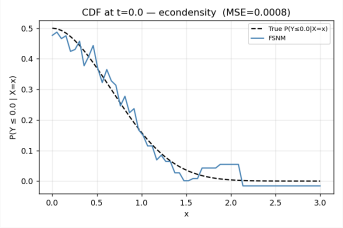
GaussianMixture (multimodal $Y \mid X$)





EconDensity ($Y = X^2 + \varepsilon$)





Top: $\mathbb{E}[Y \mid X = x]$. Bottom: $\mathbb{P}(Y \leq 0 \mid X = x)$. Estimates from $\hat{\kappa}$ (orange) vs. analytical truth (black).

Summary

- **Problem.** Learn rank-d truncation of the Radon–Nikodym $\kappa = dP_{X,Y}/d(P_X \otimes P_Y)$ — a single object underlying regression, conditional CDFs, and independence testing.
- **Method.** FSNM: block functional Newton with closed-form $d \times d$ Hessian inverses; each step = relaxed move towards the exact best response.
- **Theory.** Monotone descent, exact contraction identities, asymptotic stationarity under uniform spectral control of Σ_ϕ, Σ_ψ.
- **Application.** Plug-in T_n + permutation calibration $\Rightarrow$ finite-sample valid independence test.

Thank you! Questions?