# Functional Newton Methods for Operator Learning

Thiago R. Ramos

Joint work with Alek Fröhlich, Daniel Perazzo, and Massimiliano Pontil

Universidade Federal de São Carlos – UFSCar

July 27, 2026

# Learn One Operator, Answer Many Questions


$$(\mathcal{E}f)(x) = \mathbb{E}\big[f(Y) \mid X = x\big]$$

**What we do not know**

$P_{X,Y}$, the joint distribution

**What we observe**

$\mathcal{D}_n = \{(X_i, Y_i)\}_{i=1}^n$

**Main question: How can we learn $\mathcal{E}$ from the sample?**

# Supervised Learning

First, let us learn one conditional expectation.

# Supervised Learning in Three Steps

**1. OBSERVE**

$(X_i, Y_i)$

input–output pairs

$\rightarrow$

**2. LEARN**

$g : \mathcal{X} \to \mathcal{Y}$

a prediction rule

$\rightarrow$

**3. PREDICT**

$x \mapsto g(x)$

an output close to $y$

The pairs are independent draws from an unknown distribution:

$$\mathcal{D}_n = \{(X_i, Y_i)\}_{i=1}^n, \qquad (X_i, Y_i) \overset{\text{i.i.d.}}{\sim} P_{X,Y}.$$

**We use examples to learn a rule that predicts new outputs.**

# A Loss Function Defines "Close"

One prediction produces one error:

$$\text{observed } Y \quad \longleftrightarrow \quad \text{predicted } g(X) \quad \xrightarrow{\ell} \quad \text{loss.}$$

**Squared loss**

$$\ell(y, \widehat{y}) = (y - \widehat{y})^2$$

Penalizes large errors more strongly.

**Absolute loss**

$$\ell(y, \widehat{y}) = |y - \widehat{y}|$$

Grows linearly with the error.

**Population risk is simply the average loss:** $R(g) = \mathbb{E}[\ell(Y, g(X))]$.

# The Loss Determines What We Learn

For each $x$, choose the prediction $a$ that minimizes the conditional loss:

$$g^\star(x) \in \operatorname*{argmin}_a \mathbb{E}[\ell(Y, a) \mid X = x].$$

**Squared loss** $(Y - a)^2$

derivative equals zero at

$$g^\star(x) = \mathbb{E}[Y \mid X = x]$$

**Target: conditional mean**

**Absolute loss** $|Y - a|$

subgradient contains zero at

$$g^\star(x) \in \text{Median}(Y \mid X = x)$$

**Target: conditional median**

**Changing the loss changes the conditional quantity we learn.**

# From Now On: Squared Loss

**OBJECTIVE**

$$\min_g \ \mathbb{E}[(Y - g(X))^2]$$

Find the function with the smallest average squared error.

$\Longrightarrow$

**POPULATION SOLUTION**

$$g^\star(x) = \mathbb{E}[Y \mid X = x]$$

The best predictor is the conditional mean.

**This conditional mean will be our bridge to operator learning.**

# Different Models, Same Target

All these models try to approximate $g^\star(x) = \mathbb{E}[Y \mid X = x]$.

**Linear regression**

Affine relationships

**Kernel methods**

Similarities between observations

**Decision trees**

Partitions of the input space

**Neural networks**

Flexible nonlinear representations

**Random forests**

Averages of randomized trees

**Gradient boosting**

Small corrections learned sequentially

# Gradient Boosting

Next, we move from parameters to functions.

# Boosting Builds the Model Step by Step

**START**

$g_0(x)$

initial prediction

$+$

**CORRECTION** $m$

$\eta\, h_m(x)$

a small tree

$+ \cdots$

**FINAL MODEL**

$g_M(x)$

sum of corrections

$$\boxed{g_M(x) = g_0(x) + \eta \sum_{m=1}^{M} h_m(x)} \qquad \approx \qquad \mathbb{E}[Y \mid X = x].$$

**Simple learners**

Each $h_m$ can be a shallow decision tree.

**Learning rate**

$\eta$ controls how much each correction changes the model.

# Gradient Descent: Follow the Downhill Direction

**1. Current point**

$\theta_t$

Evaluate the gradient.

$\rightarrow$

**2. Direction**

$-\nabla f(\theta_t)$

Move downhill.

$\rightarrow$

**3. New point**

$\theta_{t+1}$

Lower the objective.

**Goal**

$$\min_{\theta \in \mathbb{R}^p} f(\theta)$$

**Update**

$$\boxed{\theta_{t+1} = \theta_t - \eta \nabla f(\theta_t)}$$

**The gradient points uphill, so its negative points downhill.**

# Each Step Adds a Correction

Name the descent direction

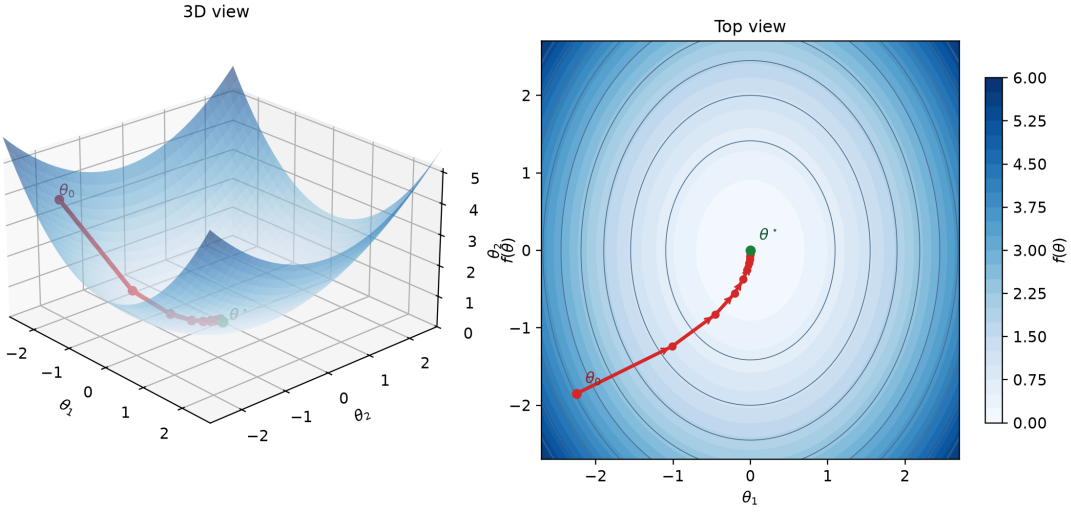
$$h_t = -\nabla f(\theta_t).$$

**CURRENT** $\theta_t$ $+$ **CORRECTION** $\eta h_t$ $=$ **NEXT** $\theta_{t+1}$

With a suitable step size, $f(\theta_{t+1}) \leq f(\theta_t)$.

**This additive form is the connection to gradient boosting.**

# Gradient Descent on a Quadratic Landscape



# Gradient Descent and Boosting Share the Same Pattern

**GRADIENT DESCENT**

$$\theta_{t+1} = \theta_t + \eta h_t$$

$$\Downarrow \qquad \theta_T = \theta_0 + \eta \sum_{t=0}^{T-1} h_t$$

corrections are vectors $h_t$

**GRADIENT BOOSTING**

$$g_m = g_{m-1} + \eta h_m$$

$$\Downarrow \qquad g_M = g_0 + \eta \sum_{m=1}^{M} h_m$$

corrections are functions $h_m$

initial solution $+$ sequence of corrections

**The algebra is the same; only the type of object changes.**

# Gradient Boosting Optimizes a Whole Function

**Ordinary gradient descent**

**Variable:** vector $\theta \in \mathbb{R}^p$

**Objective:** number $f(\theta)$

$$\theta_{t+1} = \theta_t - \eta \nabla f(\theta_t)$$

**Gradient boosting**

**Variable:** function $g \in \mathcal{H}$

**Objective:** risk $\mathcal{R}(g)$

$$g_{t+1} = g_t - \eta \nabla \mathcal{R}(g_t)$$

**Gradient boosting is gradient descent in function space.**

# Three Ingredients Move Optimization to Function Space

**1. SEARCH SPACE**

Where does the variable live?

$$g \in \mathcal{H}$$

**2. GEOMETRY**

How do we measure angles?

$$\langle g, h \rangle_{\mathcal{H}}$$

**3. GRADIENT**

Which direction is uphill?

$$\nabla \mathcal{R}(g)$$

| **Vectors** | | **Functions** |
|---|---|---|
| $\theta \in \mathbb{R}^p$ | $\longrightarrow$ | $g \in \mathcal{H}$ |
| $u^\top v$ | $\longrightarrow$ | $\langle g, h \rangle_{\mathcal{H}}$ |
| $\nabla f(\theta)$ | $\longrightarrow$ | $\nabla \mathcal{R}(g)$ |

# Squared-Loss Regression Lives in $L^2(P_X)$

**FUNCTION SPACE**

$$\mathcal{H} = L^2(P_X)$$

Functions with finite average squared magnitude:

$$\mathbb{E}[g(X)^2] < \infty.$$

**GEOMETRY**

$$\langle g, h \rangle_{\mathcal{H}} = \mathbb{E}[g(X)h(X)]$$

and therefore

$$\|g\|_{\mathcal{H}}^2 = \mathbb{E}[g(X)^2].$$

**Our optimization variable is now a function:** $\min_{g \in \mathcal{H}} \mathcal{R}(g)$.

$$\mathcal{R}(g) = \mathbb{E}[(Y - g(X))^2].$$

# A Functional Derivative Asks: "What If We Perturb $g$?"

$$g \quad \longrightarrow \quad g + \varepsilon h \qquad \text{perturb } g \text{ in direction } h$$

**Directional change**

$$D\mathcal{F}(g)[h] = \lim_{\varepsilon \to 0} \frac{\mathcal{F}(g + \varepsilon h) - \mathcal{F}(g)}{\varepsilon}$$

This is a number for each direction $h$.

**Functional gradient**

$$D\mathcal{F}(g)[h] = \langle h, \nabla \mathcal{F}(g) \rangle_{\mathcal{H}}$$

Riesz represents all directional changes by one gradient function.

**It is the usual derivative idea, but the variable is a function.**

# The Squared-Loss Gradient Appears in Three Lines

**1. RISK**

$$\mathcal{R}(g) = \mathbb{E}[(Y - g(X))^2]$$

Start with squared loss.

**2. DIFFERENTIATE**

$$D\mathcal{R}(g)[h] = 2\mathbb{E}[(g(X) - Y)h(X)]$$

Perturb in direction $h$.

**3. CONDITION ON $X$**

$$D\mathcal{R}(g)[h] = \langle 2\mathbb{E}[g(X) - Y \mid X], h \rangle$$

Use the tower property.

**The function inside the inner product is the functional gradient.**

# The Negative Gradient Is a Conditional Residual

**FUNCTIONAL GRADIENT**

$$\nabla \mathcal{R}(g)(x) = 2\,\mathbb{E}[g(X) - Y \mid X = x]$$

**DESCENT DIRECTION**

$$-\nabla \mathcal{R}(g)(x) \propto \mathbb{E}[Y - g(X) \mid X = x]$$

expected residual at input $x$

$$g_{t+1}(x) = g_t(x) + 2\eta\,\mathbb{E}[Y - g_t(X) \mid X = x].$$

# One Boosting Iteration Is an Ordinary Regression

**1. RESIDUALS**

$$r_i^{(t)} = Y_i - g_t(X_i)$$

compute what the model misses

$\rightarrow$

**2. FIT A TREE**

$$(X_i, r_i^{(t)})$$

estimate the conditional residual

$\rightarrow$

**3. UPDATE**

$$g_{t+1} = g_t + \eta h_t$$

add the fitted correction

$h_t(x)$ approximates $\quad \mathbb{E}[Y - g_t(X) \mid X = x]$.

**The abstract functional step becomes a familiar supervised-learning problem.**

# The Translation from Vectors to Functions

| | **Gradient descent** | **Gradient boosting** |
|---|---|---|
| Variable | $\theta \in \mathbb{R}^p$ | $g \in L^2(P_X)$ |
| Objective | $f(\theta)$ | $\mathcal{R}(g) = \mathbb{E}[(Y - g(X))^2]$ |
| Direction | $-\nabla f(\theta_t)$ | conditional residual |
| Approximation | exact vector | fitted tree $h_t$ |
| Update | $\theta_{t+1} = \theta_t + \eta d_t$ | $g_{t+1} = g_t + \eta h_t$ |

**Same principle**

Move along a direction that decreases the objective.

**Essential difference**

In boosting, the correction itself is a learned function.

# Operator Learning

Now, we learn all conditional expectations at once.

# From One Conditional Expectation to All of Them

**ORDINARY REGRESSION**

**Fix one input function**

$$f(y) = y$$

$$\Downarrow$$

$$\mathbb{E}[Y \mid X = x]$$

learns one conditional query

**OPERATOR LEARNING**

**Allow every admissible $f$**

$$f \in L^2(P_Y)$$

$$\Downarrow$$

$$\mathbb{E}[f(Y) \mid X = x]$$

learns a reusable map

$\mathcal{E} : L^2(P_Y) \to L^2(P_X)$ **maps $f$ to the function** $x \mapsto \mathbb{E}[f(Y) \mid X = x]$.

# A Compact Operator Has a Matrix-Like SVD

**MATRIX**

$$A = \sum_j \sigma_j u_j v_j^\top$$

orthonormal singular vectors $u_j, v_j$
rank-one matrices $u_j v_j^\top$

$\leftrightarrow$

**OPERATOR**

$$\mathcal{E} = \sum_j \sigma_j \phi_j \otimes \psi_j$$

orthonormal singular functions $\phi_j, \psi_j$
rank-one operators $\phi_j \otimes \psi_j$

$\{\phi_j\} \subset L^2(P_X)$ and $\{\psi_j\} \subset L^2(P_Y)$ are orthonormal.

**As in matrix SVD, orthonormality separates the spectral directions.**

# The Density Ratio Determines the Operator

**DEPENDENCE KERNEL**

$$\kappa(x,y) = \frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)}$$

compares the joint law with independence

$\Longrightarrow$

**SUBSTITUTE THE CONDITIONAL DENSITY**

$$\begin{aligned}(\mathcal{E}f)(x) &= \int f(y)p_{Y|X}(y \mid x)\,dy \\ &= \int f(y)\frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)}p_Y(y)\,dy \\ &= \int f(y)\kappa(x,y)\,dP_Y(y).\end{aligned}$$

$p_{Y|X}(y \mid x) = \kappa(x,y)p_Y(y) \quad \Longrightarrow \quad$ learning $\kappa$ is enough to learn $\mathcal{E}$.

# Comparing the Two Forms Identifies $\kappa$

**USING THE SVD**

$$(\mathcal{E}f)(x) = \sum_{j\geq 0} \sigma_j \langle f, \psi_j \rangle \phi_j(x) = \int f(y) \left[ \sum_{j\geq 0} \sigma_j \phi_j(x) \psi_j(y) \right] dP_Y(y).$$

**USING THE KERNEL**

$$(\mathcal{E}f)(x) = \int f(y) \kappa(x, y)\, dP_Y(y).$$

Since both expressions are equal for every $f$, the terms multiplying $f(y)$ must be equal:

$$\boxed{\kappa(x, y) = \sum_{j\geq 0} \sigma_j \phi_j(x) \psi_j(y).}$$

# Choose $f$ to Choose the Question

**CONDITIONAL MEAN**

$$f(y) = y$$

$$\Downarrow$$

$$(\mathcal{E}f)(x) = \mathbb{E}[Y \mid X = x]$$

ordinary regression

**CONDITIONAL CDF**

$$f_t(y) = 1\{y \le t\}$$

$$\Downarrow$$

$$(\mathcal{E}f_t)(x) = \mathbb{P}(Y \le t \mid X = x)$$

uncertainty quantification

**The operator stays fixed; only the input function changes.**

# The Operator Also Reveals Uncertainty and Dependence

**CONDITIONAL VARIANCE**

Use $f(y) = y$ and $f(y) = y^2$:

$$\mathrm{Var}(Y \mid X = x) = \mathbb{E}[Y^2 \mid X = x] - \mathbb{E}[Y \mid X = x]^2.$$

One operator supplies both moments.

**INDEPENDENCE**

$$X \perp Y \iff \kappa(x, y) = 1$$

no nonconstant singular components

**The learned spectrum summarizes dependence beyond a single regression target.**

# How to Learn $\kappa$?

We now turn the operator into a learnable low-rank model.

# A Low-Rank Model Makes $\kappa$ Learnable

**KNOWN BASELINE**

$$1$$

independence component
$\sigma_0 = 1$

$+$

**LEARNED LOW-RANK CORRECTION**

$$\sum_{j=1}^{k} \sigma_j \phi_j^\star(x) \psi_j^\star(y)$$

keep only $k$ nonconstant components

Absorb $\sqrt{\sigma_j}$ and stack the functions into two feature vectors:

$$\phi(x) = \begin{pmatrix} \sqrt{\sigma_1}\phi_1^\star(x) \\ \vdots \\ \sqrt{\sigma_k}\phi_k^\star(x) \end{pmatrix}, \qquad \psi(y) = \begin{pmatrix} \sqrt{\sigma_1}\psi_1^\star(y) \\ \vdots \\ \sqrt{\sigma_k}\psi_k^\star(y) \end{pmatrix}.$$

**The model to learn is simply $\kappa_{\phi,\psi}(x, y) = 1 + \phi(x)^\top \psi(y)$.**

# Learn the Features by Matching the True Kernel

**UNKNOWN TARGET**

$$\kappa(x, y)$$

true density ratio

compare

**LOW-RANK MODEL**

$$\kappa_{\phi,\psi}(x, y) = 1 + \phi(x)^\top \psi(y)$$

feature maps to learn

**Measure their squared $L^2$ distance:**

$$\mathcal{L}(\phi, \psi) = \|\kappa - \kappa_{\phi,\psi}\|^2_{L^2(P_X \otimes P_Y)}$$

**Choose $\phi$ and $\psi$ so the low-rank kernel is as close as possible to $\kappa$.**

# Expanding the Loss Reveals Three Terms

$$\mathcal{L}(\phi, \psi) = \underbrace{\mathbb{E}[\kappa^2]}_{\text{constant}} - 2\underbrace{\mathbb{E}[\kappa\, \kappa_{\phi,\psi}]}_{\text{cross term}} + \underbrace{\mathbb{E}[\kappa^2_{\phi,\psi}]}_{\text{model term}} .$$

**CONSTANT**

Does not depend on $\phi, \psi$.

Drop it.

**CROSS TERM**

Contains unknown $\kappa$.

**Rewrite it.**

**MODEL TERM**

Depends only on our model.

**Estimate it.**

**Only the cross term looks problematic—and the density-ratio identity fixes it.**

# The Density-Ratio Identity Removes the Unknown $\kappa$

## WHY THE IDENTITY HOLDS

$$\begin{aligned}\mathbb{E}_{P_X \otimes P_Y}[\kappa(X,Y)q(X,Y)] &= \int q(x,y)\,\kappa(x,y)p_X(x)p_Y(y)\,dx\,dy \\ &= \int q(x,y)\,p_{X,Y}(x,y)\,dx\,dy \\ &= \mathbb{E}_{P_{X,Y}}[q(X,Y)].\end{aligned}$$

Choose $q = \kappa_{\phi,\psi}$. Then the objective becomes

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

**Product expectation**

Estimate with independently paired $X$ and $Y$.

**Joint expectation**

Estimate with the observed pairs $(X_i, Y_i)$.

# Functional Gradient Descent: Update $\phi$ with $\psi$ Fixed

**THE SAME UPDATE USED IN BOOSTING**

$$\phi^{(t+1)} = \phi^{(t)} - \eta_\phi \nabla_\phi L(\phi^{(t)}, \psi)$$

**FUNCTIONAL GRADIENT**

$$\nabla_\phi L(x) = 2\Sigma_\psi \phi(x) - 2m_\psi(x)$$

a function of $x$

**TERMS IN THE GRADIENT**

$$\Sigma_\psi = \mathbb{E}[\psi(Y)\psi(Y)^\top]$$
$$m_\psi(x) = \mathbb{E}[\psi(Y) - \mathbb{E}\psi(Y) \mid X = x]$$

**The update for $\psi$ is identical after exchanging $X$ and $Y$.**

# From Functional Gradient Boosting to a Newton Step

## FIRST-ORDER FUNCTIONAL BOOSTING

Use only the functional gradient:

$$\phi^{(t+1)} = \phi^{(t)} - \eta_\phi \nabla_\phi L.$$

The gradient gives the direction.

## SECOND-ORDER / NEWTON BOOSTING

Also use the functional Hessian:

$$\phi^{(t+1)} = \phi^{(t)} - \eta_\phi [\nabla^2_{\phi\phi} L]^{-1} \nabla_\phi L$$

The Hessian rescales the direction.

## WHY THIS IS SIMPLE HERE

With $\psi$ fixed, the loss is quadratic in $\phi$ and $\nabla^2_{\phi\phi} L = 2\Sigma_\psi$. The same holds after exchanging $\phi$ and $\psi$.

# Alternate the Two Functional Newton Updates

**1. FIX $\phi^{(t)}$, UPDATE $\psi$**

Fix $\phi^{(t)}$.

Take one relaxed Newton step:

$$\psi^{(t+1)} = (1 - \eta_\psi)\psi^{(t)} + \eta_\psi \Sigma_{\phi,t}^{-1} m_{\phi,t}.$$

**2. FIX $\psi^{(t+1)}$, UPDATE $\phi$**

Fix the new $\psi^{(t+1)}$.

Take the symmetric Newton step:

$$\phi^{(t+1)} = (1 - \eta_\phi)\phi^{(t)} + \eta_\phi \Sigma_{\psi,t+1}^{-1} m_{\psi,t+1}.$$

**We update one block at a time; the next slide shows how the unknown terms are estimated.**

# Population Quantities Become Familiar Data Operations

**EXPECTATIONS → AVERAGES**

$$\mathbb{E}[\psi(Y)] \quad \longrightarrow \quad \overline{\psi} = \frac{1}{n}\sum_i \psi(Y_i)$$

$$\Sigma_\psi \quad \longrightarrow \quad \widehat{\Sigma}_\psi = \frac{1}{n}\sum_i \psi(Y_i)\psi(Y_i)^\top$$

**CONDITIONAL EXPECTATIONS → REGRESSIONS**

Recall the population target:

$$m_\psi(x) = \mathbb{E}[\psi(Y) - \mathbb{E}\psi(Y) \mid X = x].$$

Estimate it with the derived training pairs

$$\big(X_i,\ \psi(Y_i) - \overline{\psi}\big).$$

Reverse $X$ and $Y$ to estimate $m_\phi(y)$.

Flexible regressors such as decision trees can be used in both steps.

**Every abstract population object has a direct empirical counterpart.**

# Spectral Balancing Makes the Factors Interpretable

**BEFORE BALANCING**

$$\widehat{\kappa} = 1 + \phi^\top \psi$$

Components are not necessarily orthogonal or normalized.

change coordinates

**AFTER BALANCING**

$$\widehat{\kappa} = 1 + \sum_{j=1}^{k} \widehat{\sigma}_j \widehat{\phi}_j^\star \widehat{\psi}_j^\star$$

Orthonormal singular functions and explicit singular values.

**WHAT STAYS UNCHANGED**

$$\widetilde{\phi} = A\phi, \quad \widetilde{\psi} = A^{-\top}\psi \quad \Longrightarrow \quad \widetilde{\phi}^\top \widetilde{\psi} = \phi^\top \psi.$$

**Balancing changes the coordinates, not the estimated kernel.**

# Applications

Finally, let us see what the learned operator can do.

# Application 1: Can Our Method Recover a Known Spectrum?

### 1. CHOOSE A BASIS

$X, Y \sim \mathrm{Unif}[-1, 1]$

$$e_1(t) = \sqrt{2}\cos(\pi t)$$
$$e_2(t) = \sqrt{2}\sin(\pi t)$$
$$e_3(t) = \sqrt{2}\cos(2\pi t)$$

The functions are orthonormal.

### 2. BUILD $\kappa$

$$\kappa(x, y) = 1 + \sum_{j=1}^{3} \sigma_j e_j(x) e_j(y)$$

with

$$\sigma = (0.30, 0.18, 0.10).$$

The exact spectrum is known.

### 3. FIT OUR METHOD

Sample from the resulting joint distribution and fit a rank-3 model.

Then compare the learned spectrum with the three known values.

**Because the basis is orthonormal, the formula for $\kappa$ is already its exact SVD.**

# Application 1: Our Method Recovers the Known Spectrum

20,000 training points · 5,000 validation points · rank $k = 3$

| | $\sigma_1$ | $\sigma_2$ | $\sigma_3$ |
|---|---|---|---|
| Exact | 0.300 | 0.180 | 0.100 |
| Our method $\widehat{\sigma}_j$ | 0.299 | 0.183 | 0.104 |

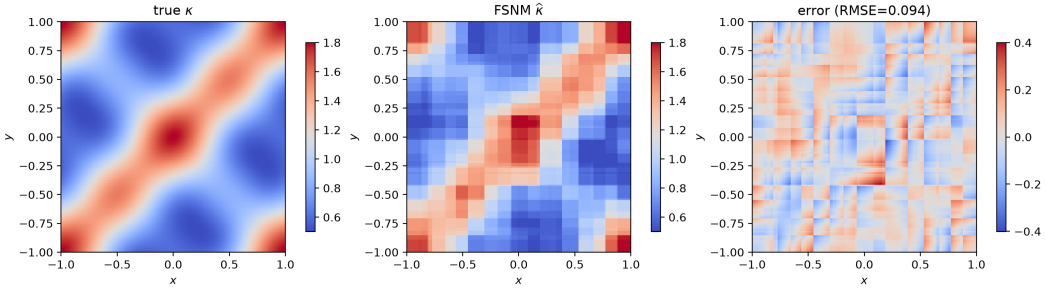
**SPECTRUM**

All three estimated singular values are close to the truth.

**ORTHONORMALITY**

$\|\widehat{\Sigma}_{\widehat{\phi}} - I\| \approx 10^{-15}$ after balancing.

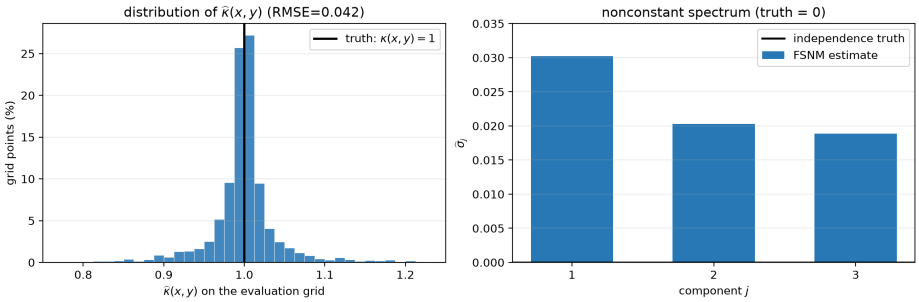
# Application 1: The Main Structure Is Recovered




**The blocky pattern comes from the decision trees used in the functional updates.**

# Application 2: Independence Leaves No Nonconstant Spectrum

If $X \perp Y$, then $p_{X,Y} = p_X p_Y$ and therefore $\kappa(x, y) = 1$. Hence $\kappa - 1 = 0$: every nonconstant singular value is zero. We simulate 20,000 independent pairs with $X, Y \sim \mathrm{Unif}[-1, 1]$ and still fit our method with rank $k = 3$.




Our method recovers $(\widehat{\sigma}_1, \widehat{\sigma}_2, \widehat{\sigma}_3) = (0.030, 0.020, 0.019)$; the small residual components are finite-sample estimation noise.

**Under independence, our method finds only small finite-sample components.**

# Application 3: Fit Once, Answer Many Queries

**FIXED**

learned operator

$\widehat{\sigma}_j, \widehat{\phi}_j, \widehat{\psi}_j$

$+$

**NEW QUERY**

input function $f$

compute $k$ projection coefficients

$=$

**ANSWER**

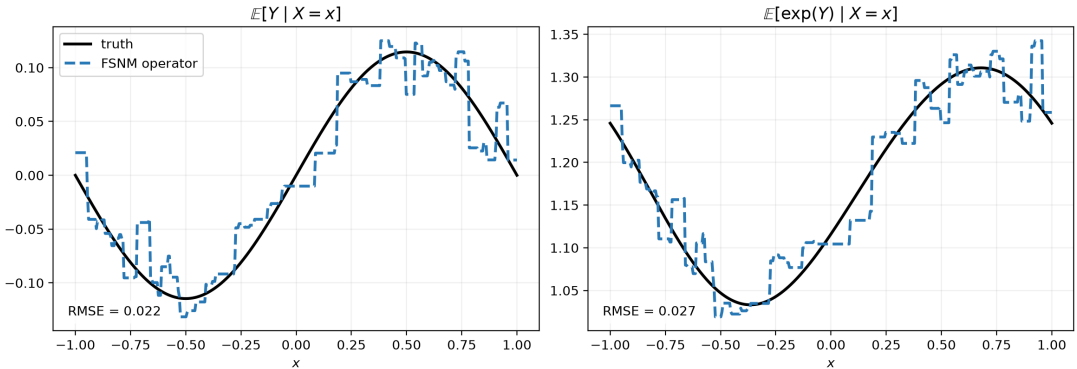
$(\widehat{\mathcal{E}} f)(x)$

no tree is refitted

$$(\widehat{\mathcal{E}} f)(x) = \overline{f} + \sum_{j=1}^{k} \widehat{\sigma}_j \widehat{\phi}_j(x) \underbrace{\left[ \frac{1}{m} \sum_{\ell=1}^{m} f(Y_\ell) \widehat{\psi}_j(Y_\ell) \right]}_{\text{only this coefficient changes}}.$$

**We will test $f(y) = y$ and $f(y) = e^y$ using the same fitted operator.**

# Application 3: The Same Fit Answers Both Queries



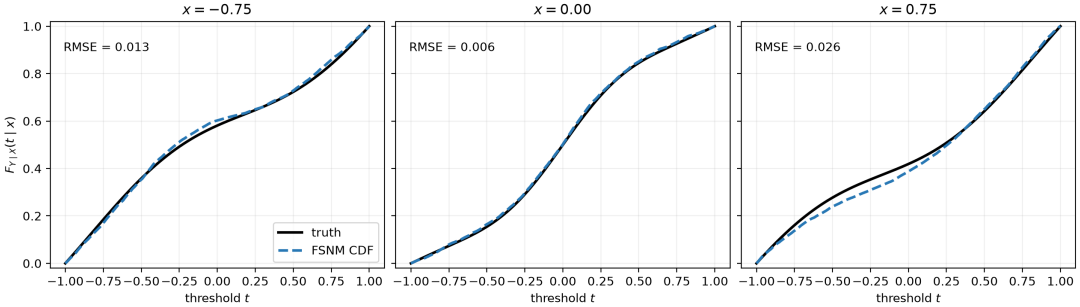

The same fitted operator answers both queries, with grid RMSEs 0.022 and 0.027, respectively.

# Application 3: The Same Fit Recovers a Conditional CDF

For each threshold $t$, take $f_t(y) = 1\{y \le t\}$. Then

$$F_{Y|X}(t \mid x) = \mathbb{P}(Y \le t \mid X = x) = (\mathcal{E} f_t)(x).$$

Evaluating the same learned operator over a grid of $t$ therefore estimates the entire conditional distribution, without refitting the model.



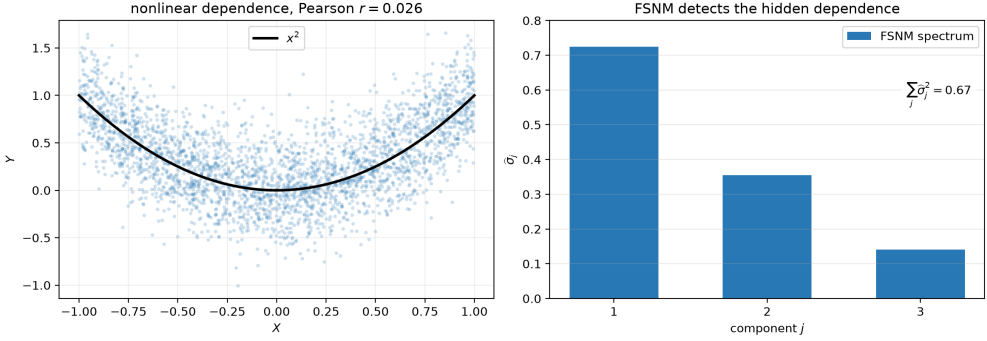

For $x \in \{-0.75, 0, 0.75\}$, the CDF RMSE ranges from 0.006 to 0.026; the estimated curves are already monotone in this run.

# Application 4: Zero Correlation Does Not Mean Independence

Let

$$X \sim \mathrm{Unif}[-1,1], \qquad Y = X^2 + \varepsilon, \qquad \varepsilon \sim N(0, 0.3^2).$$

Symmetry gives $\mathrm{Cov}(X,Y) = \mathbb{E}[X^3] = 0$, although $\mathbb{E}[Y \mid X] = X^2$: zero linear correlation does not imply independence.




Pearson gives $r = 0.026$, while our method finds $\widehat{\sigma} = (0.725, 0.354, 0.141)$ and captured spectral energy $\sum_{j=1}^{3} \widehat{\sigma}_j^2 = 0.671$.

# Summary

**1. OPERATOR VIEW**

Learn the map

$$f \longmapsto \mathbb{E}[f(Y) \mid X = \cdot]$$

instead of one regression target.

**2. LOW-RANK MODEL**

Represent the density ratio as

$$\kappa(x, y) \approx 1 + \phi(x)^\top \psi(y).$$

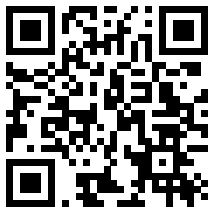
**3. FUNCTIONAL NEWTON**

Learn $\phi$ and $\psi$ with alternating Newton steps built from ordinary regressions.

**Fit the operator once, then reuse it for many conditional queries.**

# Thank you!

Questions?



Scan to read the paper

`openreview.net/pdf?id=8CXoyFIV85`