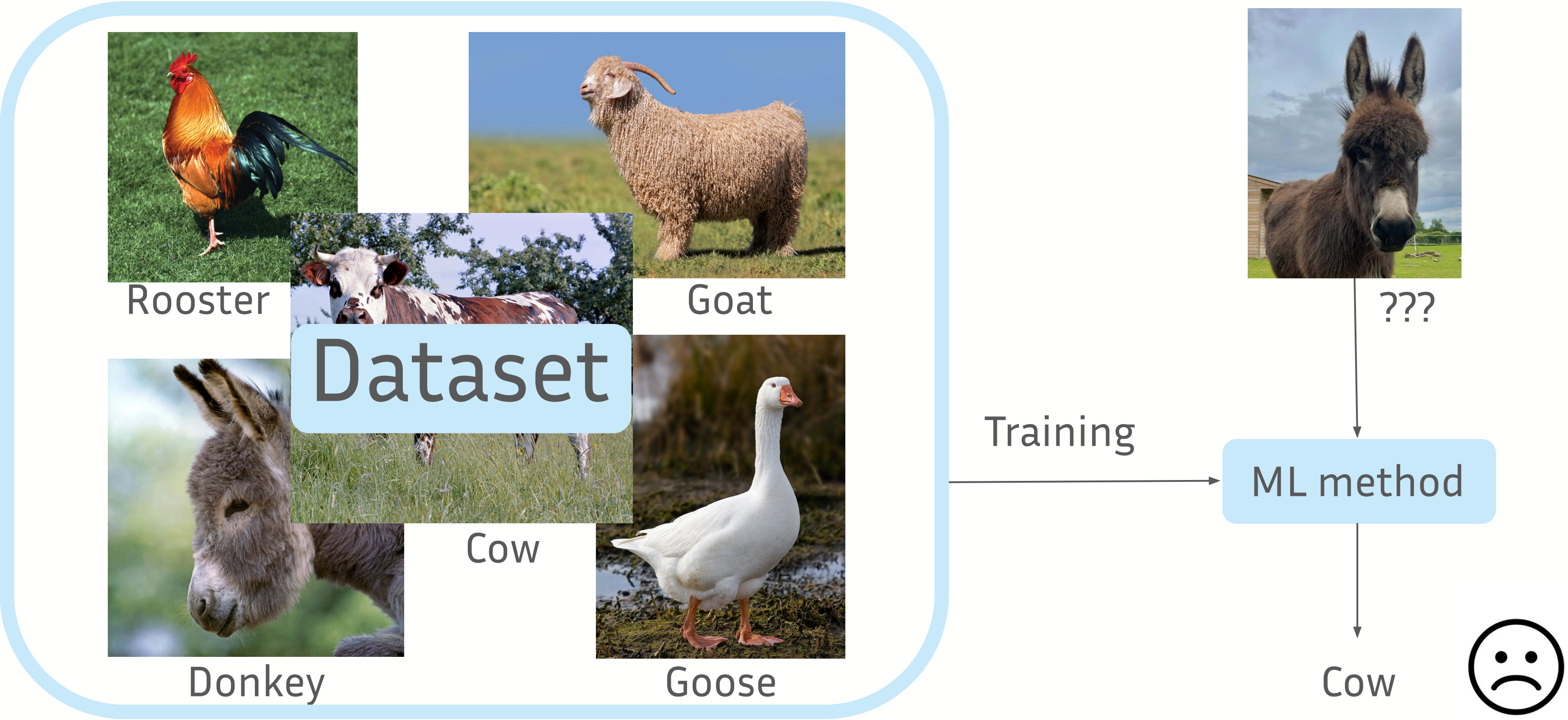
# Betting on Sparsity

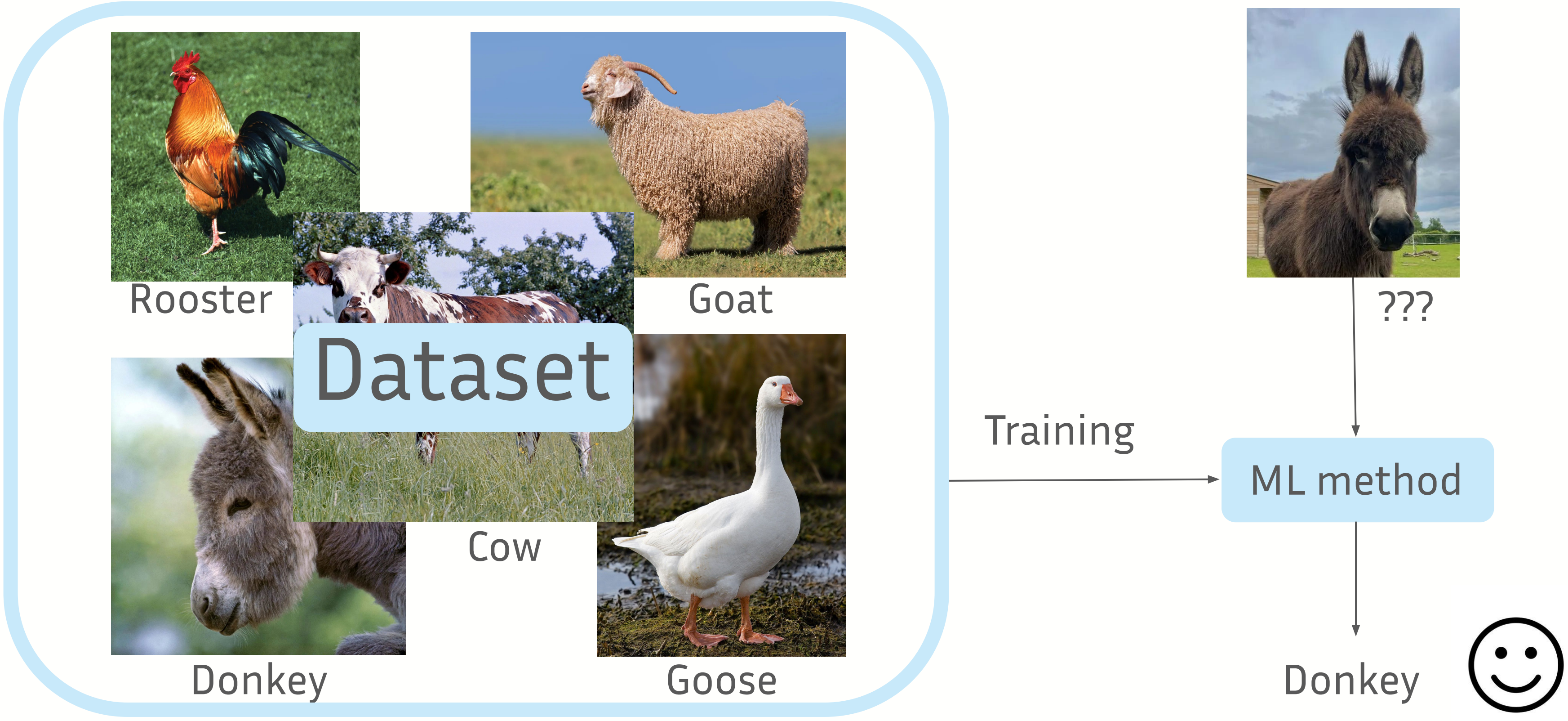
## Leveraging Hidden Linear Features through Regularisation for Supervised Learning

**Bertille Follain**
Inria & ENS
21/11/2024

# **Betting on Sparsity:** Leveraging Hidden Linear Features through Regularisation for Supervised Learning



# **Betting on Sparsity:** Leveraging Hidden Linear Features through Regularisation for Supervised Learning



Dataset (i.i.d.): $(x_i, y_i)_{i\in[n]}$

$X \in \mathbb{R}^d, Y \in \mathbb{R}$

Risk:

$$\mathcal{R}(f) := \mathbb{E}_{(X,Y)}(\ell(Y, f(X)))$$

True regression function:

$$f^* := \operatorname{argmin}_f \mathcal{R}(f)$$

Estimate using some method: $\hat{f}$

## Learning Theory

For an estimator $\hat{f}$, under some assumptions on the dataset and $f^*$, with probability larger than $1-\delta$:

$$\mathcal{R}(\hat{f}) \leq \mathcal{R}(f^*) + \epsilon(n, d, \delta, \dots)$$

Curse of dimensionality:

$$\epsilon(n, d, \delta, \dots) \approx n^{-1/d}$$
$$\epsilon(n, d, \delta, \dots) \approx \exp(d)$$

☹

(Bellman, 1966)

# Betting on Sparsity: Leveraging Hidden Linear Features through Regularisation for Supervised Learning

## Sparsity Assumptions

- Few relevant coordinates of $x$

Ex: $f^*(x) = g^*(x_1, x_3)$

- Few $(s)$ relevant projections of $x$

Ex: $f^*(x) = g^*(w_1^\top x, w_2^\top x)$

**Multi-Index Model**

$(s \ll d)$

## Learning Theory

For an estimator $\hat{f}$, under some assumptions on the dataset and $f^*$, with probability larger than $1 - \delta$:

$$\mathcal{R}(\hat{f}) \leq \mathcal{R}(f^*) + \epsilon(n, d, \delta, \ldots)$$

Curse c… mensionalit…

$\epsilon(n, d, \ldots$ … $^{1/d}$

$\epsilon(n$ … $\ldots)$ … $(d)$

(Li, 1991)

# Betting on Sparsity: Leveraging Hidden Linear Features through Regularisation for Supervised Learning

**Regularised Empirical Risk Minimisation**

$$f^* := \operatorname*{argmin}_f \mathcal{R}(f) \quad \Rightarrow \quad \hat{f} := \operatorname*{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(x_i)) + \lambda \Omega(f)$$

Parametric methods:
- Linear regression
- Lasso
- Neural networks

Nonparametric methods:
- Kernel methods
- Neural Networks?

Well-specified model: $f^* \in \mathcal{F}$

Variable selection by changing penalty:

$$(\mathcal{F}, \Omega_{\text{feature}}) \Rightarrow (\mathcal{F}, \Omega_{\text{variable}})$$

(Vapnik, 1991)

# Existing Methods for Multi-Index Models & Goals

**Multi-index model**

$f^*(\cdot) = g^*(P^\top \cdot)$

$P \in \mathbb{R}^{d \times s}, s \ll d$

Moment-based vs optimisation-based methods

Methods to compare against: MAVE & Neural networks

$$\hat{f} := \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(x_i)) + \lambda \Omega(f)$$

**Goals**

- Use regularised empirical risk minimisation (RERM) for flexibility
- Make few assumptions on data distribution
- Make limited assumptions on $f^*$
- Obtain theoretical bounds with limited dependence on data dimension $d$



(Xia et al., 2002)(Bach, 2024)

# Reproducing Kernel Hilbert Spaces (RKHS)

Function space: $(\mathcal{H}, \langle \cdot, \cdot \rangle_{\mathcal{H}})$

$f \in \mathcal{H}, f : \mathcal{X} \to \mathbb{R}$

Associated reproducing kernel:

$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

Reproducing property:

$$f(x) = \langle f, k_x \rangle_{\mathcal{H}}$$

$k_x := k(x, \cdot)$

Ex: $k(x, x') = \exp(-\|x - x'\|^2/2)$

$\mathcal{H}$: set of functions with square integrable derivatives of all order

Kernel ridge regression (KRR) (square loss):

$$\hat{f} := \underset{f \in \mathcal{H}}{\operatorname{argmin}} \widehat{\mathcal{R}}(f) + \lambda \|f\|_{\mathcal{H}}^2$$

Representer theorem: $\hat{f} = \sum_{i=1}^{n} \alpha_i k_{x_i}$

(Aronszajn, 1950)

# Trace Norm Penalty on Sample Matrix of Gradients

KTNGrad

Chapter 2: unpublished work, extension of L. Rosasco et al. Nonparametric Sparsity and Regularisation. Journal of Machine Learning Research 14(52):1665–1714. 2013.

# Main Idea

Trace Norm Penalty on Sample Matrix of Gradients
KTNGrad

**Using the gradients**

Multi-index model: $f^*(\cdot) = g^*(P^\top \cdot)$
$P \in \mathbb{R}^{d \times s}, s \ll d$

$$\nabla f^*(\cdot) = P \nabla g^*(P^\top \cdot)$$

Sample Matrix of Gradients

$$\nabla_n f := (\nabla f(x_1)^T, \ldots, \nabla f(x_n)^T)^T / \sqrt{n} \in \mathbb{R}^{n \times d}$$

$$(\nabla_n f^*)^\top \nabla_n f^* = P M_{g^*} P^\top$$

**Trace norm penalty**: convex relaxation of rank

$$\|A\|_* = \mathrm{trace}(\sqrt{A^\top A}) = \textstyle\sum_i \sigma_i(A)$$

**Choosing a RKHS $\mathcal{H}$ for $\mathcal{F}$**

Twice differentiable kernel (Gaussian)

↓

Easy computation of gradients

$$\frac{\partial f(x)}{\partial x_a} = \langle f, (\partial_a k)_x \rangle_{\mathcal{H}}$$

$$\hat{f} = \operatorname*{argmin}_{f \in \mathcal{H}} \widehat{\mathcal{R}}(f) + \lambda(2\|\nabla_n f\|_* + \nu \|f\|_{\mathcal{H}}^2)$$

(Dalalyan et al., 2007)

# Algorithm

Trace Norm Penalty on Sample Matrix of Gradients
KTNGrad

## Adapted representer theorem

$$\hat{f} = \sum_{i=1}^{n} \sum_{a=1}^{d} \alpha_{i,a} (\partial_a k)_{x_i} + \alpha_i k_{x_i}$$

## Reweighted formulation

$$\|\nabla_n f\|_* = \frac{1}{2} \inf_{\Lambda \succeq 0} \operatorname{trace}(\nabla_n f^\top \Lambda^{-1} \nabla_n f + \Lambda)$$

Alternating minimisation in closed-form

## Cost of one iteration

$$O(n^3 d^4)$$

Solution: Nyström?

## Convergence of optimisation

Convex, quick, proven

## Obtaining the features

Use $\nabla_n \hat{f}$ to estimate

- the features (leading singular vectors)
- the dimension (using the rank or a threshold)

(Zhou, 2008)(Bach et al., 2011)(Drineas and Mahoney, 2005)

# Informal Theorem



**Convergence of the expected risk (square loss)**

In the well-specified setting ( $f^* \in \mathcal{H}$ ), with bounded responses, there exists a constant $C_\nu$ such that for any $\delta \in (0, 1]$, with probability larger than $1 - \delta$

$$\mathcal{R}(\hat{f}) \leq \mathcal{R}(f^*) + C_\nu \left( \frac{1}{\lambda\sqrt{n}} + \sqrt{\lambda} \frac{d^{5/4}}{n^{1/4}} \right) \log \frac{d}{\delta}$$
$$+\lambda \left( 2\|\nabla f^*\|_* + \nu \|f^*\|_{\mathcal{H}}^2 \right)$$

**Recovery of the hidden linear features**

When, $n \to \infty$ for a well chosen sequence $(\lambda_n)_{n\in\mathbb{N}}$, with $\Pi_Q$ the projection matrix associated to features $Q$:

$$\|\Pi_P (I_d - \Pi_{\hat{P}})\|_F^2 \xrightarrow{P} 0$$

(Rosasco et al., 2013)

# Numerical Experiments

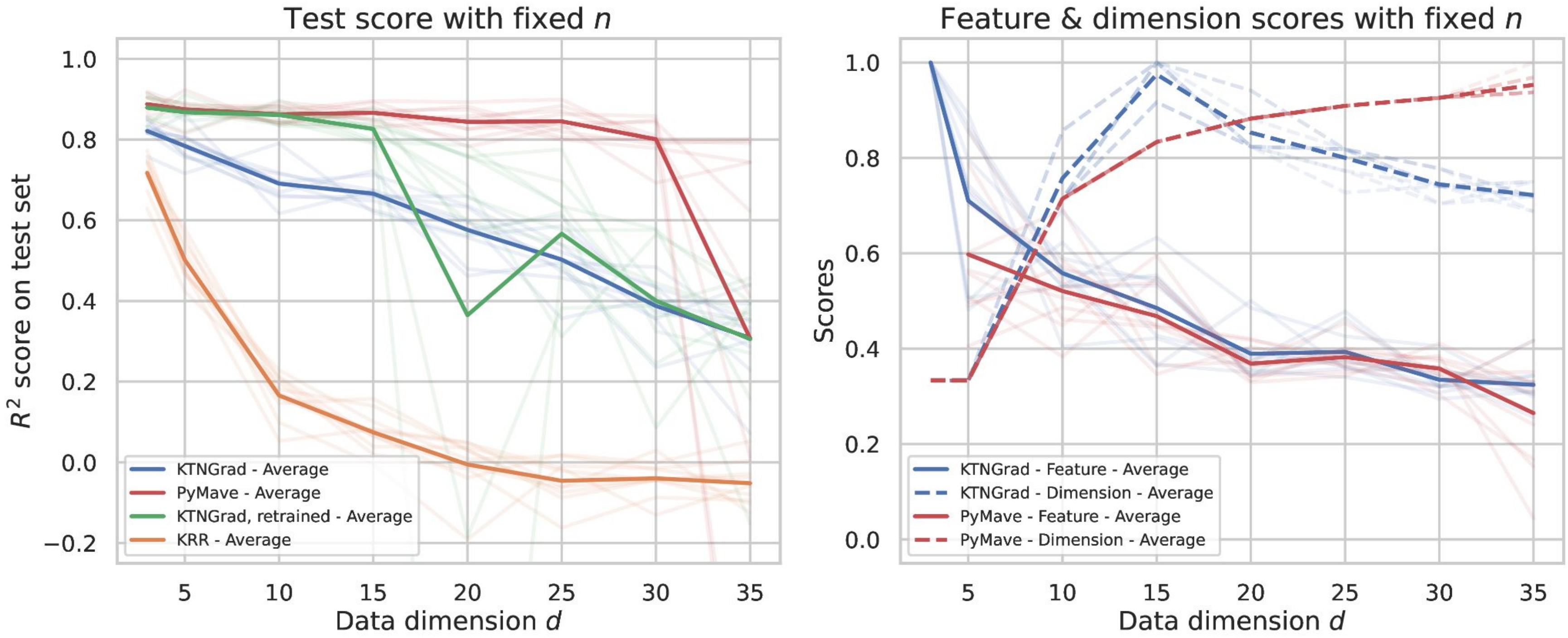



Figure 1: Performance for varying sample size, dimension 40, and a “sinus” dataset

# Analysis

## Convergence guarantees

Not exponential in the dimension!

But strong assumption $f^* \in \mathcal{H}$

Gaussian kernel: derivatives of all orders are square integrable

## Computational complexity

High cost per iteration: $O(n^3 d^4)$
but few iterations

## Feature space recovery

Consistent estimation of features
but difficulty with the dimension

## Inadequate function space

Gaussian RKHS and multi-index model are incompatible

$$f^*(x) = g^*(x_1) \quad \int_{\mathbb{R}^d} \left((g^*)'(x_1)\right)^2 \mathrm{d}x_1 \ldots \mathrm{d}x_d < \infty$$

Next: focus on function space choice
try a Hilbert space with relevant basis!

# Group Lasso Penalty on Hermite Polynomials Decomposition

RegFeaL

Chapter 3: B. Follain and F. Bach. Nonparametric Linear Feature Learning in Regression Through Regularisation. Electronic Journal of Statistics, 18(2):4075–4118. 2024

# Main Idea

## Hermite polynomials

$$h_0(x) = 1$$
$$h_1(x) = x$$
$$h_2(x) = \frac{1}{\sqrt{2}}(x^2 - 1)$$

$$H_\alpha(x) = \prod_{a=1}^{d} h_{\alpha_a}(x_a)$$

Orthonormal basis of $L^2(q)$

$$f = \sum_{\alpha \in \mathbb{N}^d} f_\alpha H_\alpha$$

$f$ does not depend on $x_a$

$\Updownarrow$

$$\forall \alpha \in (\mathbb{N}^d)^*,\ \alpha_a \neq 0 \implies f_\alpha = 0$$

$$\Omega_{\mathrm{var}}^r(f) = \sum_{a=1}^{d} \left( \sum_{\alpha \in (\mathbb{N}^d)^*} \alpha_a \frac{1}{c_{|\alpha|}} f_\alpha^2 \right)^{r/2}$$

$$\hat{f}_{\mathrm{var}} := \underset{f \in L^2(q)}{\mathrm{argmin}}\ \widehat{\mathcal{R}}(f) + \lambda \Omega_{\mathrm{var}}^r(f) + \mu \Omega_0^2(f)$$

(Hermite, 1864)

# Main Idea



$$\int_{\mathbb{R}^d} \left(\frac{\partial f}{\partial x_a}\right)\left(\frac{\partial f}{\partial x_b}\right) q = \sum_{\alpha \in \mathbb{N}^d} \sqrt{(\alpha_a + 1)}\sqrt{(\alpha_b + 1)} f_{\alpha + e_a} f_{\alpha + e_b}$$

$$M_f \in \mathbb{R}^{d \times d} \qquad (M_f)_{a,b} = \sum_{\alpha \in \mathbb{N}^d} \frac{1}{c_{|\alpha|+1}} \sqrt{\alpha_a + 1}\sqrt{\alpha_b + 1} f_{\alpha + e_a} f_{\alpha + e_b}$$

$$\operatorname{rank}(M_{f^*}) = s \qquad \Omega^r_{\text{feat}}(f) = \operatorname{trace}\left(M_f^{r/2}\right)$$

$$\hat{f}_{\text{feat}} := \operatorname*{argmin}_{f \in L^2(q)} \widehat{\mathcal{R}}(f) + \lambda \Omega^r_{\text{feat}}(f) + \mu \Omega^2_0(f)$$

## Reweighted formulation (again!)

$$\hat{f}_{\text{feat}}, \hat{\Lambda}_{\text{feat}} = \underset{\substack{f \in L^2(q),\, \Lambda \in \mathbb{R}^{d\times d} \\ \Lambda = R\,\mathrm{Diag}(\eta) R^\top \\ \sum_{a=1}^{d} \eta_a^{r/(2-r)} = 1}}{\operatorname{argmin}} \widehat{\mathcal{R}}(f) + \lambda \operatorname{trace}(\Lambda^{-1} M_f) + \mu \Omega_0^2(f)$$

Alternating minimisation in closed-form (or descent)

- For $\Lambda$ : formula using eigenpairs of $M_f$
- For $f$ : solving kernel ridge regression with

$$k_\Lambda(x, x') = \sum_{\alpha \in (\mathbb{N}^d)^*} \frac{c_{|\alpha|} H_\alpha(R^\top x) H_\alpha(R^\top x')}{\mu + \lambda \alpha^\top \eta^{-1}}$$

## Obtaining the features

Use $M_{\hat{f}_{\text{feat}}}$ to estimate

- the features (leading singular vectors)
- the dimension (using a threshold)

## Computation

Complicated sampling scheme of $\alpha$ using $\eta$, real limitation

(less) High cost per iteration

$$O\Big(\underbrace{nd\max(m,d)}_{\text{Hermite features}} + \underbrace{d^2(m)^2 + d^3}_{\text{Eigendecomposition}} + \underbrace{nm\max(n,m)}_{\text{Kernel Ridge}}\Big)$$

(Bach et al., 2011)

# Informal Theorem

**Convergence of the expected risk**

In the well-specified setting ($f^* \in L^2(q)$), with bounded inputs ($\|X\|_2 \leq R$), a convex $G$-Lipschitz loss, the optimal choice of $\lambda$ and for $c_k = \rho^k, \rho \in (0,1)$, then for any $\delta \in (0,1]$, with probability larger than $1-\delta$

$$\mathcal{R}(\hat{f}) \leq \mathcal{R}(f^*) + \Omega(f^*) \cdot \frac{G}{\sqrt{n}} \sqrt{1 + \frac{e^{R^2/2}}{(1-\rho)^d}} \left( 16\sqrt{\frac{\pi}{2}} + 4\sqrt{2}\sqrt{\log \frac{2}{\delta}} \right)$$

**Proof technique**

Gaussian complexity $\quad G_n(\mathcal{G}) := \mathbb{E}_{(x_i)_{i\in[n]},(\varepsilon_i)_{i\in[n]}} \left( \sup_{f\in\mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \varepsilon_i f(x_i) \right)$

(Bartlett and Mendelson, 2002)

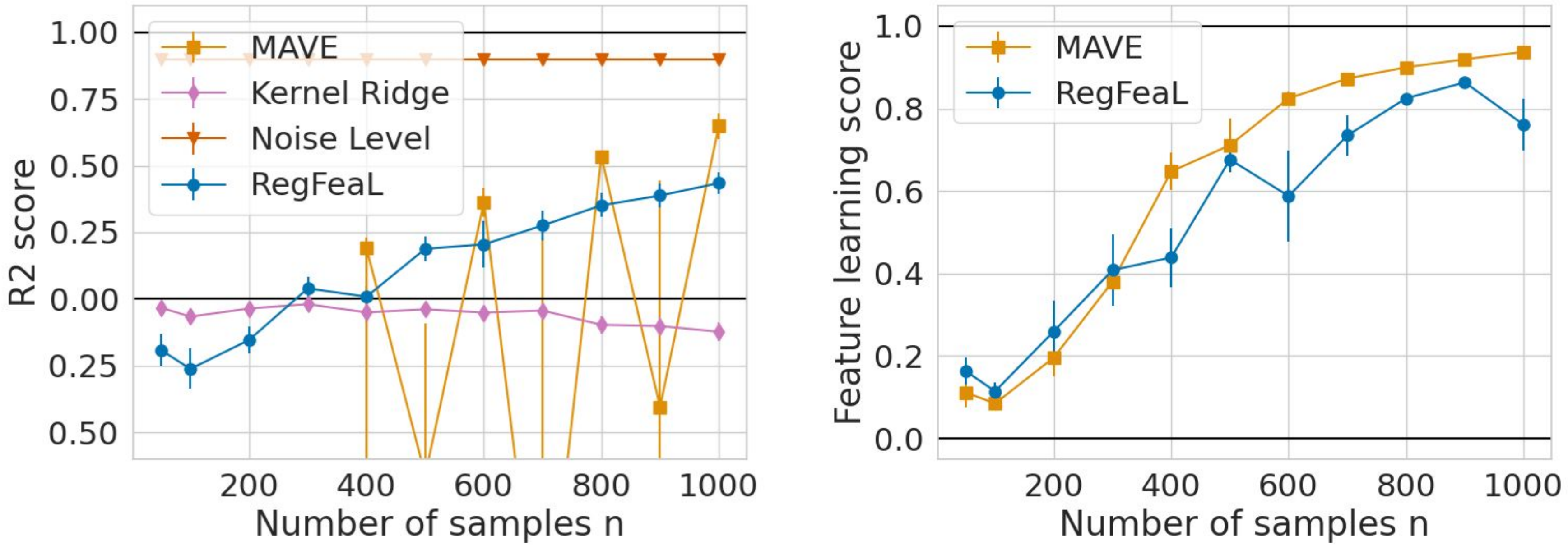



Figure 2: Performance for varying sample size, dimension 40, and a “polynomial” dataset

# Numerical Experiments

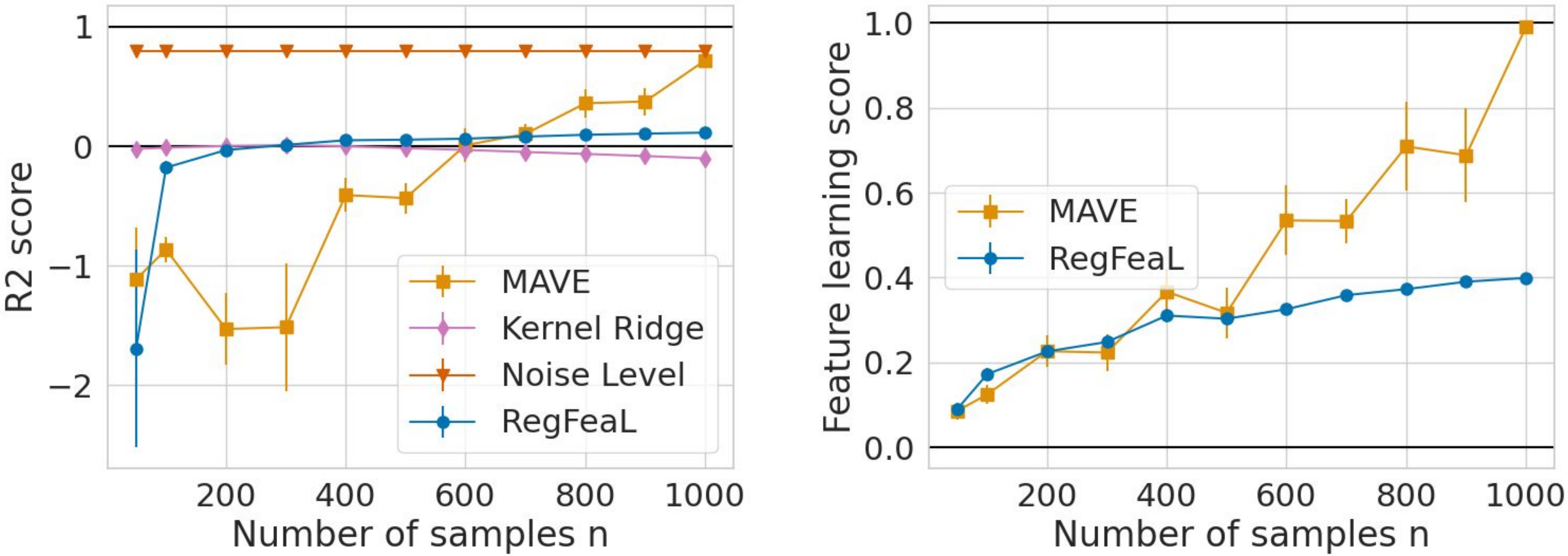



Figure 3: Performance for varying sample size, dimension 40, and a “sinus” dataset

# Analysis

## Use of Hermite polynomials

Well-suited basis for feature learning

Advantage compared to previous RKHS

## Computational complexity

Infinite basis leads to awkward sampling to approximate kernel at each step

## Generalisation guarantees

No strong assumptions on $f^*$ ( $L^2(q)$ large)

But exponential dependency in $d$

## Function space limitations

The space is actually too big! $f = \sum_{\alpha \in \mathbb{N}^d} \hat{f}(\alpha) H_\alpha$

Infinite basis complicates matters...

Next: use inspiration from neural networks!

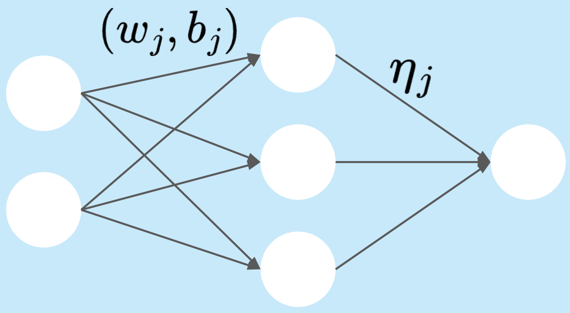
# Integrating Neural Networks And Kernel Methods

BKerNN

Chapter 4: B. Follain and F. Bach. Enhanced Feature Learning via Regularisation: Integrating Neural Networks and Kernel Methods. 2024 (under review by JMLR).

# Main Idea

## Feedforward neural network (ReLU)




$$f(x) = \frac{1}{m}\sum_{j=1}^{m} \eta_j (w_j^\top x + b_j)_+$$

Infinite-width (mean-field) limit

$$f(x) = \int \eta\sigma(w^\top x + b)\, \mathrm{d}\mu(\eta, w, b)$$

(Kurkova and Sanguineti, 2001)

## Neural network/kernel method fusion *

$f \in \mathcal{F}_\infty$

$$f(x) = c + \int_{\mathcal{S}^{d-1}} g_w(w^\top x)\, \mathrm{d}\mu(w)$$

$$\Omega_0(f) = \int_{\mathcal{S}^{d-1}} \|g_w\|_{\mathcal{H}}\, \mathrm{d}\mu(w) < +\infty$$

RKHS and Brownian kernel

$$\mathcal{H} := \{g : \mathbb{R} \to \mathbb{R} \mid g(0) = 0,\ \int_{\mathbb{R}} (g')^2 < \infty\}$$

$$k(a, b) = (|a| + |b| - |a - b|)/2 = \min(|a|, |b|)\mathbb{1}_{ab>0}$$

*More rigorous definition in manuscript

# Main Idea

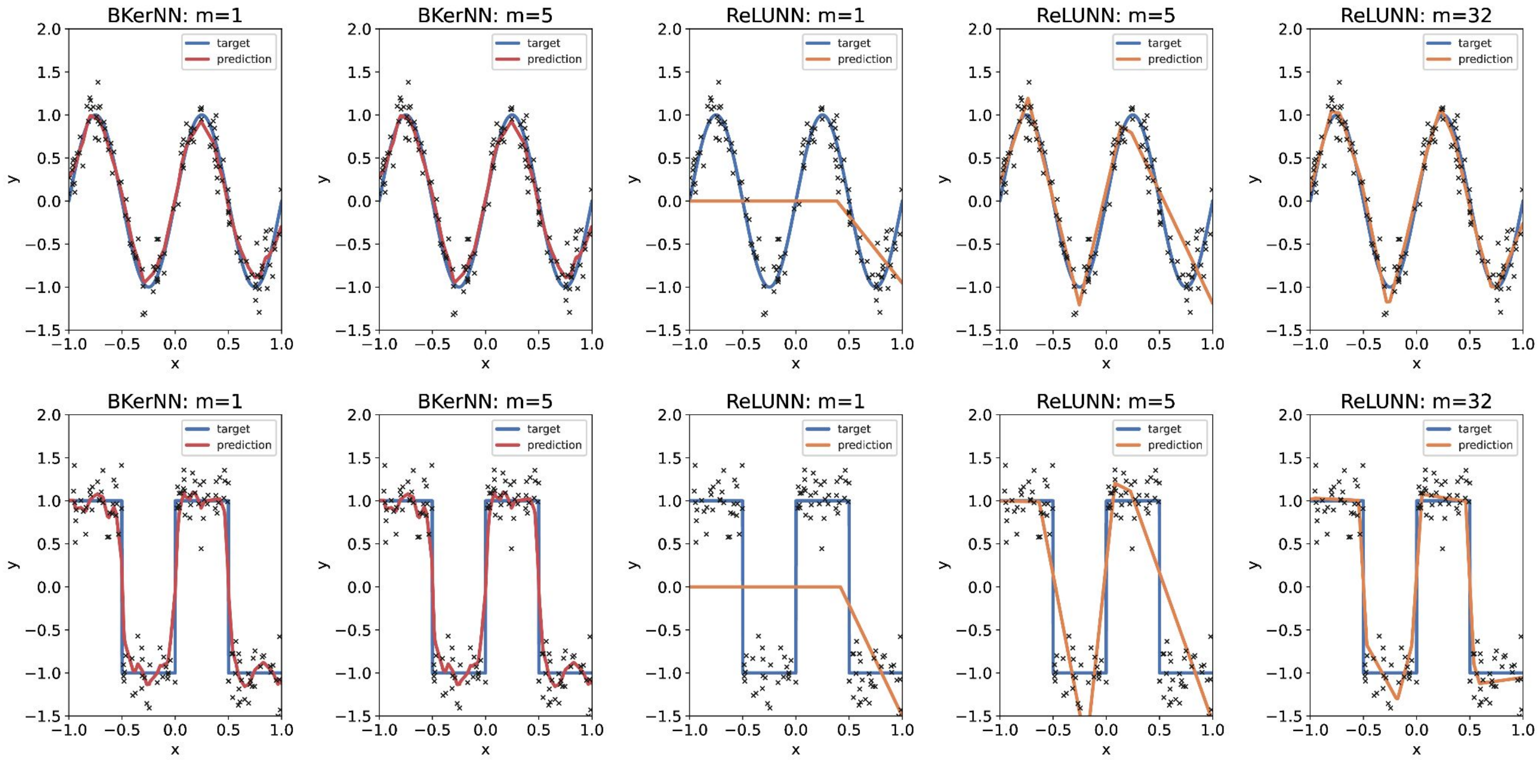



Figure 4: Comparison to neural networks on 1D examples

# Main Idea

## In practice: use particles

$f \in \mathcal{F}_m \qquad f(x) = c + \frac{1}{m}\sum_{j=1}^{m} g_j(w_j^\top x)$

## Other penalties

Replace by $\Omega_{\text{weights}}(w_1, \ldots, w_m)$ for variable selection or feature learning

## Reformulation

$$\min_{w_1,\ldots,w_m \in \mathbb{R}^d, c \in \mathbb{R}, \alpha \in \mathbb{R}^n} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, (K\alpha)_i + c) + \frac{\lambda}{2}\alpha^\top K \alpha + \frac{\lambda}{2}\frac{1}{m}\sum_{j=1}^{m} \|w_j\|$$

$$K = \frac{1}{m}\sum_{j=1}^{m} K^{(w_j)} \qquad K_{i,i'}^{(w_j)} = k(w_j^\top x_i, w_j^\top x_i')$$

**Positive 1-homogeneity of the Brownian kernel** $K^{(\kappa w_j)} = \kappa K^{(w_j)}$

# Algorithm



## Step 1: fixing the weights/kernel

Kernel ridge regression problem

$$O(n^3 + n^2 d)$$

Solution: Nyström?

## Step 2: learning the weights

No closed-form, proximal gradient descent

$$w_j \leftarrow \mathrm{prox}_{\lambda\gamma\Omega}\left(w_j - \gamma \frac{\partial G}{\partial w_j}\right)$$

$$O(md\min(m,d)) \qquad \mathrm{prox}_{\lambda\gamma\Omega_0}(u) = \left(1 - \frac{\lambda\gamma}{2m}\frac{1}{\|u\|}\right)_+ u$$

$$\min_{w_1,\ldots,w_m\in\mathbb{R}^d, c\in\mathbb{R}, \alpha\in\mathbb{R}^n} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, (K\alpha)_i + c) + \frac{\lambda}{2}\alpha^\top K\alpha + \frac{\lambda}{2}\frac{1}{m}\sum_{j=1}^{m}\|w_j\|$$

$$K = \frac{1}{m}\sum_{j=1}^{m} K^{(w_j)}$$

(Drineas and Mahoney, 2005) (Chizat and Bach, 2022)

# Algorithm

## Step 1: fixing the weights/kernel

Kernel ridge regression problem

$$O(n^3 + n^2 d)$$

Solution: Nyström?

## Step 2: learning the weights

No closed-form, proximal gradient descent

$$w_j \leftarrow \mathrm{prox}_{\lambda\gamma\Omega}\left(w_j - \gamma \frac{\partial G}{\partial w_j}\right)$$

$O(md\min(m,d))$ $\quad \mathrm{prox}_{\lambda\gamma\Omega_0}(u) = \left(1 - \frac{\lambda\gamma}{2m}\frac{1}{\|u\|}\right)_+ u$

## Obtaining the features

Use $(w_j)_{j\in[m]}$ to estimate

- the features (leading singular vectors)
- the dimension (using a threshold)

## Optimisation behaviour

No formal proof (differentiability issues)

But: insights from mean-field theory

& well-behaved in practice

(Drineas and Mahoney, 2005) (Chizat and Bach, 2022)

# Informal Theorem

## Convergence of the expected risk

In the well-specified setting ( $f^* \in \mathcal{F}_\infty$ ), with $1+\sqrt{\|X\|^*}$ being subgaussian with variance proxy $\sigma^2$, a convex $G$-Lipschitz loss, the optimal choice of $\lambda$, with $C, C'$ universal constants, then for any $\delta \in (0,1]$, with probability larger than $1-\delta$

$$\mathcal{R}(\hat{f}) \leq \mathcal{R}(f^*) + \Omega_0(f^*) CG \left( \frac{1}{\sqrt{n}} + G_n + \frac{\sigma}{\sqrt{n}} \sqrt{\log \frac{1}{\delta}} \right)$$

## Bound on Gaussian complexity

$$G_n \leq C' \min \left( \sqrt{\frac{d}{n}} \sqrt{\log(n)} \sqrt{\mathbb{E}_X \|X\|^*}, \ \frac{1}{n^{1/6}} (\log d)^{1/4} \left( \mathbb{E}_{X_1 \ldots X_n} \left( \max_{i \in [n]} \|X_i\|^* \right)^2 \right)^{1/4} \right)$$

(Bartlett and Mendelson, 2002)

# Numerical Experiments

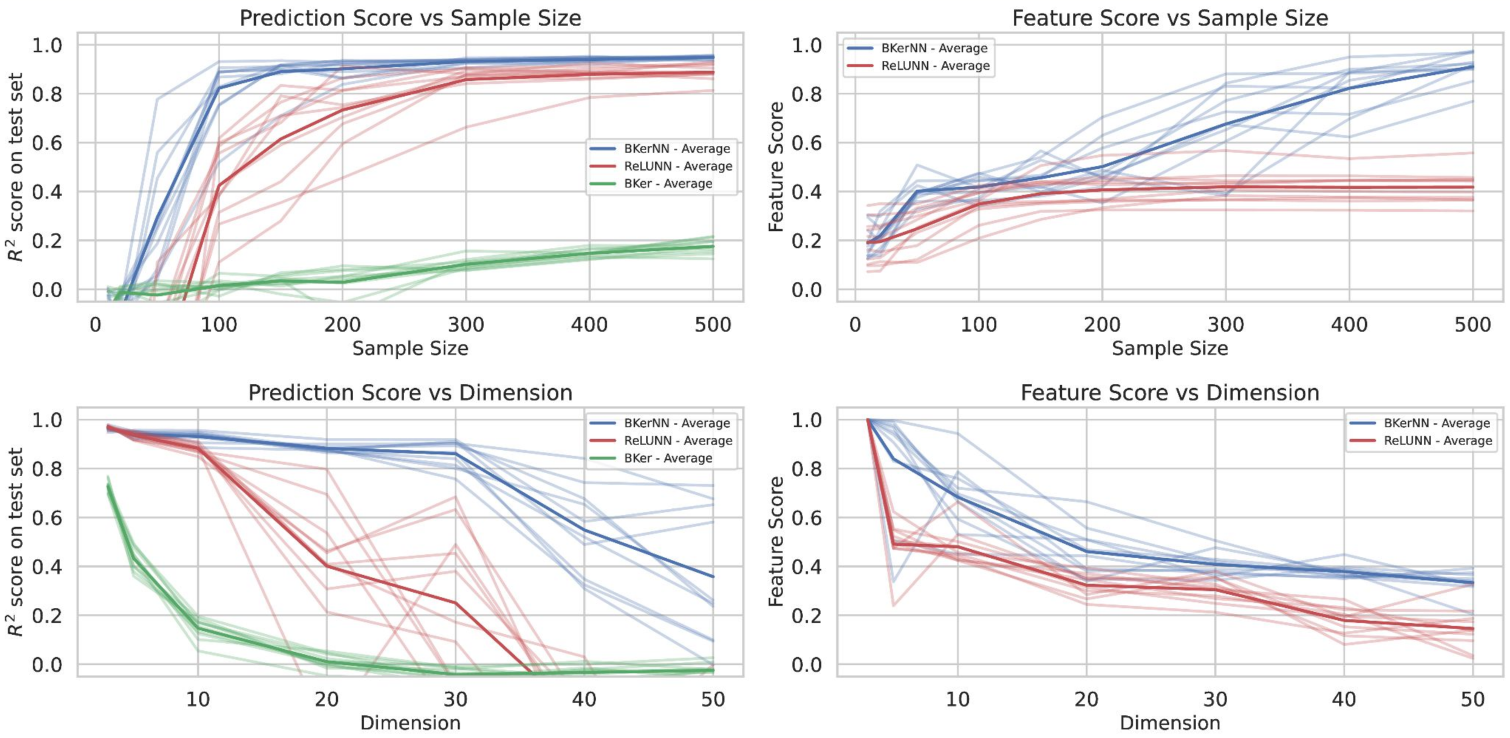



Figure 5: Performance comparison across varying sample sizes and dimensions

# Numerical Experiments

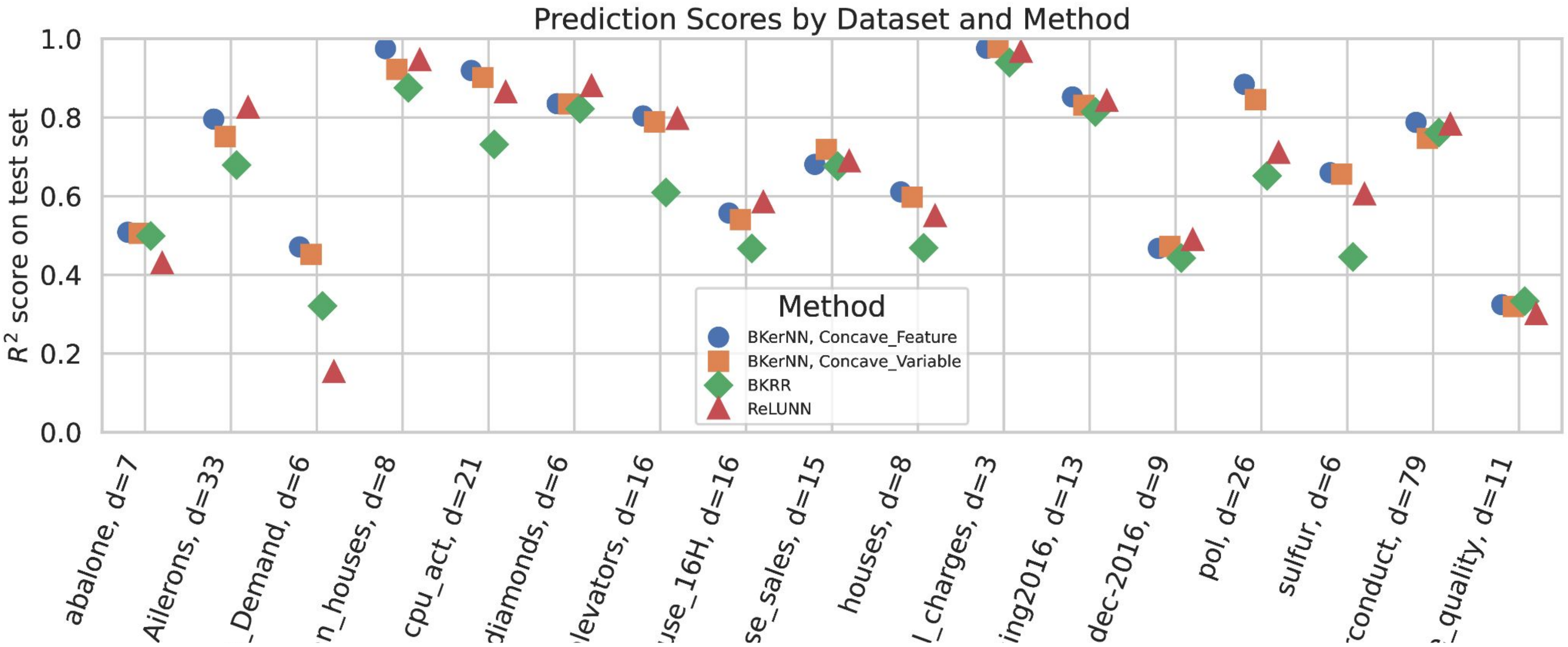



Figure 6: Performance comparison across real datasets

# Analysis

## Generalisation guarantees

Lightest assumption on data

No exponential dependency on $d$

Light assumption on $f^*$ ($\mathcal{F}_\infty$ large)

## In practice

Good performance and easy to train

Reasonable computational cost

## Adaptivity in misspecified settings

| No | Yes |
| --- | --- |
| Kernel methods | Neural networks (BKerNN) |

## Function space analysis

Linear features encoded in design of function space

Yet still large function space (as seen by using Fourier transform analysis)

(Bach, 2024)

# Conclusion

# Goals and Achievements

Multi-Index Model

$$f^*(\cdot) = g^*(P^\top \cdot)$$

$$P \in \mathbb{R}^{d \times s}, s \ll d$$

**Careful design of appropriate function space and penalty**

Regular RKHS

Hermite polynomials Hilbert space

Neural net/kernel fusion

$$\hat{f} := \operatorname*{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(x_i)) + \lambda \Omega(f)$$

**Goals**

- Use regularised empirical risk minimisation (RERM) for flexibility
- Make few assumptions on data distribution
- Limit functional assumptions on $f^*$
- Obtain theoretical bounds with limited dependence on data dimension $d$

# Goals and Achievements

Multi-Index Model

$$f^*(\cdot) = g^*(P^\top \cdot)$$
$$P \in \mathbb{R}^{d\times s},\ s \ll d$$

**Careful design of appropriate function space and penalty**

Regular RKHS

Hermite polynomials Hilbert space

Neural net/kernel fusion

$$\hat{f} := \underset{f\in\mathcal{F}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(x_i)) + \lambda\Omega(f)$$

**Achievements**

- Sparsity-inducing penalties: trace norm regularisation
- Computable methods: representer theorem, alternating minimisation
- Progress on quest for adapted function space
- Last method has limited assumptions and no exponential dependency!

# Take-Home Message

**Betting on Sparsity:** Leveraging Hidden Linear Features through Regularisation for Supervised Learning

No free lunch theorem
Some assumptions must be made: few relevant features

Bypass high-dimensional issues while limiting other assumptions
by
**learning linear features!**

# Perspectives

Improving computation of BKerNN
Explicit adaptivity results for BKerNN

Extension to other function estimation problems
beyond i.i.d. covariates/response pairs

Exploration of non-linear feature learning
to capture more complex patterns

# Thank you!

# Extra: Bound on Gaussian Complexity for BKerNN

$$G_n(\{f \in \mathcal{F}_\infty \mid \Omega(f) \leq D\}) \leq D \left( \frac{1}{\sqrt{n}} + G_n \right)$$

$$G_n := \mathbb{E}_{\varepsilon, \mathcal{D}_n} \left( \sup_{\|g\|_{\mathcal{H}} \leq 1, w \in \mathcal{S}^{d-1}} \frac{1}{n} \sum_{i=1}^{n} \varepsilon_i g(w^\top x_i) \right)$$

**Option 1: Dimension-dependent bound** $G_n \leq 8 \sqrt{\frac{d}{n}} \sqrt{\log(n+1)} \sqrt{\mathbb{E}_X \|X\|^*}$

Step 1: Optimise explicitly for $g$

$$G_n = \mathbb{E}_{\varepsilon, \mathcal{D}_n} \left( \sup_{w \in \mathcal{S}^{d-1}} \frac{\sqrt{\varepsilon^\top K^{(w)} \varepsilon}}{n} \right)$$

Step 2: Use $\zeta$-covering of $\mathcal{S}^{d-1}$ in $\|\cdot\|$ norm

$$M \leq (1 + 2/\zeta)^d$$

# Extra: Bound on Gaussian Complexity for BKerNN

**Option 2: Dimension-independent bound**

$$G_n \leq \frac{6}{n^{1/6}} \left( (\log 2d)^{1/4} \mathbb{1}_{*=\infty} + \mathbb{1}_{*=2} \right) \left( \mathbb{E}_{\mathcal{D}_n} \left( \max_{i \in [n]} (\|X_i\|^*)^2 \right) \right)^{1/4}$$

Step 1: Use Lipschitz approximation for $g$

$$\exists\ (1/\zeta) - \textbf{Lipschitz}\ g_\zeta : \mathbb{R} \to \mathbb{R}, g_\zeta(0) = 0, \|g - g_\zeta\|_\infty \leq \zeta$$

Step 2: Use covering of 1-Lipschitz set of functions in $\|\cdot\|_\infty$ norm

Step 3: Use Lemma based on Slepian's lemma and Bartlett and Mendelson (2002)

$$\mathbb{E}_\varepsilon \left( \sup_{h \in \{h_1, \ldots, h_M\}, w \in \mathcal{S}^{d-1}} \frac{1}{n} \sum_{i=1}^n \varepsilon_i h(w^\top x_i) \right) \leq \mathbb{E}_\varepsilon \left( \left\| \frac{\sqrt{2}}{n} \sum_{i=1}^n \varepsilon_i x_i \right\|^* + \sqrt{8 \frac{\sum_{i=1}^n (\|x_i\|^*)^2}{n^2}} \sqrt{2 \log M} \right)$$