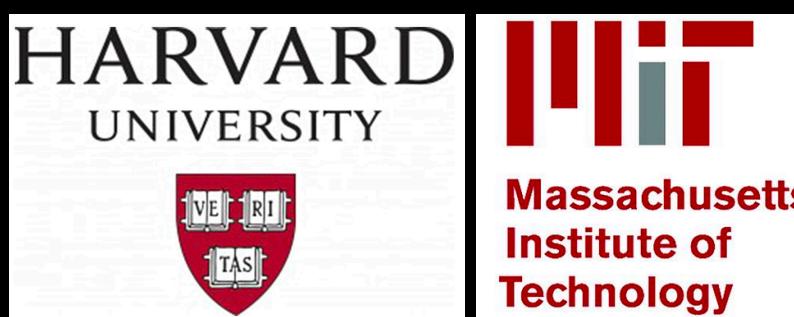
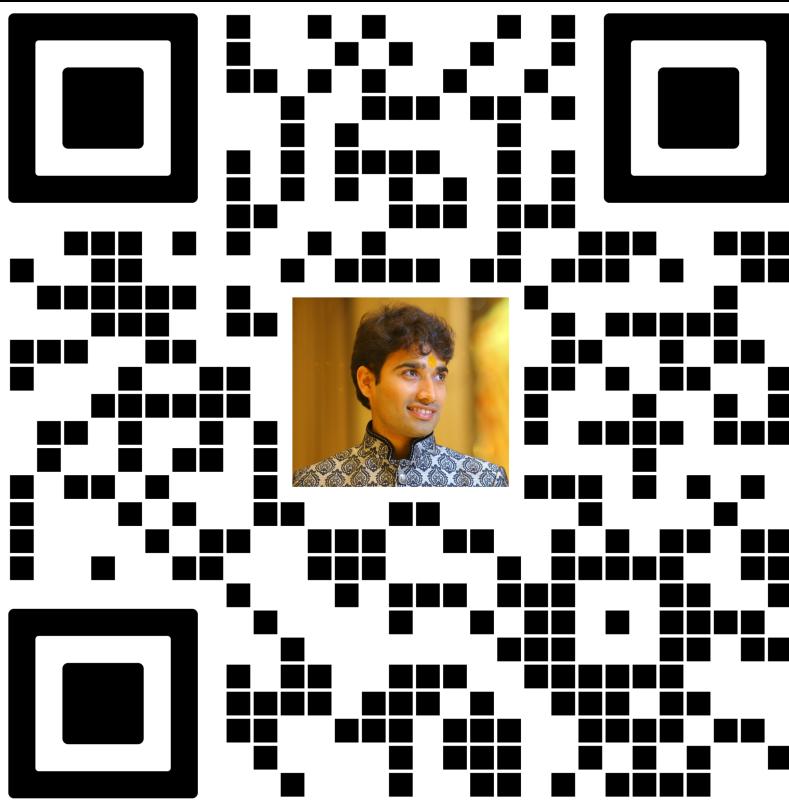


Revisiting minimum description length complexity for overparameterized models

Raaz Dwivedi, Chandan Singh, Bin Yu, Martin Wainwright

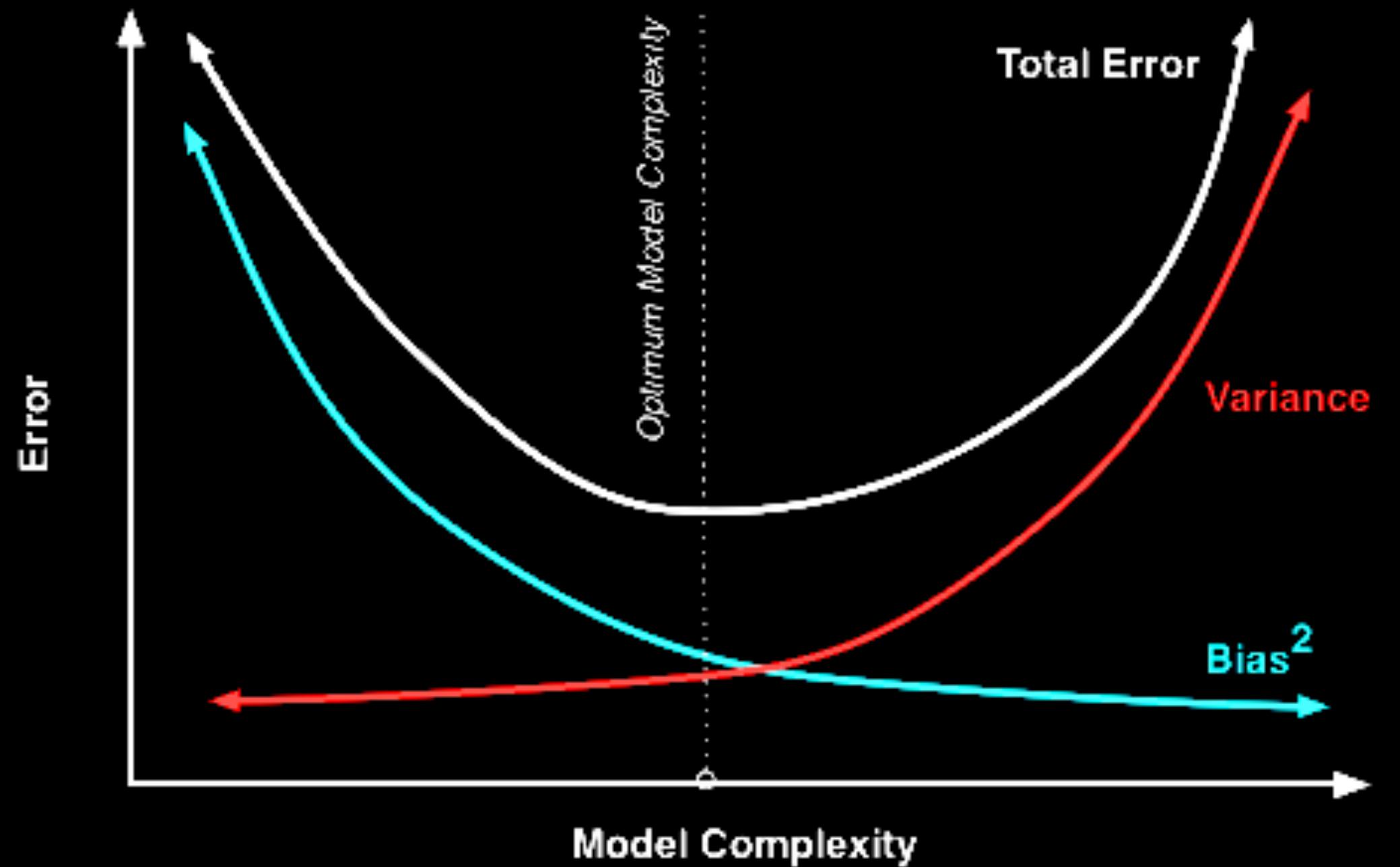


<https://rzs.github.io>

Symposium on Algorithmic Information Theory and Machine Learning,
Alan Turing Institute, July 4, 2022

Classical Wisdom: **Occam's razor**
Use the simplest model that fits the data

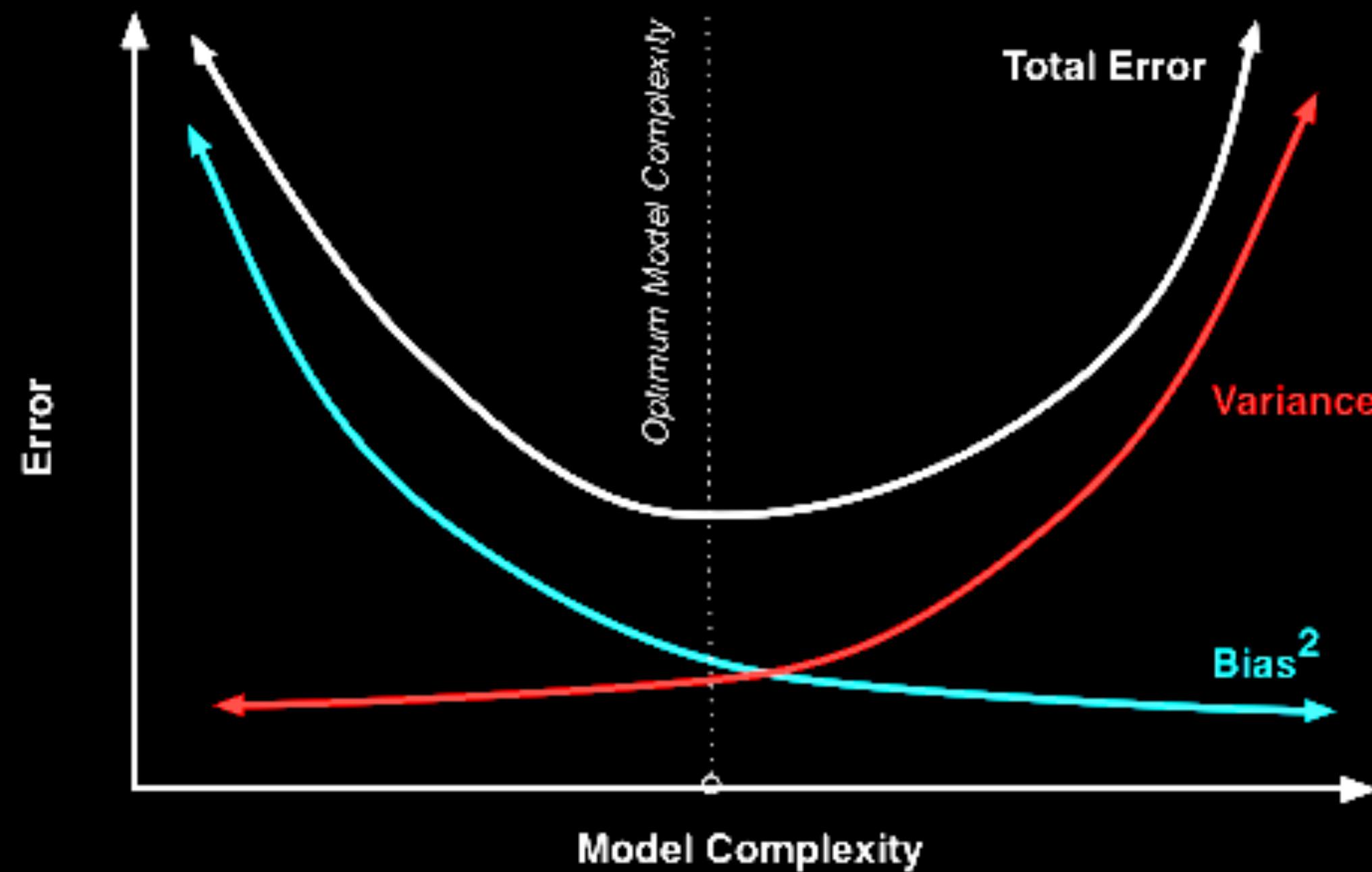
U-shaped bias-variance tradeoff: Low-dimensional settings with “good” estimators



Classical Wisdom: **Occam's razor**

Use the simplest model that fits the data

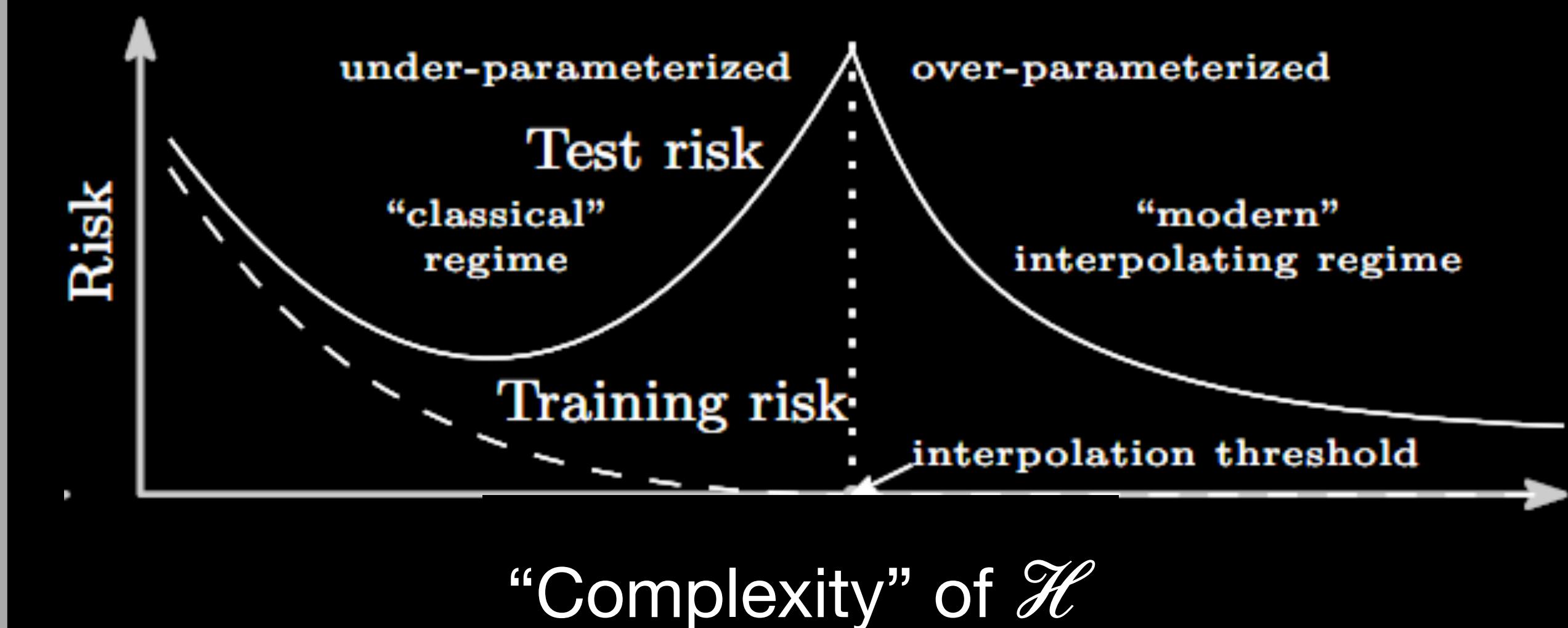
U-shaped bias-variance tradeoff:
Low-dimensional settings with
“good” estimators



Modern Phenomenon: **Fit without fear!**

Use the largest model that achieves zero training error

Non-U shaped risk curves in
modern overparameterized models



..., Skurichina-Duin 98,00, Belkin-Hsu-Ma-Mandal 18,
Muthukumar-Vodrahalli-Sahai 19, Hastie-Montanari-Rosset-
Tibshirani 19, ...

Is double descent a conundrum for the classical wisdom?

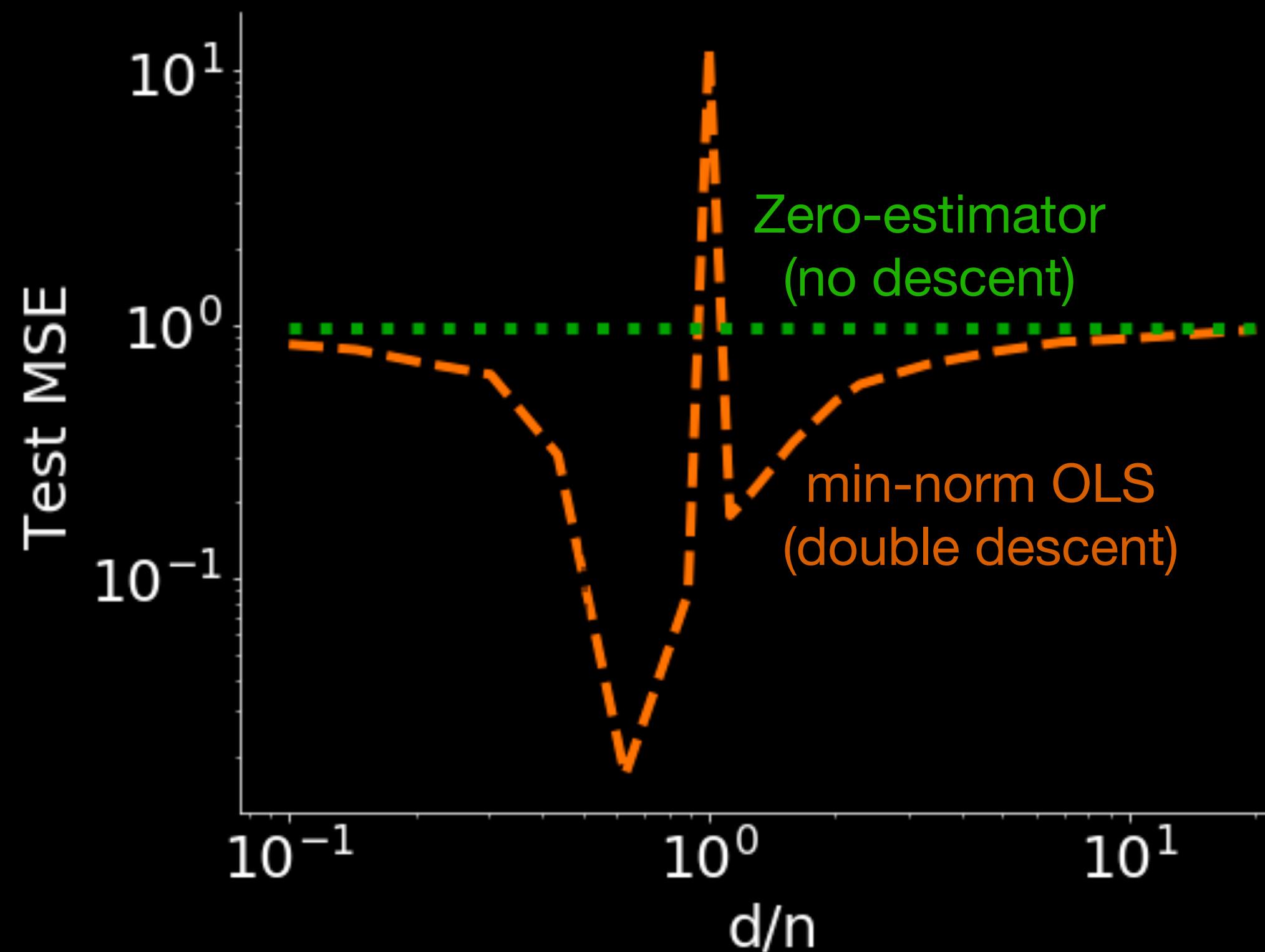
We expect the classical U-shaped tradeoff given a **fixed dataset**, and as the “**complexity**” of the fitted estimator varies

d = number of features

n = number of samples

Is double descent a conundrum for the classical wisdom?

We expect the classical U-shaped tradeoff given a **fixed dataset**, and as the “**complexity**” of the fitted estimator varies

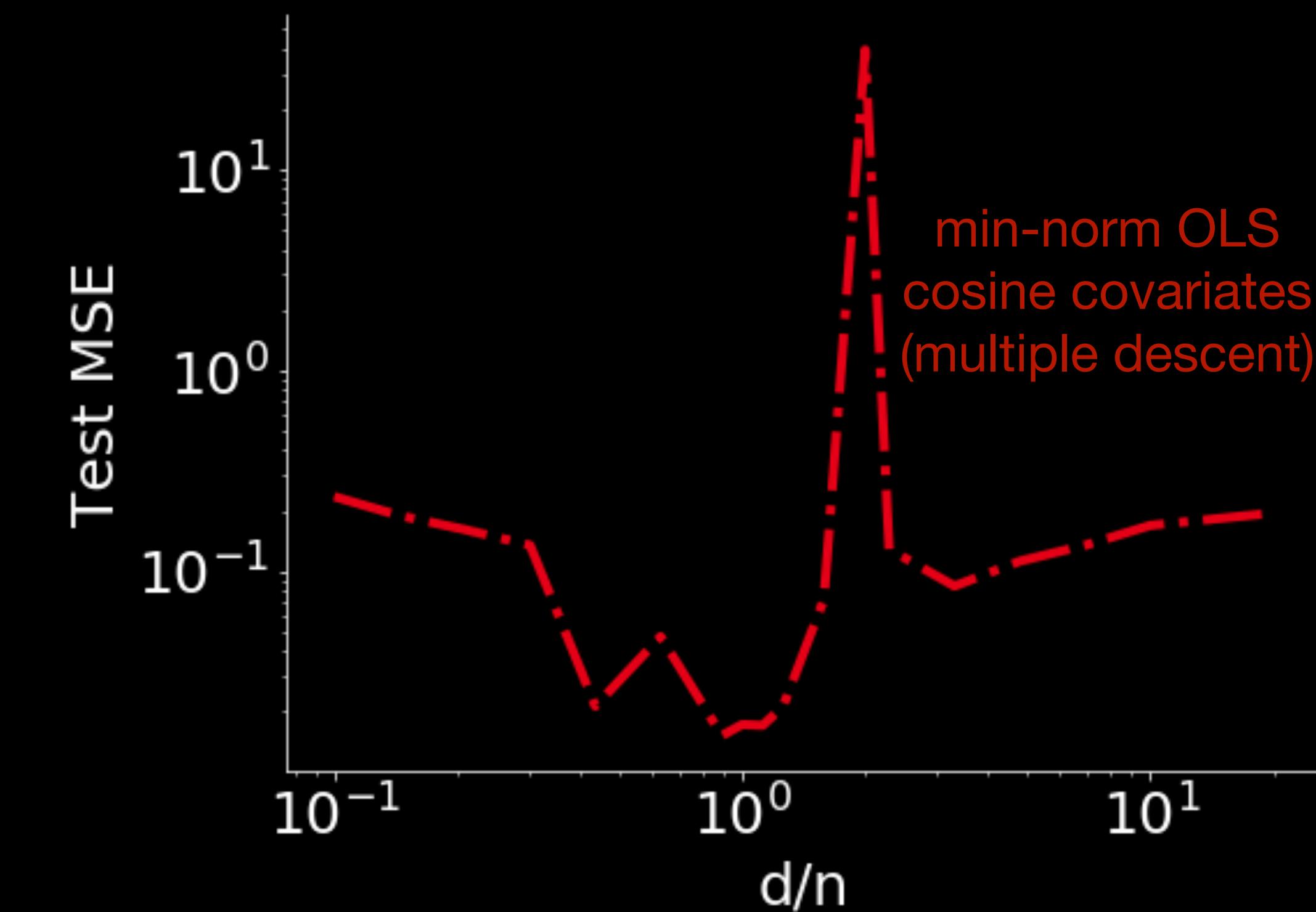
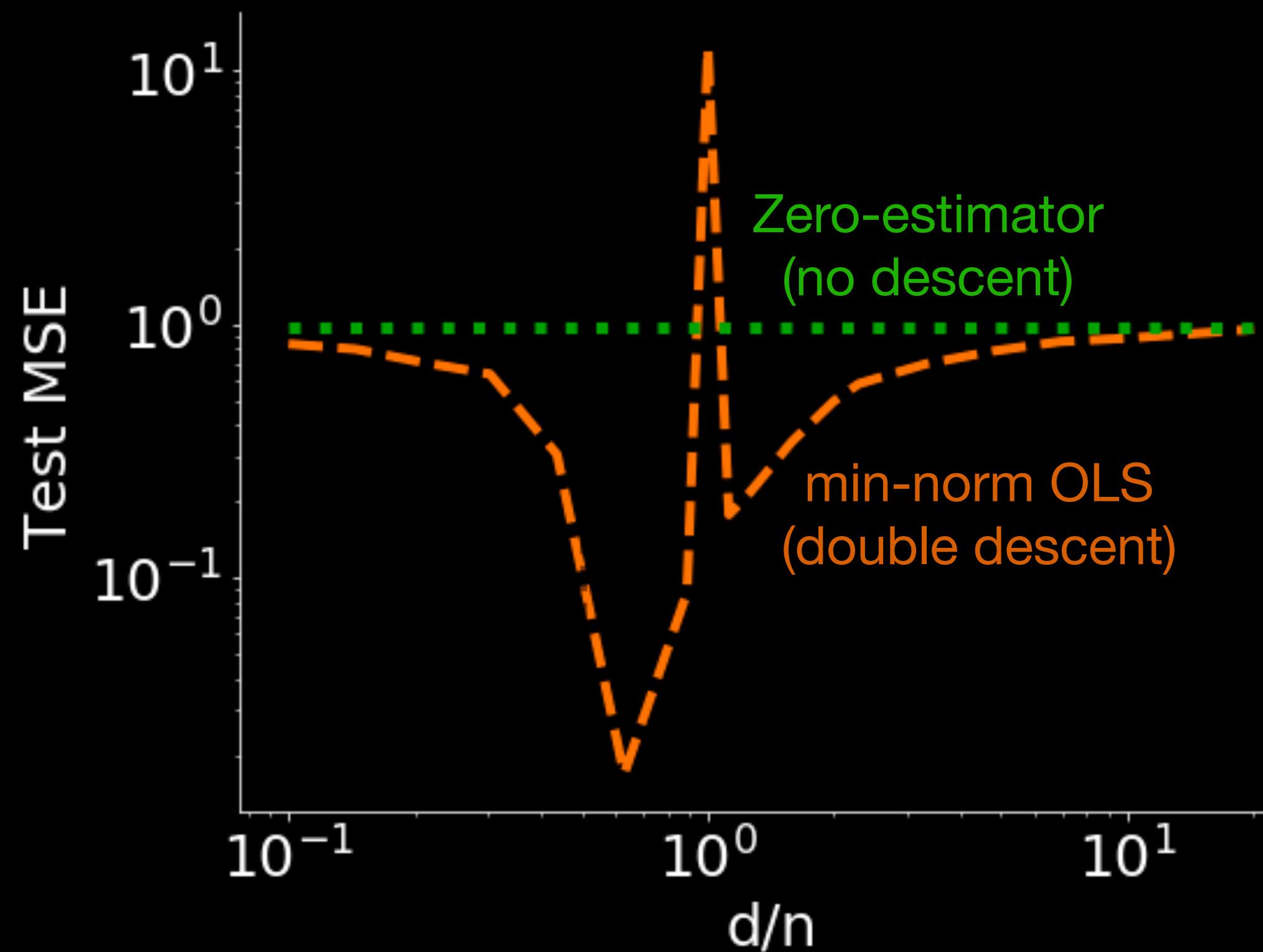


d = number of features

n = number of samples

Is double descent a conundrum for the classical wisdom?

We expect the classical U-shaped tradeoff given a **fixed dataset**, and as the “**complexity**” of the fitted estimator varies

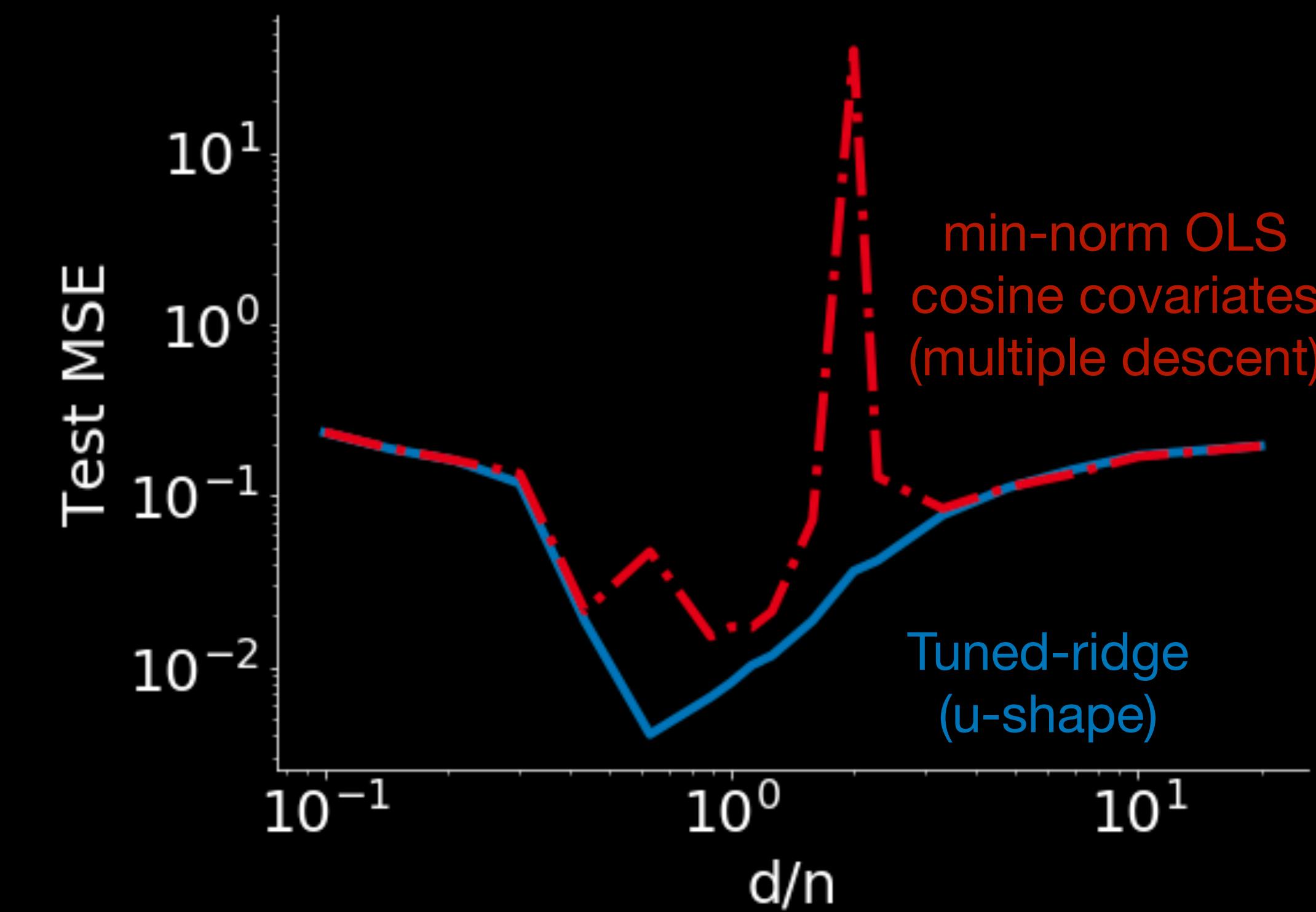
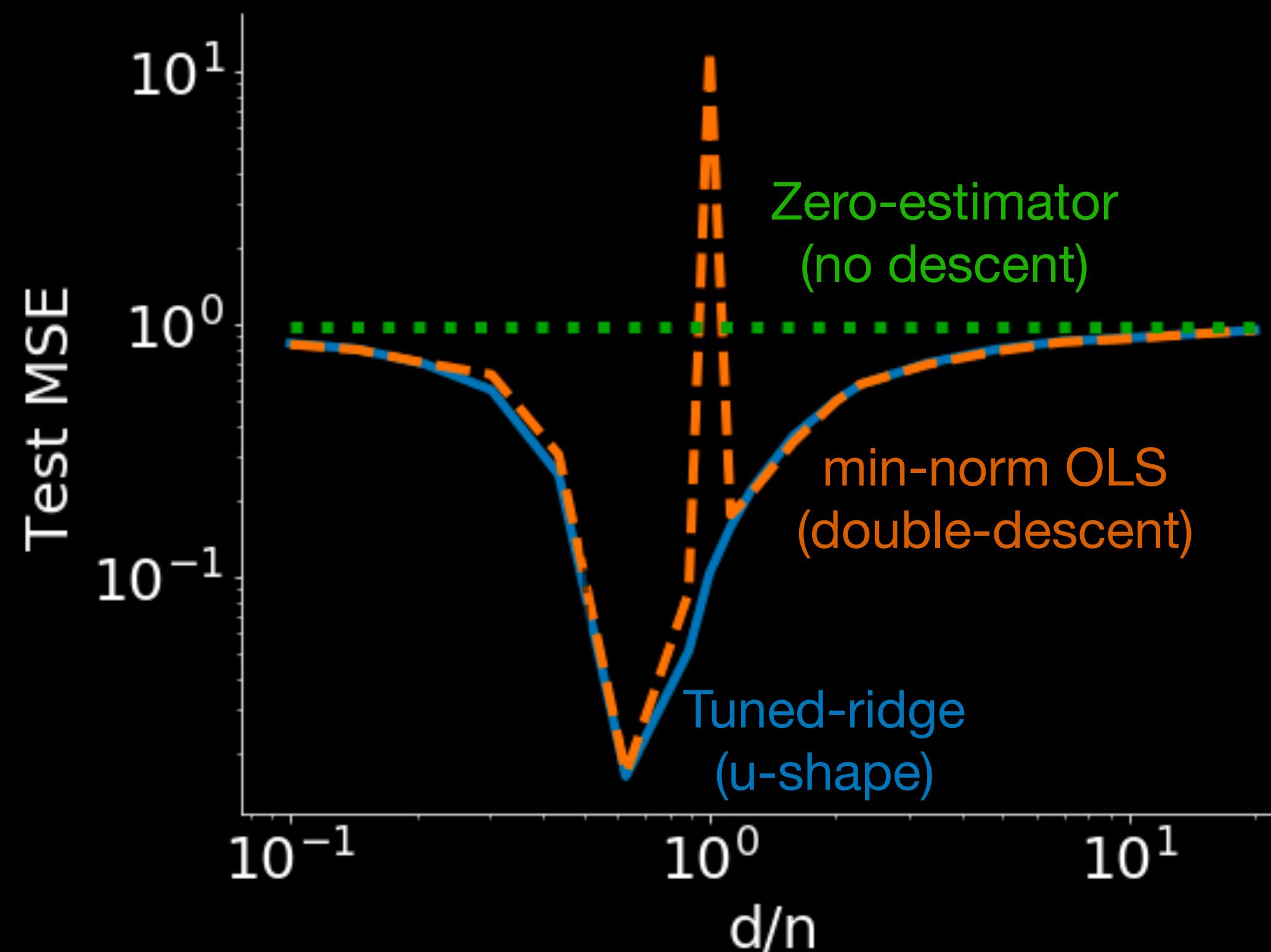


d = number of features

n = number of samples

Is double descent a conundrum for the classical wisdom?

We expect the classical U-shaped tradeoff given a **fixed dataset**, and as the “**complexity**” of the fitted estimator varies [in a good estimator class](#)



d = number of features

n = number of samples

Is double descent a conundrum for the classical wisdom?

We expect the classical U-shaped tradeoff given a **fixed dataset**, and as the “**complexity**” of the fitted estimator varies in a good estimator class

Need to pay attention to

- the estimator class, as well as
- the complexity measure

Is parameter counting a valid complexity measure,
especially for $d \gg n$?

d = number of features

n = number of samples

Complexity: A tricky concept

A fundamental notion: Kolmogorov's algorithmic complexity

Complexity in Statistics and ML: Useful for model selection

- Test error \sim Train error + Complexity / n^a
- x-axis in bias-variance—often vaguely defined; parameter count often used

Parameter counting as complexity

Origins for linear regression: $y = X\theta^* + \varepsilon$

- Akaike Information Criterion (AIC): $d/2$
- Bayesian information criterion (BIC): $\frac{d}{2} \log n$
- Rademacher complexity: $\mathbb{E} \left[\sup_{\theta \in \Theta} \sum \varepsilon_i x_i^\top \theta \right] \sim d$
- Degrees of freedom: $\text{trace}(X^\top X) \sim d$
- Vapnik-Chervonenkis dimension: d
- Minimum description length complexity: $\frac{d}{2} \log n$ (asymptotically)

OLS achieves the minimax error of
order $\frac{d}{n}$ in low-dimensions $d \ll n$

d = number of features
 n = number of samples

but in high-dimensions these complexity measure neither work nor theoretically well-justified

this talk:
a data-dependent complexity using minimum description length that is not just parameter count

Minimum Description Length (MDL)

“Choose the model that gives the shortest description of data”

- Developed by Rissanen with roots in Kolmogorov's algorithmic complexity, also seen as a computable variant based on Shannon's information theory
- **Probability models interpreted as codes:**
A probability model Q on \mathcal{Y} \longleftrightarrow Encoding observation y with $\log(\frac{1}{Q(y)})$ bits
Good fit \longleftrightarrow Shorter codelength (description)
- Over the years: Two-stage MDL, mixture MDL, normalized maximum likelihood

Optimal codes from an MDL perspective

With correctly specified generative model

Two possible objectives

- Minimum expected redundancy

$$\min_q \textcolor{red}{\mathbb{E}}_{y \sim p^*} \left[\log \left(\frac{1}{q(y)} \right) - \log \left(\frac{1}{p^*(y)} \right) \right]$$

- Minimum worst-case redundancy

$$\min_q \max_y \left[\log \left(\frac{1}{q(y)} \right) - \log \left(\frac{1}{p^*(y)} \right) \right]$$

Optimal code $q = p^*$

With unspecified generative model

One objective can be generalized

- Worst-case redundancy w.r.t. a postulated class of codes $\{p_\theta, \theta \in \Theta\}$

$$\min_q \max_y \left[\log \left(\frac{1}{q(y)} \right) - \min_\theta \log \left(\frac{1}{p_\theta(y)} \right) \right]$$

Optimal code: Normalized maximum likelihood (NML)

$$q_{NML}(y) = \frac{\max_\theta p_\theta(y)}{\int \max_{\theta'} p_{\theta'}(z) dz}$$

$$\text{NML Complexity} = \int \max_{\theta} p_{\theta}(y) dy$$

= Worst-case redundancy

= Expected redundancy under known P^*

$\sim \frac{d}{2} \log n$ with d -dimensional parametric codes ($d \ll n$)

- However when $\int \max_{\theta} p_{\theta}(y) dy = \infty$, the q_{NML} code is not defined.

- E.g., even for the linear model $y = X\theta^* + \varepsilon$ with Gaussian noise & linear codes:

$$\text{when } y \in \mathbb{R}^n \implies \int \max_{\theta} p_{\theta}(y) dy \propto \int \exp\left(-\frac{1}{2\sigma^2} \|X\hat{\theta}_{OLS} - y\|^2\right) dy = \infty.$$

This talk: Tackling the infinity problem of NML

- Prior fixes: Truncate the output space [Barron-Rissanen-Yu 98]
- This talk: Regularization + Modified NML complexity
 - In particular, using **ridge regularization** as that allows to obtain best prediction performance across range of models and also enable analytical calculations

Ridge luckiness normalized maximum likelihood

Weight the codes with a luckiness function on parameter space

- LNML principle: $\max_{\theta} p_{\theta}(y) \leftrightarrow \max_{\theta} p_{\theta}(y)w_{\theta}$ for some ``luckiness function'' w_{θ}

- Our approach: Choose luckiness factors $w_{\theta, \Lambda}$ induced by ridge regularization

- For linear models, each $w_{\theta, \Lambda}$ leads to a modified LNML code q_{Λ} :

$$q_{\Lambda}(y) \propto \exp \left(-\frac{1}{2\sigma^2} \|X\hat{\theta}_{\Lambda} - y\|^2 - \frac{1}{2\sigma^2} \hat{\theta}_{\Lambda}^T \Lambda \hat{\theta}_{\Lambda} \right)$$

with $\hat{\theta}_{\Lambda} = \min_{\theta} \|X\theta - y\|^2 + \theta^T \Lambda \theta = (X^T X + \Lambda)^{-1} X^T y$

Defining the MDL-Complexity

Via the optimal LNML code in a rich ridge class

- To derive complexity: Optimize over Λ — $\min_{\Lambda} \mathbb{E}_{y \sim p^*} \left[\log \left(\frac{1}{q_{\Lambda}(y)} \right) - \log \left(\frac{1}{p^*(y)} \right) \right] =: R_{opt}$
- Choose a rich class of Λ : $\{UDU^T, D \succeq 0\}$ with U = eigenvectors($X^T X$)
- Account codelength for Λ (not present in usual NML):

$$\mathcal{L}(\Lambda) = \sum \log(\lambda_i / \Delta) \text{ for small enough (discretization) } \Delta$$

- Define the MDL-complexity as

$$MDL-COMP = \frac{1}{n} (R_{opt} + \mathcal{L}(\Lambda_{opt})) \text{ where } \Lambda_{opt} \text{ achieves } R_{opt}$$

Main result: Analytical MDL-COMP for linear models

Not just parameter count, instead a function of covariate design & its interaction with signal

If $y \sim \mathcal{N}(X\theta^*, \sigma^2 I_n)$, and $X^\top X = U \text{diag}(\rho_1, \dots, \rho_d) U^\top$, $w_i = U^\top \theta^*$, then

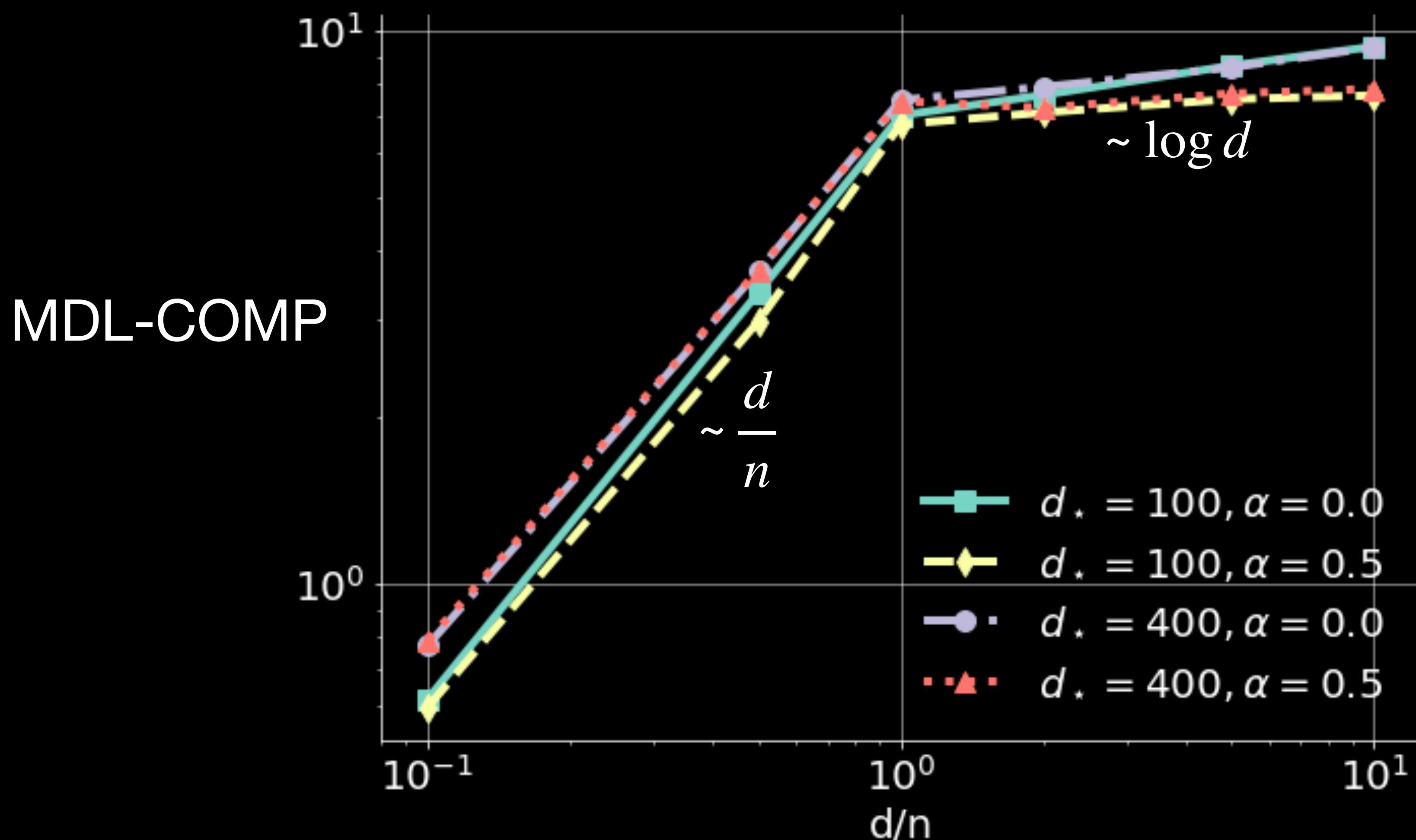
$$R_{opt} = \frac{1}{n} \sum_{i=1}^{\min\{n,d\}} \log \left(1 + \frac{\rho_i w_i^2}{\sigma^2} \right)$$

$$MDL - COMP = \frac{1}{n} \sum_{i=1}^{\min\{n,d\}} \log \left(\rho_i + \frac{\sigma^2}{w_i^2} \right) + \min \left\{ 1, \frac{d}{n} \right\} \log \left(\frac{1}{\Delta} \right)$$

Unpacking the result for Gaussian X

Slow logarithmic growth in overparameterized regime ($d \gg n$)

$$n = 200, \sigma^2 = 0.1, X \sim \mathcal{N}(0, \text{diag}(1, 2^{-\alpha}, \dots, d^{-\alpha}))$$



Other favorable properties about MDL-COMP

The optimal regularization parameter defining linear MDL-COMP also achieves

- **optimal regularization for the in-sample risk** $\min_{\Lambda} \mathbb{E}\left[\sum_{i=1}^n (x_i^\top \hat{\theta}_\Lambda - x_i^\top \theta^\star)^2\right]$
- **best worst-case redundancy over a family of noise distributions**
$$\min_{\Lambda} \max_{P \in \mathcal{P}} \mathbb{E}_{y \sim P} \log\left(\frac{1}{q_\Lambda(y)}\right)$$
 with $\mathcal{P} = \{P \mid E_P(y \mid X) = X\theta^\star, \text{Var}(y \mid X) \leq \sigma^2 I_n\}$

Also, MDL-COMP can be **easily extended to kernel methods, where it informs minimax in-sample risk**

Consequences for double descent

Perhaps the issue lies in the estimator

- Since MDL-COMP is monotone in d , using it as the complexity does not change the qualitative nature of the test MSE curves for OLS or ridge
- Double descent likely due to the choice of the estimator, e.g., min-norm OLS
- Conclusion not contradictory with literature on benign overfitting, which provide sufficient conditions for interpolation to generalize well for $d \gg n$ (**often estimator and dataset vary together with d/n**)

... Belkin-Hsu-Ma-Mandal 18, Muthukumar-Vodrahalli-Sahai 19, Hastie-Montanari-Rosset-Tibshirani 19, Tsigler-Bartlett 20, Wu-Xu 20, Nakkiran-Venkat-Kakade-Ma 20 ...

Can MDL-COMP be useful for practice?

Let's make it computable from data

$$\text{Prac-MDL-COMP} = \min_{\lambda} \log \left(\frac{1}{q_{\lambda}(y)} \right)$$

$$= \min_{\lambda} \left[\frac{\|X\hat{\theta}_{\lambda} - y\|^2}{2\sigma^2} + \frac{\lambda \|\hat{\theta}_{\lambda}\|^2}{2\sigma^2} + \boxed{\sum_{i=1}^{\min\{n,d\}} \log \left(1 + \frac{\rho_i}{\lambda} \right)} \right]$$

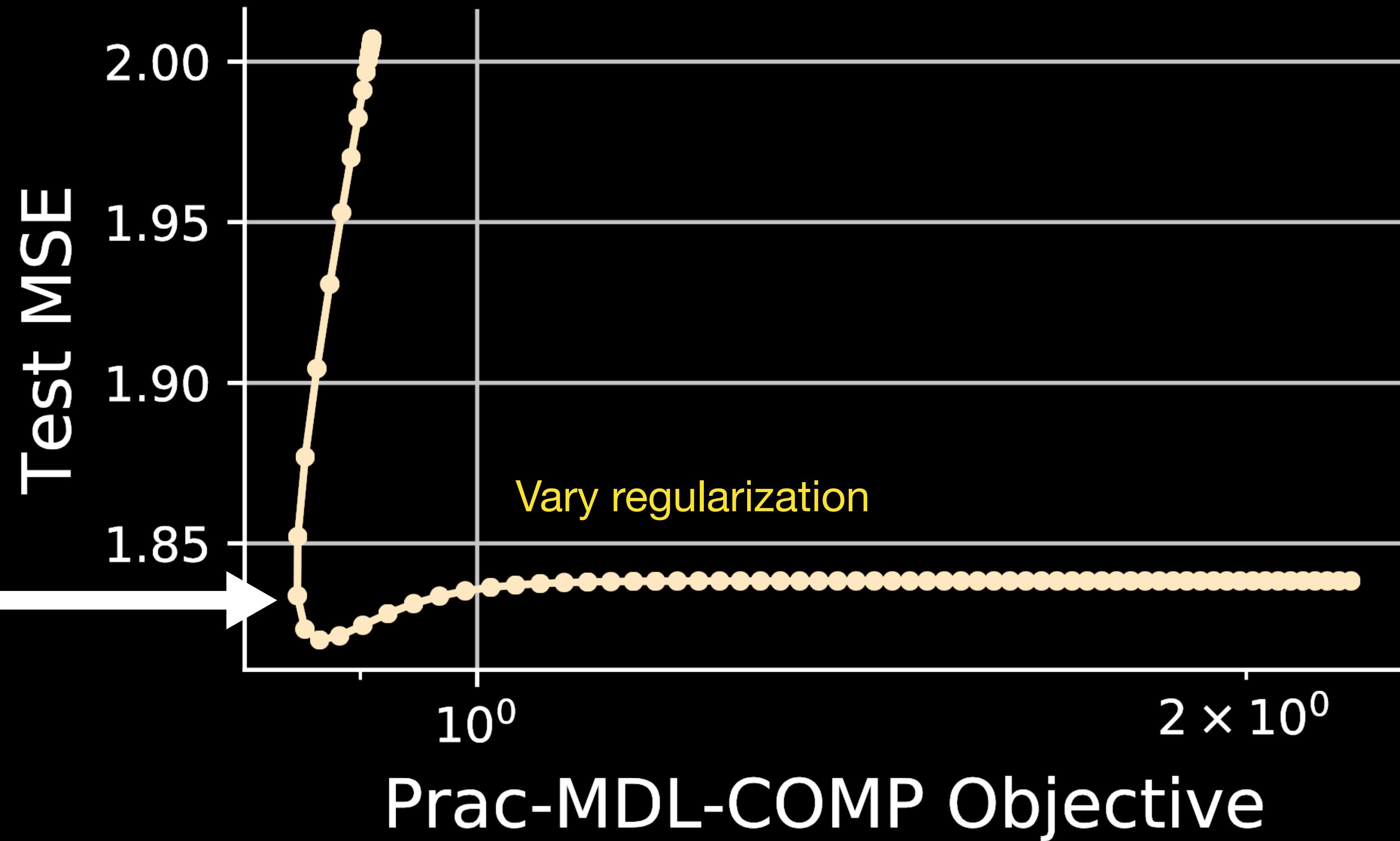
where

$$\hat{\theta}_{\lambda} = (X^T X + \lambda I)^{-1} X^T y \text{ and } \rho_i \text{ denote the eigenvalues of } X^T X$$

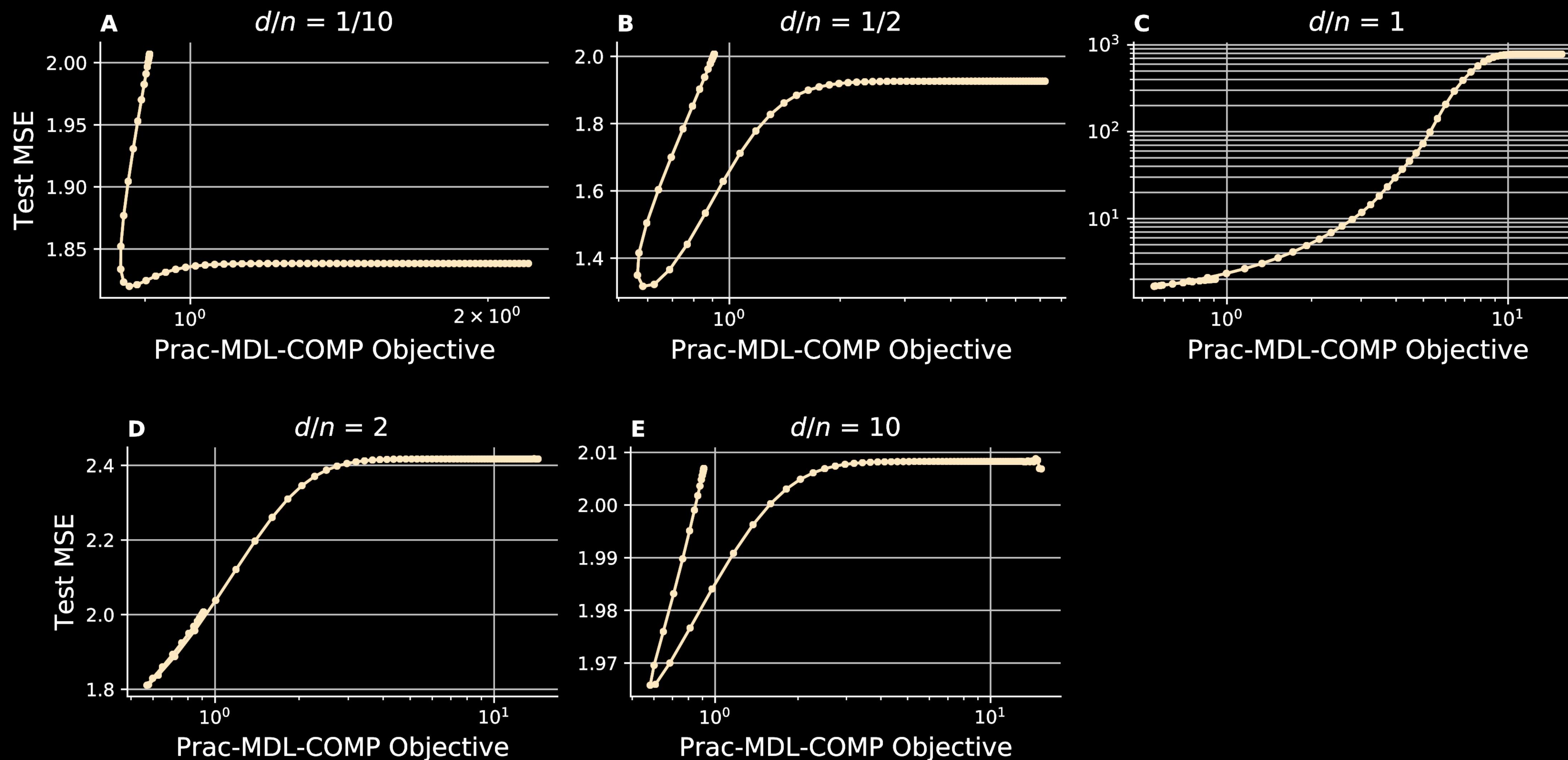
Suggested by our theorems + also consistent with
recent suggestions by Grunwald and Roos '20

Model selection with Prac-MDL-COMP

Test MSE minimizer close to the minimizer of Prac-MDL-COMP objective

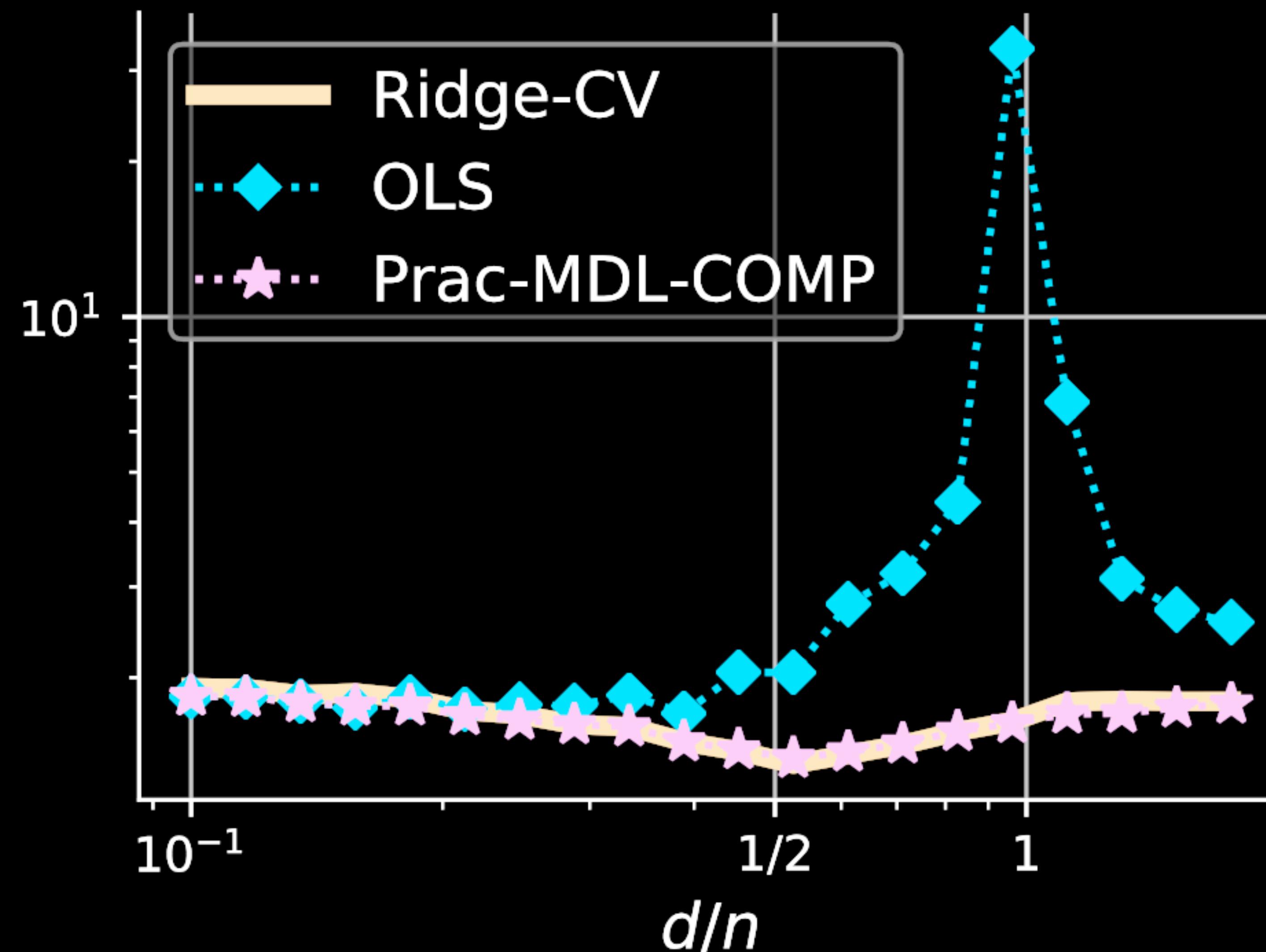


Model selection with Prac-MDL-COMP



Prac-MDL-COMP removes the double descent

Test performance competitive compared to cross-validation



Using Prac-MDL-COMP for hyperparameter tuning

$$\min_{\lambda} \left[\frac{\|X\hat{\theta}_\lambda - y\|^2}{2\sigma^2} + \frac{\lambda\|\hat{\theta}_\lambda\|^2}{2\sigma^2} + \sum_{i=1}^{\min\{n,d\}} \log\left(1 + \frac{\rho_i}{\lambda}\right) \right]$$

K-fold computational savings compared to K-fold cross validation (CV)

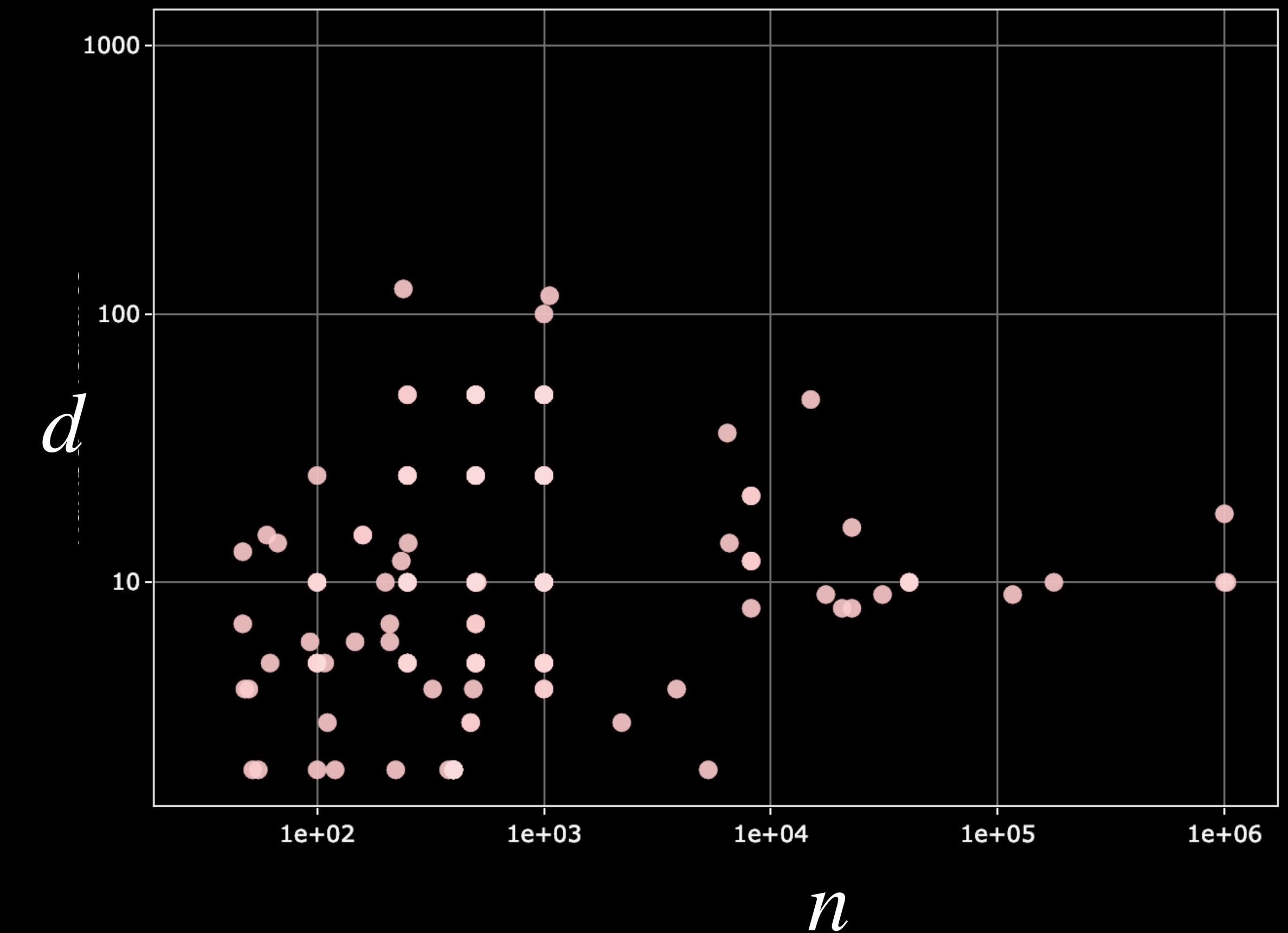
Experiments on PMLB datasets

Diverse set of tabular datasets

Predicting breast cancer from image features

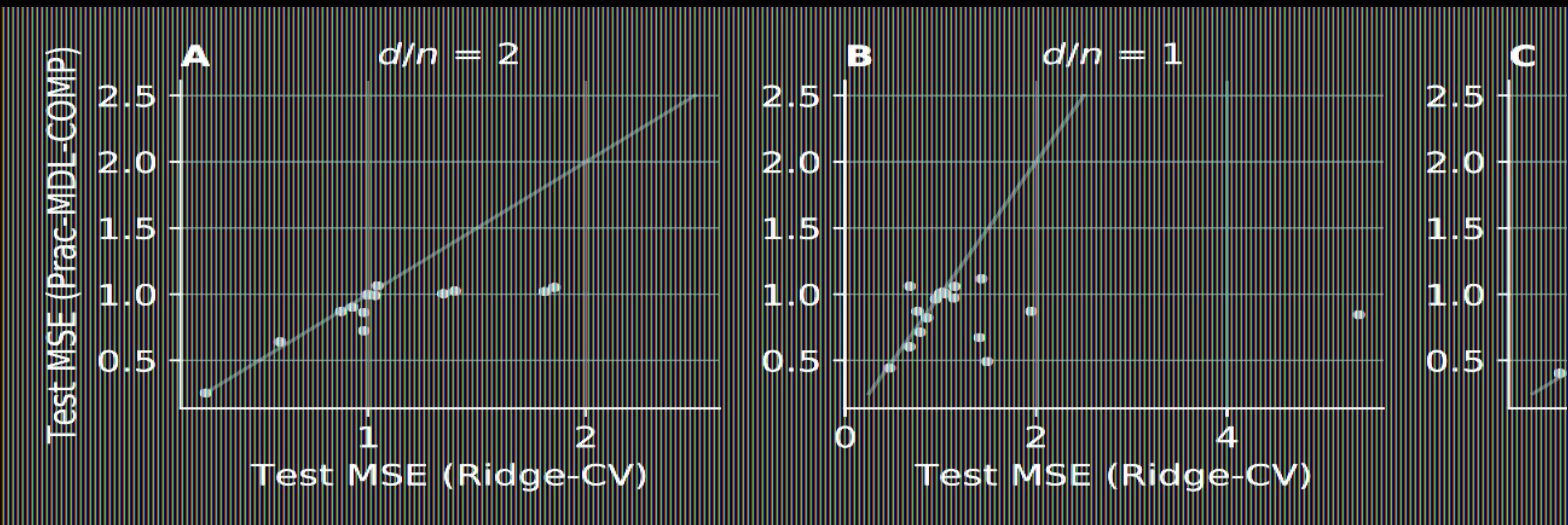
Predicting automobile prices

Election results from previous elections

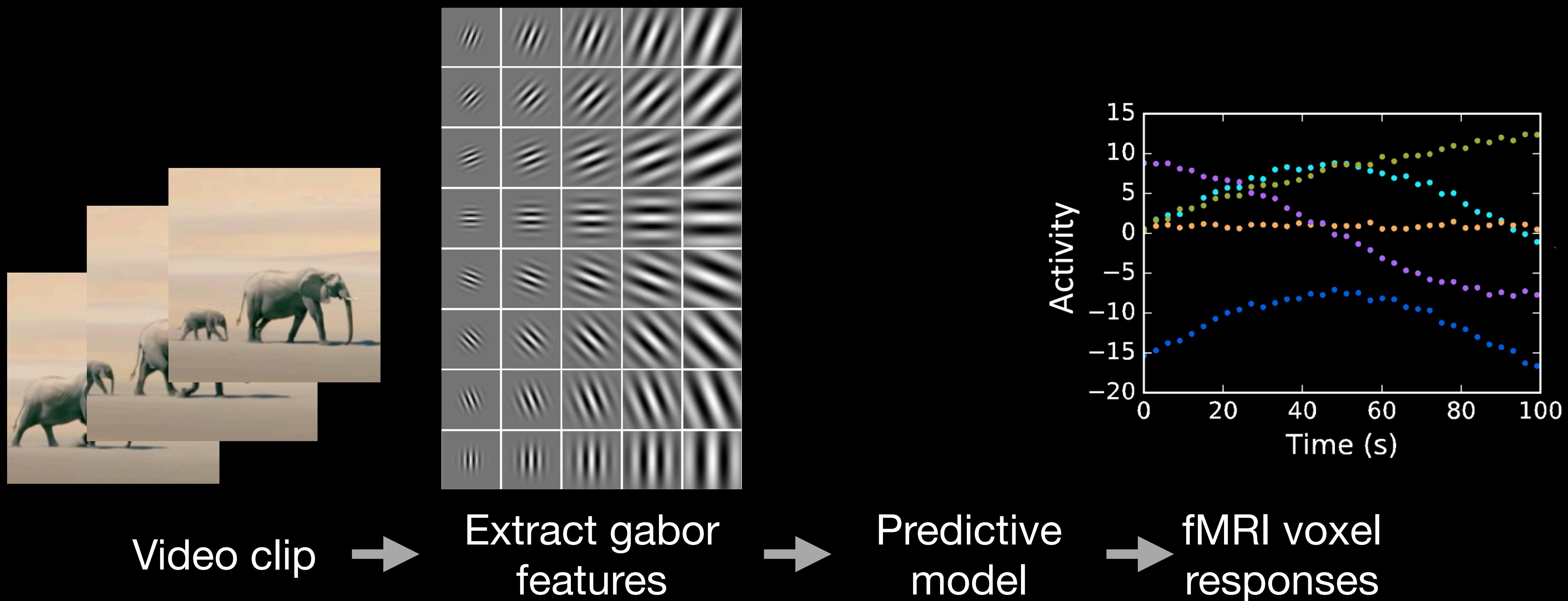


Experiments on PMLB datasets

MDL-COMP better for hyper-parameter tuning in low-data settings



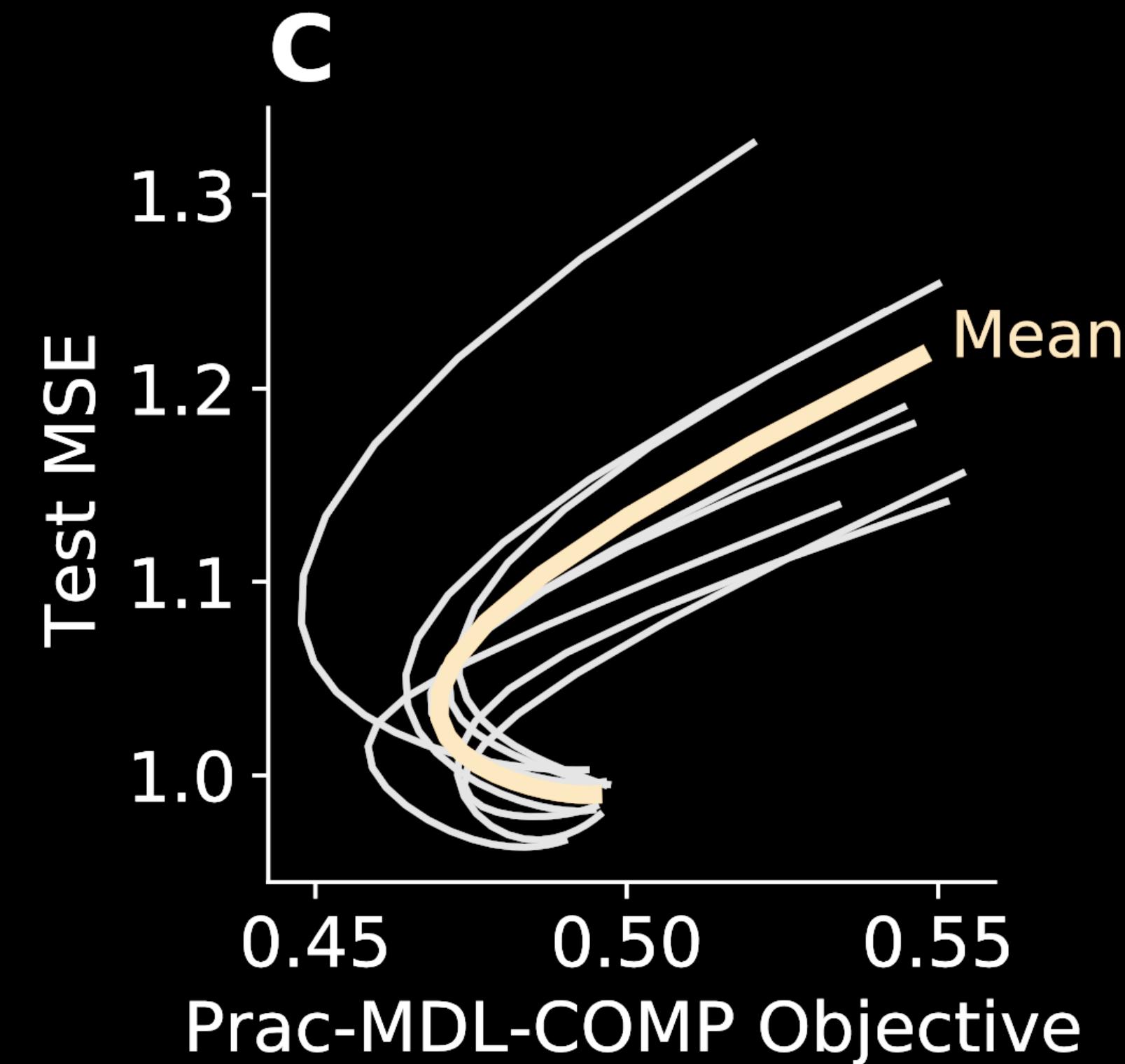
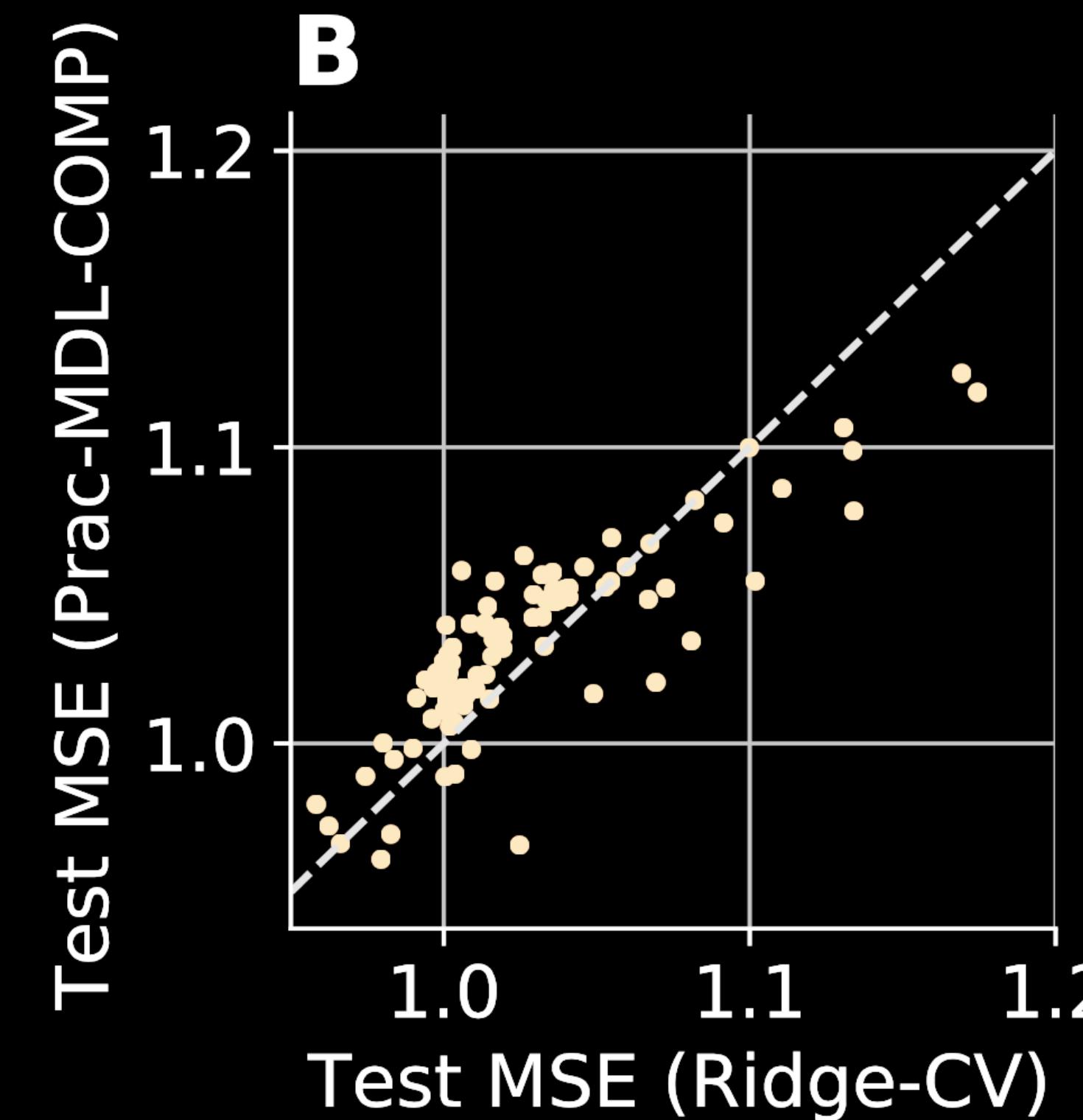
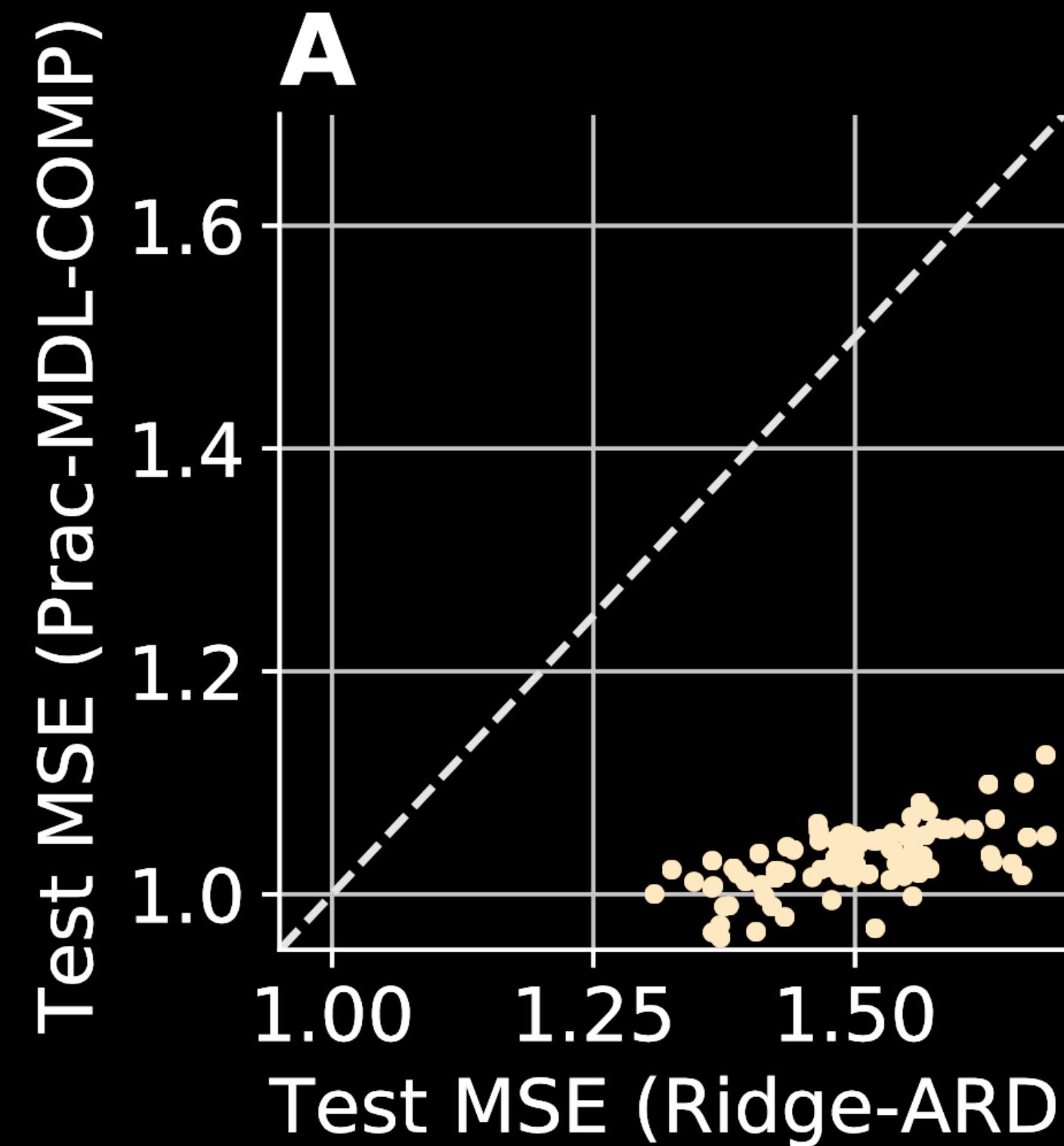
fMRI experimental setup



Nishimoto-Vu-Naselaris-Benjamini-Yu-Gallant 11

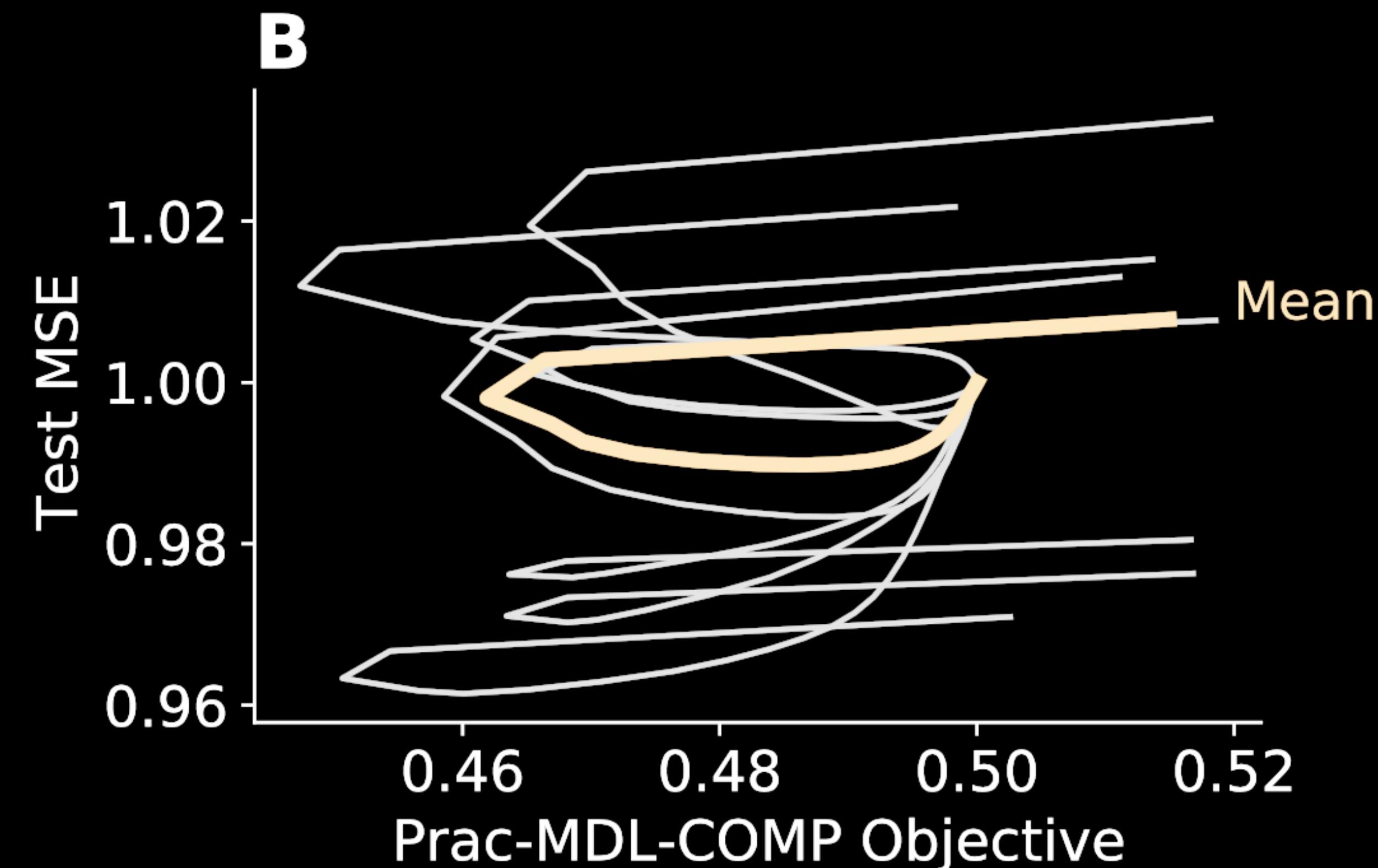
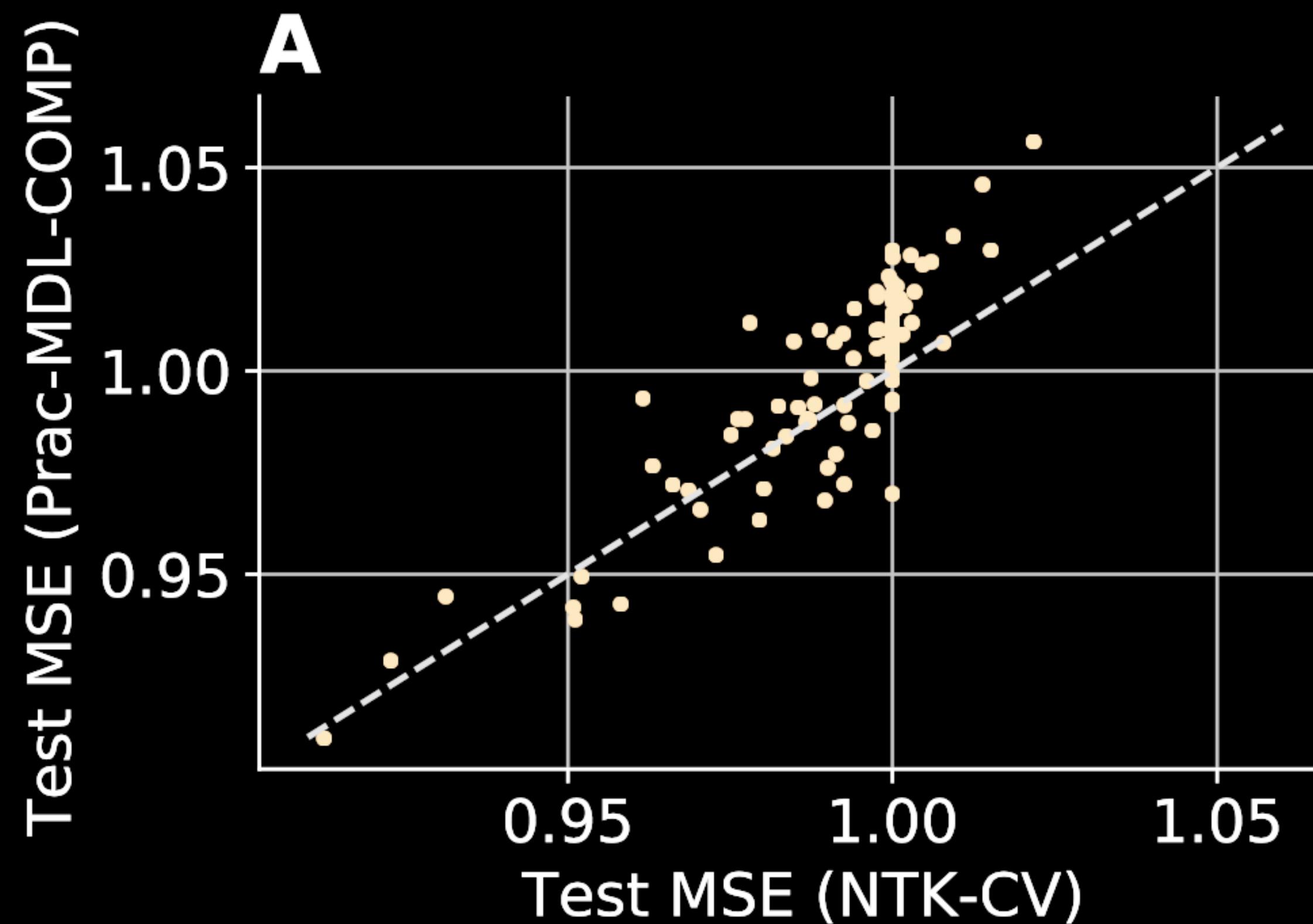
Experiments on fMRI data from 100 voxels

MDL-COMP better than Bayesian-ARD regression, and pretty comparable to CV tuning



Experiments on neural tangent kernels

(2 layer RELU neural tangent kernel)



Once again, MDL-COMP pretty comparable to CV tuning

Summary

MDL-COMP—**a modified NML complexity measure using optimal ridge estimators**

- Not just parameter count— $\log d$ scaling for $d \gg n$ with Gaussian covariates
- Hints that double descent likely due to choice of estimator
- Provides competitive-to-CV but computationally-better hyper-parameter tuning
- Open questions:

Analytical relations between MDL-COMP and out-of-sample generalization?

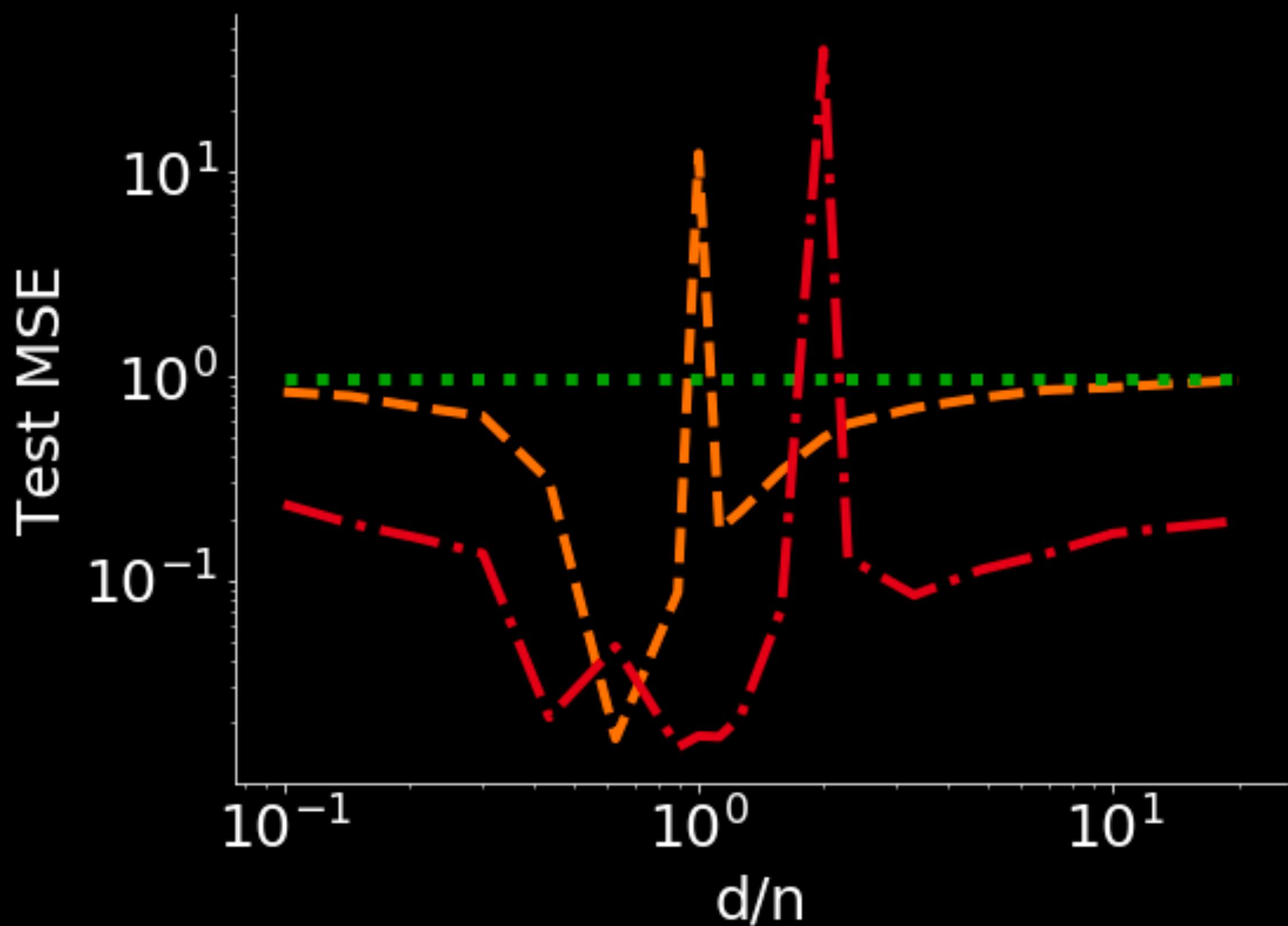
MDL-COMP for classification + complex models like neural networks?



Additional slides

Bias-variance tradeoff: Few things to note..

- We should expect a tradeoff *given*
 - some fixed data
 - as the “complexity” of the fitted estimator changes
- Do not expect a tradeoff for
 - poor choice of estimators
 - poor choice of complexity



MDL-COMP for kernel methods

Universal codes induced by kernel ridge regression

- Define the code Q_λ :

$$q_\lambda(y) \propto \exp\left(-\frac{1}{2\sigma^2} \|K\hat{\theta}_\lambda - y\|^2 - \frac{\lambda}{2\sigma^2} \hat{\theta}_\lambda^\top K \hat{\theta}_\lambda\right)$$

where

$$\hat{\theta}_\lambda = \min_{\theta} \|K\theta - y\|^2 + \lambda \theta^\top K \theta = (K + \lambda I)^{-1} y$$

- This choice comes from kernel ridge regression:

$$\min_{f \in \mathcal{H}} \sum_{i=1}^n (f(x_i) - y_i)^2 + \lambda \|f\|_{\mathcal{H}}^2$$

Kernel ridge regression

- One can show that for the optimization problem

$$\min_{f \in \mathcal{H}} \sum_{i=1}^n (f(x_i) - y_i)^2 + \lambda \|f\|_{\mathcal{H}}^2,$$

it suffices to consider the functions of the form

$$f = \sum_{i=1}^n \theta_i K(x_i, \cdot),$$

and this leads to the kernel ridge regression problem in the previous slide

MDL-COMP for kernel regression

- Let ρ_i denote the eigenvalues of the kernel matrix $(K(x_i, x_j))_{i,j=1}^n$ and suppose $y \sim \mathcal{N}(f^\star(X), \sigma^2 I_n)$ for some f^\star in RKHS of K , then

$$\mathcal{R}_{opt} \leq \frac{1}{2n} \left[\min_{\lambda} \frac{\lambda \|f^\star\|_{\mathcal{H}}^2}{\sigma^2} + \sum_{i=1}^n \log \left(1 + \frac{\rho_i}{\lambda} \right) \right]$$

(no easy closed-form)

- Since there is only a single hyper-parameter, we can directly take

$$MDL - COMP = \mathcal{R}_{opt}$$

Unpacking MDL-COMP for Sobolev kernels

- For Sobolev kernel of smoothness α , the eigenvalues decay like $\rho_i \sim i^{-2\alpha}$, and one can derive

$$\mathcal{R}_{opt} \leq C \left(\frac{\|f^\star\|_{\mathcal{H}}^2}{\sigma^2} \right)^{\frac{1}{2\alpha+1}} \cdot n^{-\frac{2\alpha}{2\alpha+1}}$$

Neural tangent kernels (NTK)

NTK approximates neural net with infinite width

Jacot et al. 2018

- Varies with number of layers and nonlinearity
 - $K(x, x') = \mathbb{E}_{\theta \sim W} \left[\left\langle \frac{\partial f(\theta, x)}{\partial \theta}, \frac{\partial f(\theta, x')}{\partial \theta} \right\rangle \right]$
- Analytical expressions for simple architectures (e.g., cosine kernel for 2 layer Relu networks)
- Software libraries for computing the kernel for deeper networks

Kernel version of the computation

$$\begin{aligned}\text{Prac-MDL-COMP} &= \min_{\lambda} \log \left(\frac{1}{q_{\lambda}(y)} \right) \\ &= \min_{\lambda} \left[\frac{\|K\hat{\theta}_{\lambda} - y\|^2}{2\sigma^2} + \frac{\lambda \hat{\theta}_{\lambda}^T K \hat{\theta}_{\lambda}}{2\sigma^2} + \sum_{i=1}^n \log \left(1 + \frac{\rho_i}{\lambda} \right) \right]\end{aligned}$$

where

$$\hat{\theta}_{\lambda} = (K + \lambda I)^{-1} y \quad \text{and} \quad \rho_i \text{ denote the eigenvalues of the kernel matrix } K$$

Proofs

Proof sketch for linear models

$$\begin{aligned}
\mathcal{D}_{\text{KL}}(\mathbb{P}_{\theta_\star} \parallel \mathbb{Q}_\Lambda) &= \mathbb{E}_\mathbf{y} \left[\log \frac{p(\mathbf{y}; \mathbf{X}, \theta_\star)}{q_\Lambda(\mathbf{y})} \right] \\
&= \mathbb{E}_\mathbf{y} \left[\log \left(\frac{\frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{X}\theta_\star\|^2\right)}{\frac{1}{C_\Lambda(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{X}\hat{\theta}\|^2 - \frac{1}{2\sigma^2} \hat{\theta}^\top \Lambda \hat{\theta}\right)} \right) \right] \\
(31) \quad &= -\underbrace{\mathbb{E}_\mathbf{y} \left[\frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{X}\theta_\star\|^2 \right]}_{=:T_1} + \underbrace{\mathbb{E} \left[\frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{X}\hat{\theta}\|^2 + \frac{1}{2\sigma^2} \hat{\theta}^\top \Lambda \hat{\theta} \right]}_{=:T_2} + \underbrace{\log C_\Lambda}_{=:T_3}.
\end{aligned}$$

$$(33a) \quad T_2 = \frac{(n - \min\{n, d\})}{2} + \frac{1}{2} \sum_{i=1}^{\min\{n, d\}} \frac{(\rho_i w_i^2 / \sigma^2 + 1) \lambda_i}{\lambda_i + \rho_i}, \quad \text{and}$$

$$(33b) \quad T_3 = \frac{1}{2} \sum_{i=1}^{\min\{n, d\}} \log \left(\frac{\rho_i + \lambda_i}{\lambda_i} \right)$$

$$\begin{aligned}
\frac{1}{n} \mathcal{D}_{\text{KL}}(\mathbb{P}_{\theta_*} \| \mathbb{Q}_{\Lambda}) &= T_1 + T_2 + T_3 \\
(34) \quad &= -\frac{\min\{n, d\}}{2n} + \frac{1}{2n} \sum_{i=1}^{\min\{n, d\}} \underbrace{\left(\frac{(\rho_i w_i^2 / \sigma^2 + 1) \lambda_i}{\lambda_i + \rho_i} + \log \left(\frac{\rho_i + \lambda_i}{\lambda_i} \right) \right)}_{=: f_i(\lambda_i)}.
\end{aligned}$$

Finally to compute the \mathcal{R}_{opt} (32), we need to minimize the KL-divergence (34) where we note the objective depends merely on $\lambda_1, \dots, \lambda_{\min\{n, d\}}$. We note that the objective (RHS of equation (34)) is separable in each term λ_i . We have

$$(35) \quad f'_i(\lambda_i) = 0 \iff -\frac{(\rho_i w_i^2 / \sigma^2 + 1)}{(1 + \rho_i / \lambda_i)^2} + \frac{1}{1 + \rho_i / \lambda_i} = 0 \iff \lambda_i^{\text{opt}} = \frac{\sigma^2}{w_i^2}.$$

Proof sketch for the result with Gaussian X

- When $X \in \mathbb{R}^{n \times d}$ has i.i.d. $\mathcal{N}(0, 1/n)$ entries, then for $X^\top X = U \text{diag}(\rho_1, \dots, \rho_d) U^\top$
 - The matrix U has uniform distribution over the set of $d \times d$ orthonormal matrices and hence for any fixed θ^* , the coordinates of $w = U^\top \theta^*$ are identically distributed, and we can use the approximation $w_i^2 \approx \frac{\|\theta^*\|^2}{d}$

Proof sketch for the result with Gaussian X

- When $X \in \mathbb{R}^{n \times d}$ has i.i.d. $\mathcal{N}(0, 1/n)$ entries, then for $X^\top X = U \text{diag}(\rho_1, \dots, \rho_d) U^\top$
 - The eigenvalues ρ_i follow Marcenko-Pastur Law with the following approximation
 - $d \ll n, \quad X^\top X \approx I_d, \quad \rho_i \approx 1$
 - $d > n, \quad X^\top X \approx \begin{bmatrix} \frac{d}{n} I_n & 0 \\ 0 & 0 \end{bmatrix}, \quad \rho_i \begin{cases} \approx \frac{d}{n}, & i \leq n \\ = 0, & i > n \end{cases}$

Two-stage MDL

Two-stage MDL

- Consider a parametric class of codes $\{p_\theta, \theta \in \Theta\}$, and then use the valid codelength for any fixed p_θ

$$\log \left(\frac{1}{p_\theta(y)} \right)$$

- Minimizing this codelength is same as MLE over the given parametric class
- But the choice of $\hat{\theta}$ varies with y , so need to account for the codelength needed to transmit the value of $\hat{\theta}$

Two-stage MDL

- Thus the overall codelength is

$$\log \left(\frac{1}{p_{\hat{\theta}}(y)} \right) + \frac{d}{2} \log n$$

Code length for data

Code length for d -dimensional
parameter upto $1/\sqrt{n}$ resolution

- For a fixed parametric class, same as MLE (since the second term is constant)
- For a family of parametric classes, same as BIC procedure (model selection)

MDL-COMP vs Cross-validation

- For $n \times d$ covariates, for each value of λ , the computational costs are
 - **K-fold cross-validation:** $K \times \text{OLS solver} = K \times (nd^2 + \min(n^3, d^3))$
 - **Prac-MDL-COMP:** $1 \times \text{SVD solver} = nd^2 + n^2d$

Prac-MDL-COMP provides a proxy for complexity and saves K-fold computation!

Issues with NML

Issues with NML: Linear model

- Then Q_{NML} is given by

$$q_{NML}(y) \propto \max_{\theta} p_{\theta}(y) = p_{\hat{\theta}}(y) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2} \|X\hat{\theta} - y\|^2\right)$$

$$\hat{\theta} = \arg \max_{\theta} p_{\theta}(y) = \arg \min_{\theta} \|X\theta - y\|^2 = \hat{\theta}_{OLS}$$

(We can use min-norm OLS when $d > n$)

Issues with NML: Linear model

- If \mathcal{Y} is not compact (even when $d < n$)

$$\int \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2} \|X\hat{\theta} - y\|^2\right) dy = \infty$$

- Easiest to see when $d > n$ so that $X\hat{\theta} = y$, and we have

$$\int \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2} \|X\hat{\theta} - y\|^2\right) dy = \int_{\mathbb{R}^n} \frac{1}{(2\pi\sigma^2)^{n/2}} dy = \infty$$

Optimal code: With **known** true model P^* is P^*

- When $y \sim P^*$, the expected code-length when using code Q is given by

$$\mathbb{E}_{y \sim P^*} \log \left(\frac{1}{Q(y)} \right) = \text{KL}(P^* || Q) + H(P^*)$$

- Minimized when $Q = P^*$, since redundancy is non-negative
- P^* also minimizes the worst-case regret

$$p^* = \arg \min_q \max_y \left[\log \left(\frac{1}{q(y)} \right) - \log \left(\frac{1}{p^*(y)} \right) \right] \text{ such that } \int q(z) dz \leq 1$$

Expressions

Unpacking the result for Gaussian X

Slow logarithmic growth in overparameterized regime ($d \gg n$)

- When $X \in \mathbb{R}^{n \times d}$ has i.i.d. $\mathcal{N}(0, 1/n)$ entries, then

$$\text{MDL-COMP} \approx \begin{cases} \frac{d}{n} \log \left(1 + \frac{d_\star}{r^2} \right) + \frac{d}{n} \log \left(\frac{1}{\Delta} \right), & \text{if } d \in [1, d_\star] \\ \frac{d}{n} \log \left(1 + \frac{d}{r^2} \right) + \frac{d}{n} \log \left(\frac{1}{\Delta} \right), & \text{if } d \in [d_\star, n] \\ \log \left(\frac{d}{n} + \frac{d}{r^2} \right) + \log \left(\frac{1}{\Delta} \right), & \text{if } d \in [n, \infty) \end{cases}$$

[here d_\star denotes the true dimensionality of θ^\star , and we assume $\mathbb{E}[y | X] = \tilde{X}\theta^\star$ where \tilde{X} denotes the first d_\star columns of X ; and $r^2 = \|\theta^\star\|^2$]

MDL-COMP scaling for other designs

Numerical computation: Another example

$$n = 200, \sigma^2 = 0.1, X \sim \mathcal{N}(0, \begin{bmatrix} 16\mathbf{I}_s & 0 \\ 0 & \mathbf{I}_{d-s} \end{bmatrix})$$

