Machine Learning for Economists: Part 1 – Learning

Michal Andrle
International Monetary Fund

Washington, D.C.,
October, 2018

Disclaimer #1:

The views expressed herein are those of the authors and should not be attributed to the International Monetary Fund, its Executive Board, or its management.

Learning $\neq$ Fitting

Learning: Choose a model with sufficient **representational capacity** that will perform well out-of-sample.

For given data, increasing model's capacity increases **over-fitting**.

Overfitting:

- Fits available data, does not **generalize well** to new data from the population (nature's) distribution
- Small with **training sample** error, large **test sample** error
- Occurs when model complexity too large for **available** data sample (low 'degrees of freedom')

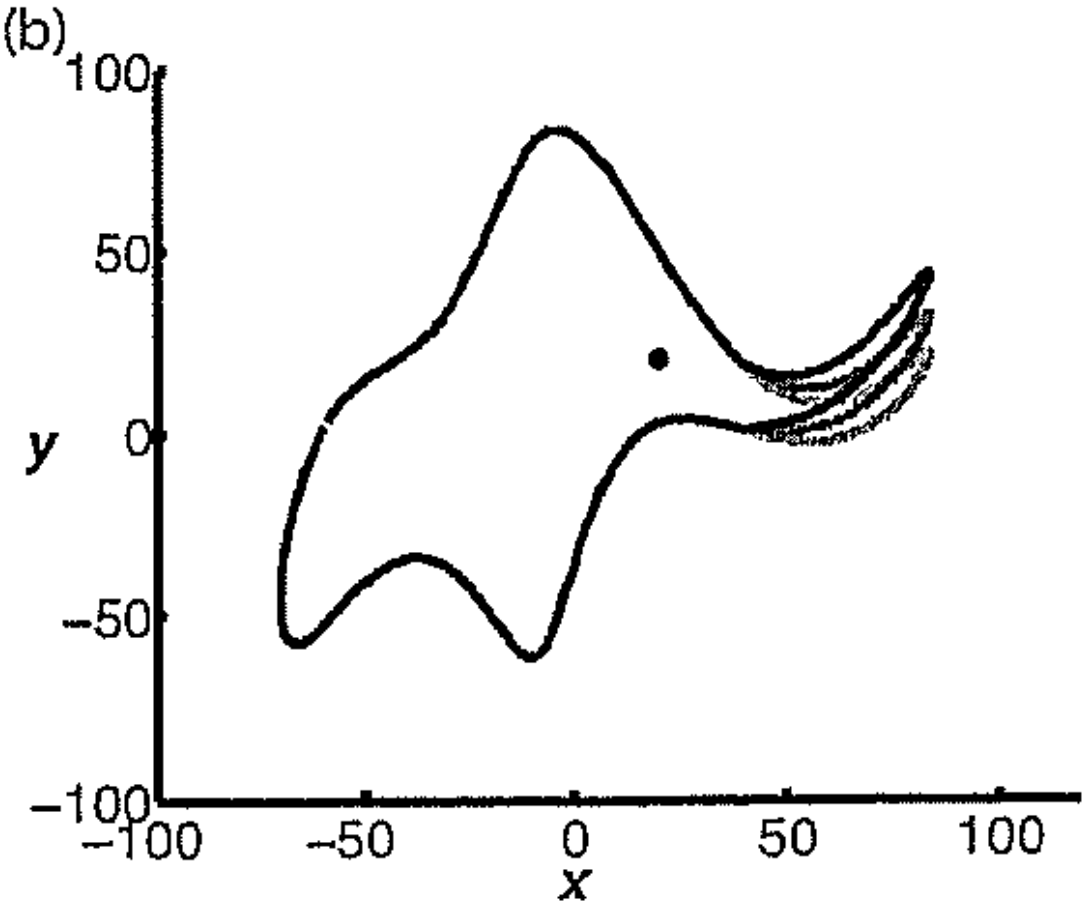
Avoiding overfitting is one of top issues in all types of machine learning and deep learning.

Overfitting

**“With four parameters I can fit an elephant,
and with five I can make him wiggle his trunk.”**

John von Neumann [quoted by Enrico Fermi]

Von Neuman-Mayer Elephant



Mayer et al., Am. J. Phys. 78(6), June 2010

Capacity Selection: Extreme Curve Fitting

For $\{x_1, \ldots, x_n\}$ and $\{y_1, \ldots, y_n\}$, choose a model

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_M x_i^M = f(x, \beta)$$

to minimize

$$err = \sum_i^N \{y_i - f(x, \beta)^2\}.$$

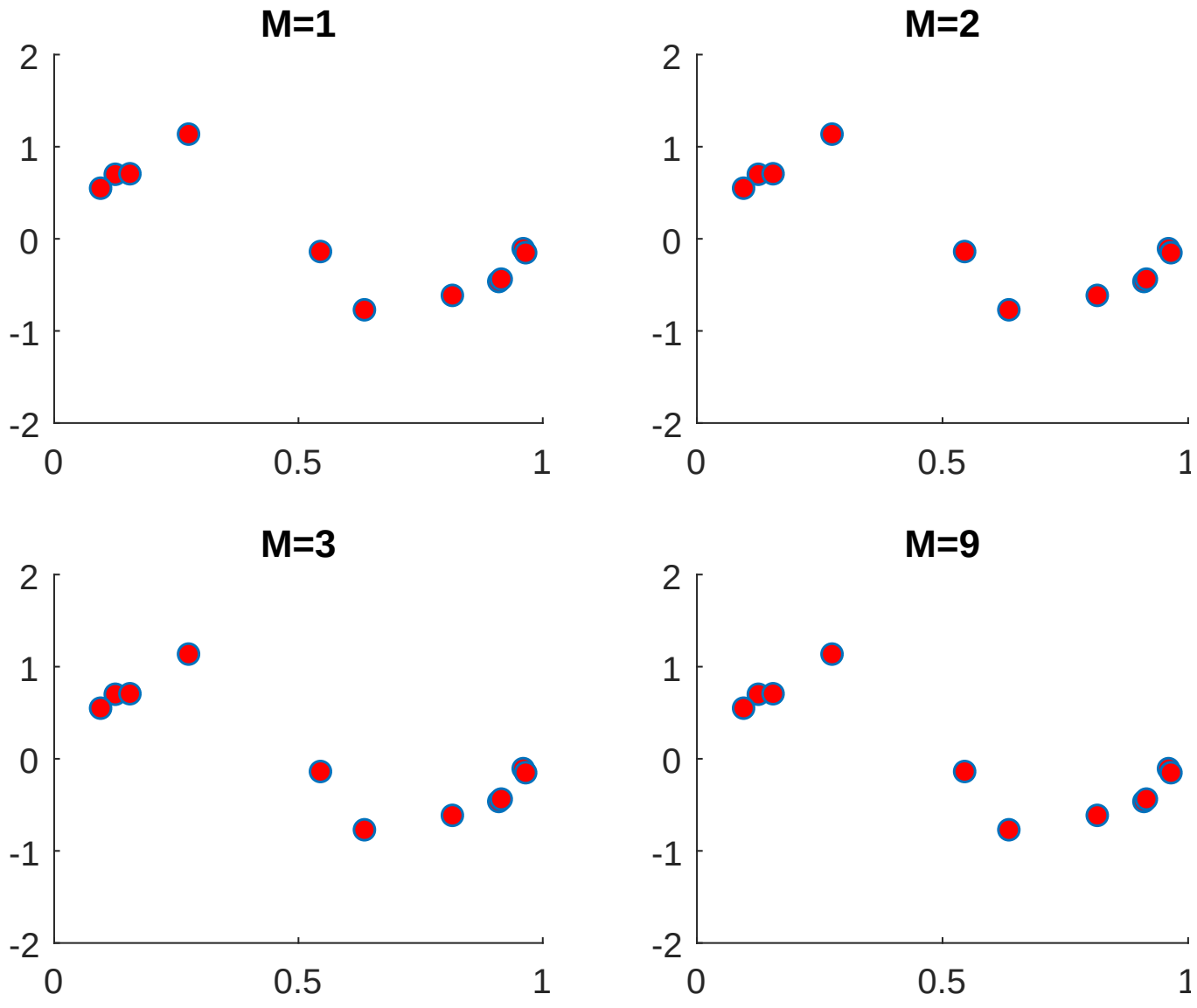
However:
Any n points can be **fit exactly** by $n - 1$ degree polynomial!

In-sample Fit: Perfect

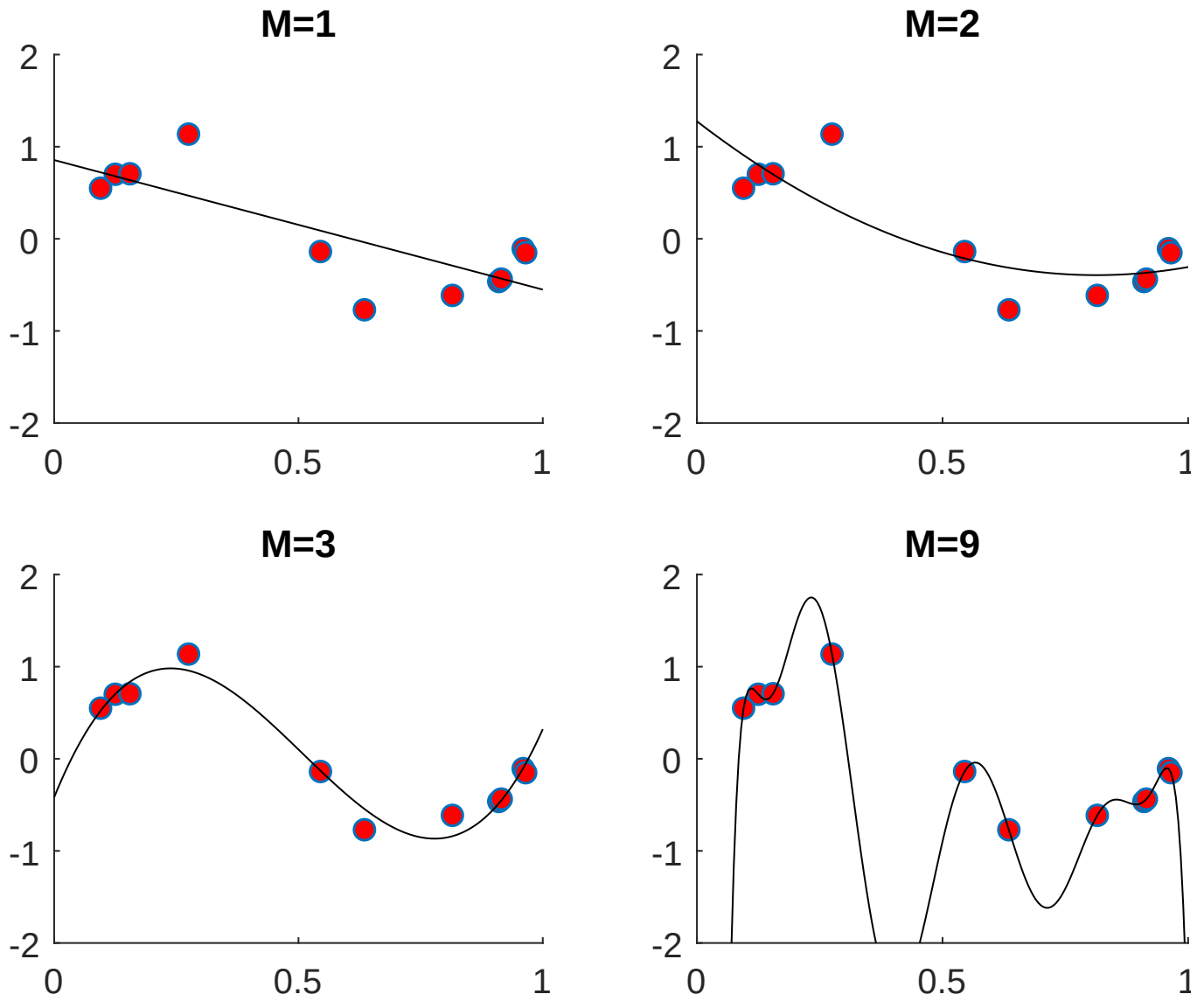
Out-of-Sample Fit: Disaster!

Note: These are not orthogonal polynomials.

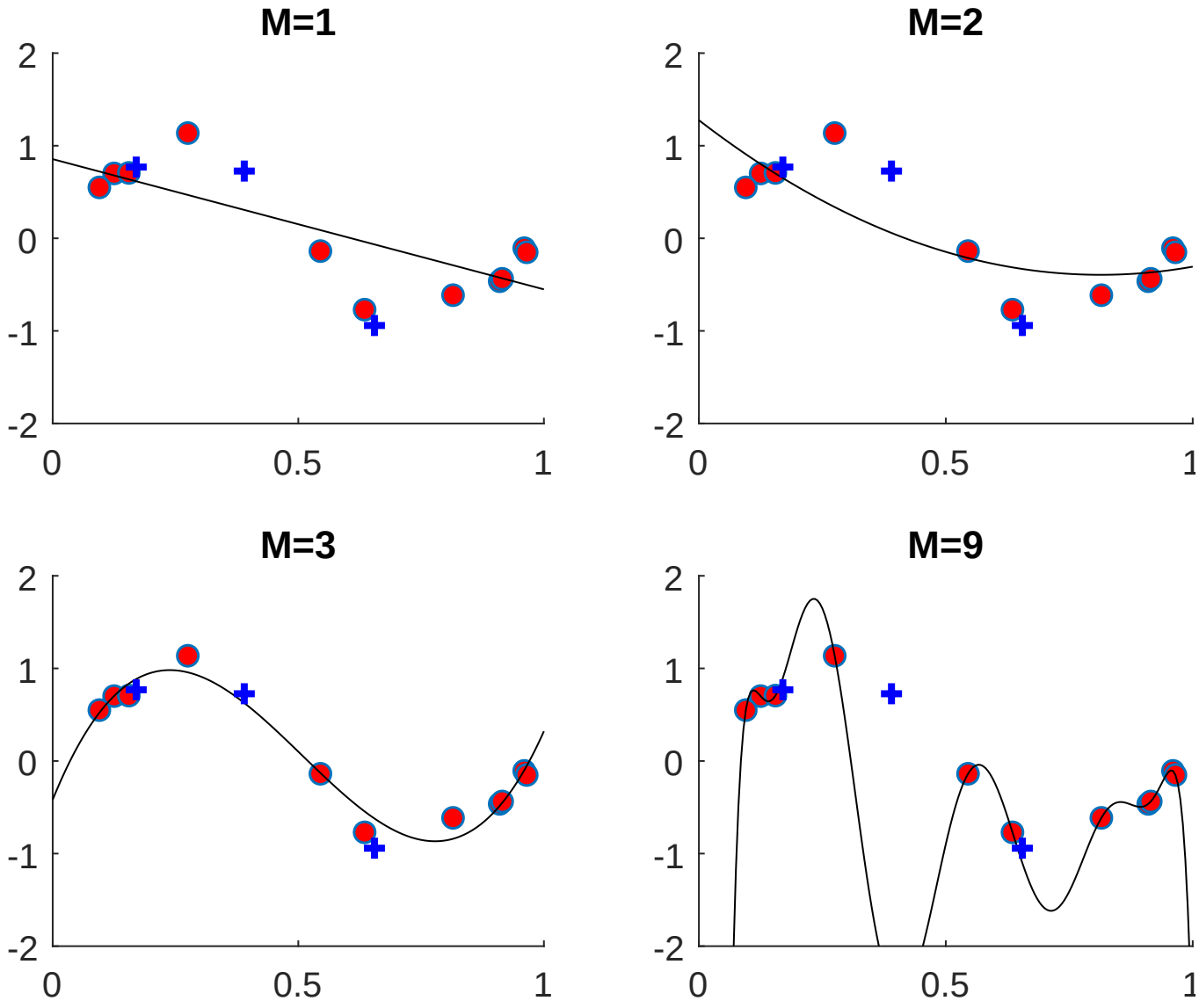
Capacity Selection: Over-fitting



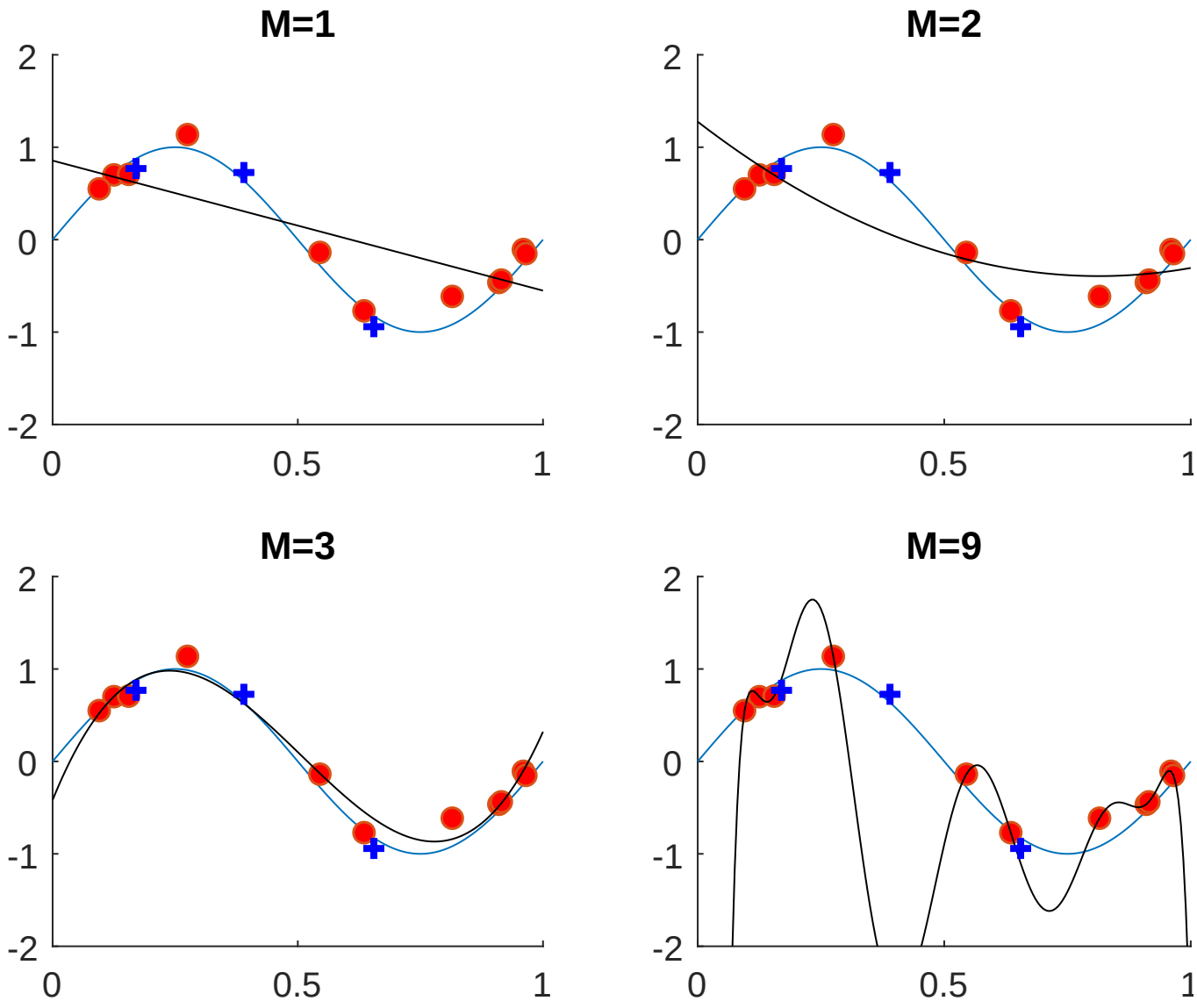
Capacity Selection: Over-fitting



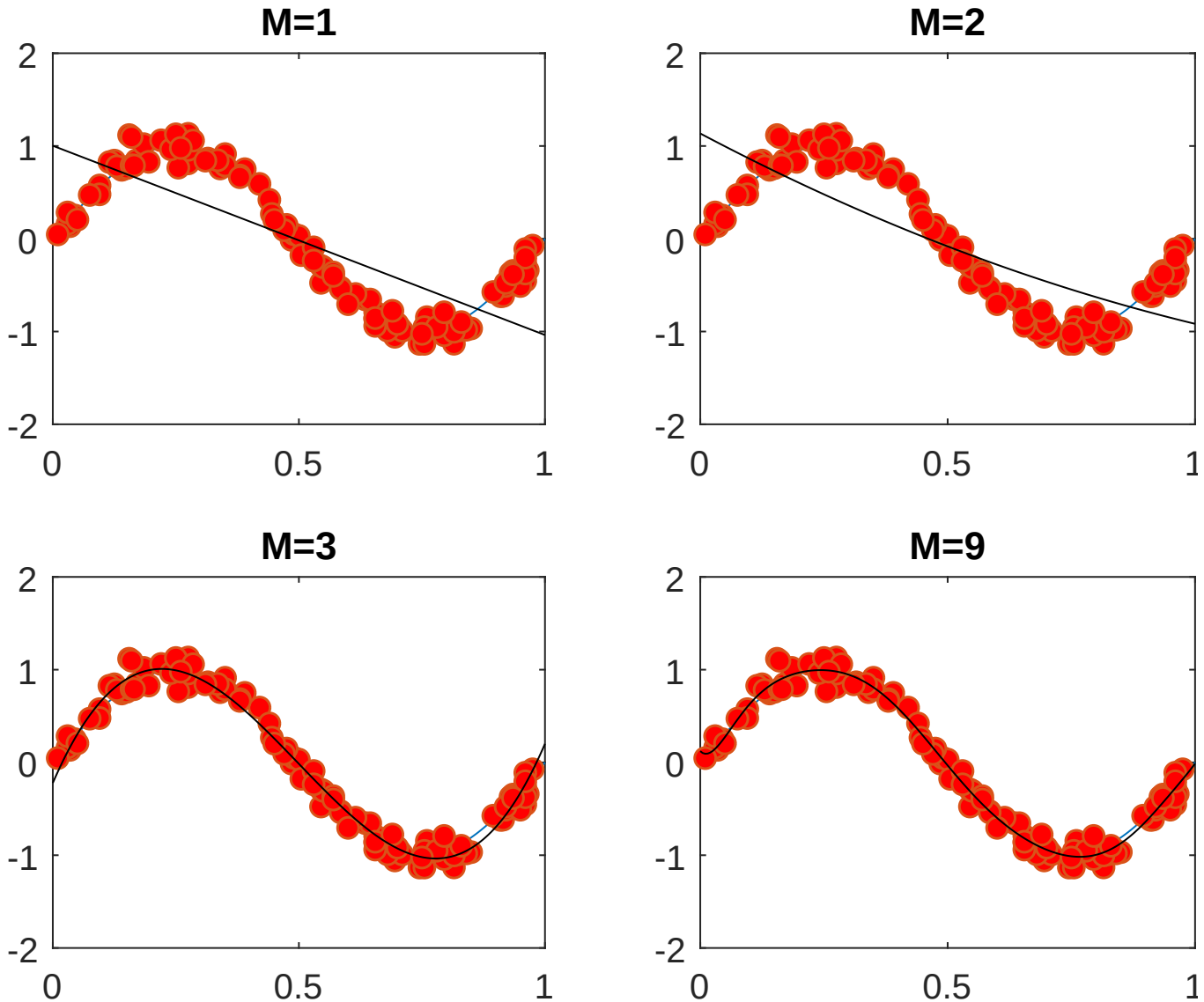
Capacity Selection: Over-fitting



Capacity Selection: Over-fitting

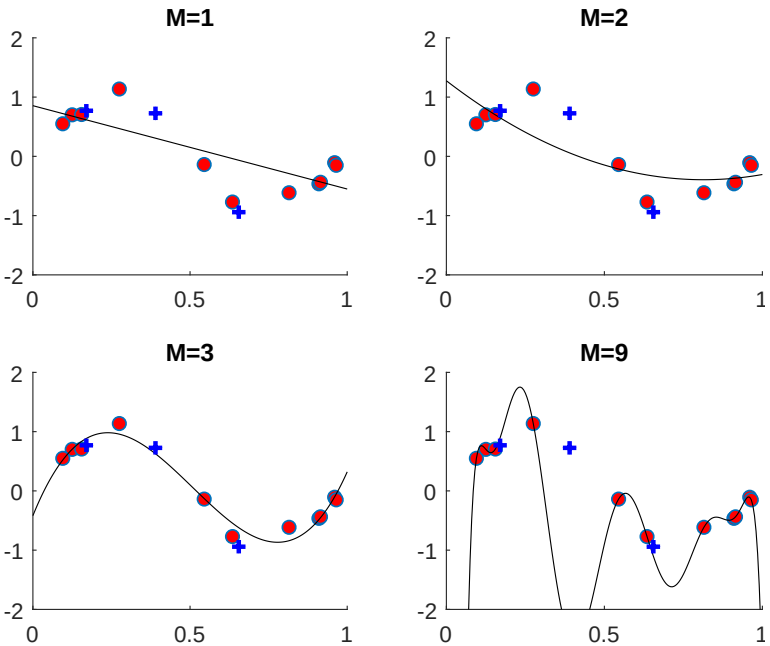


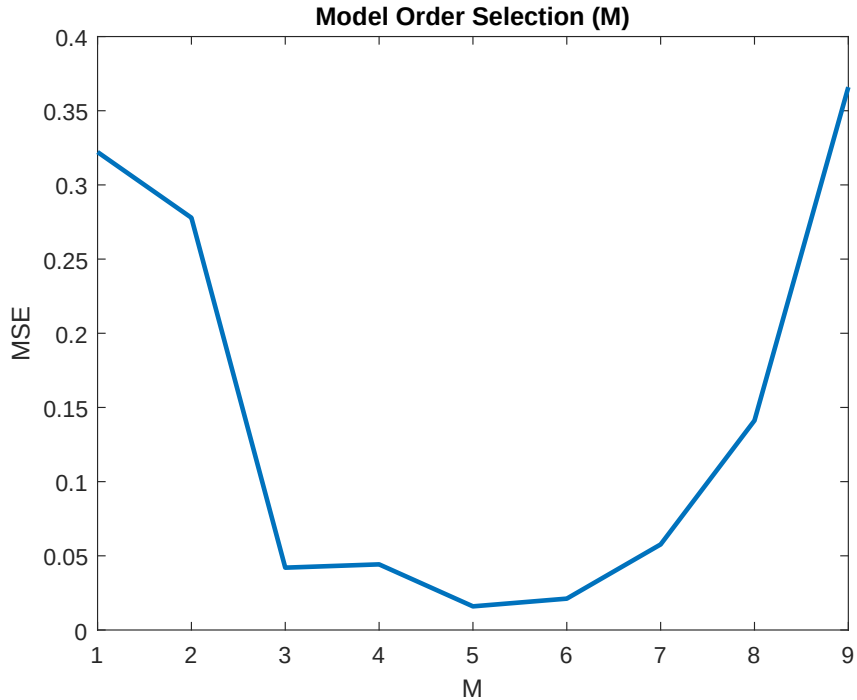
Capacity Selection: Getting More Observations...



Model Order Selection ("Structural Stabilization")

Estimate using **training sample**,**o**, and choose the best model on a **test sample**,**+**

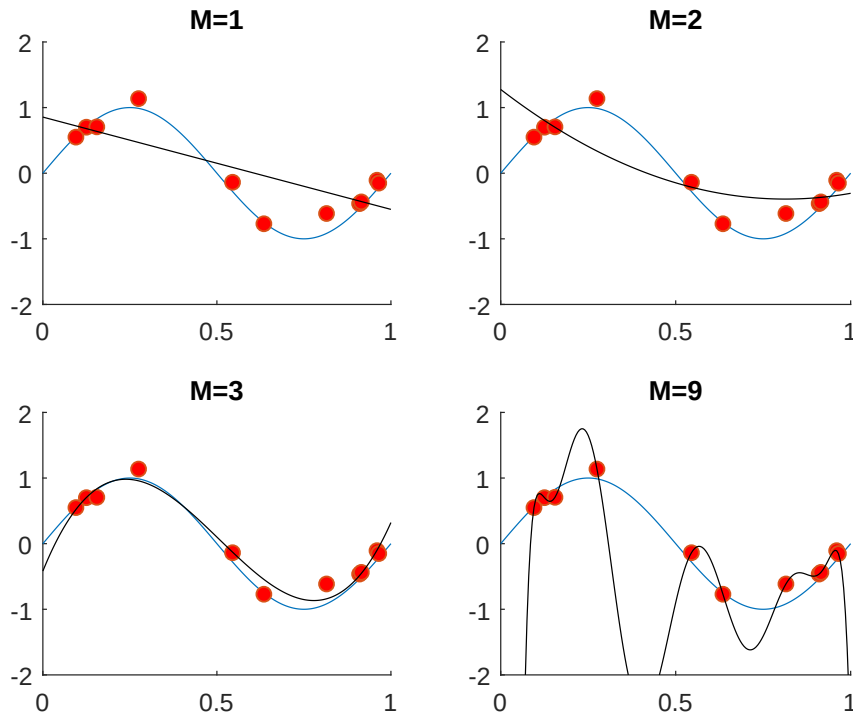




Capacity Selection: Over-fitting

Very unsetling. . .

Coefficients	M = 1	M = 4	M = 9
β_0	0.746	-0.112	0.00048
β_1	-1.57	11.139	6.276
β_2		-32.621	0.157
β_3		21.635	-43.086
β_4			10.871
β_5			39.874
β_6			102.809
β_7			-239.546
β_8			157.725
β_9			-35.089



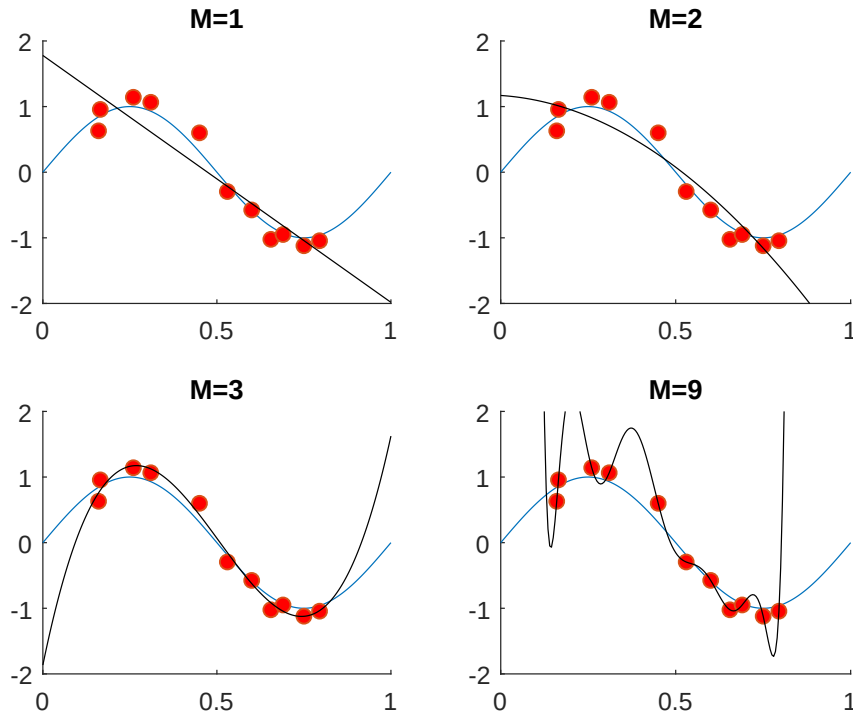
+ In-sample fit is great
– Really bad generalization
– Unstable coefficients
– Predictors highly correlated

What's happening?
Try to **regularize** the problem!

Capacity Selection: Over-fitting

Very unsetling. . .

Coefficients	M = 1	M = 4	M = 9
β_0	0.746	-0.112	0.00048
β_1	-1.57	11.139	6.276
β_2		-32.621	0.157
β_3		21.635	-43.086
β_4			10.871
β_5			39.874
β_6			102.809
β_7			-239.546
β_8			157.725
β_9			-35.089



+ In-sample fit is great
– Really bad generalization
– Unstable coefficients
– Predictors highly correlated

What's the solution?
Try to **regularize** the problem!

Regularization

Regularization:

"Any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error."

Goodfellow et al. 2017 (pp. 117, Deep Learning)

Regularization embeds a set of **priors** for solutions into the learning process.

As $N \to \infty$ the model puts less and less weight on the prior restrictions, adapting to data, and increasing effective capacity

Regularization (1)

Regularization:
Introducing **prior** constraints to solve ill-conditioned problems.

Let's try to penalize the size of the regression coefficients to prevent overfitting...

New problem to minimize:

$$err = \sum_{i}^{N} \{y_i - f(x, \beta)\}^2 + \lambda \left\{ \sum_{j=1}^{M} (\beta_j - 0)^2 \right\}.$$

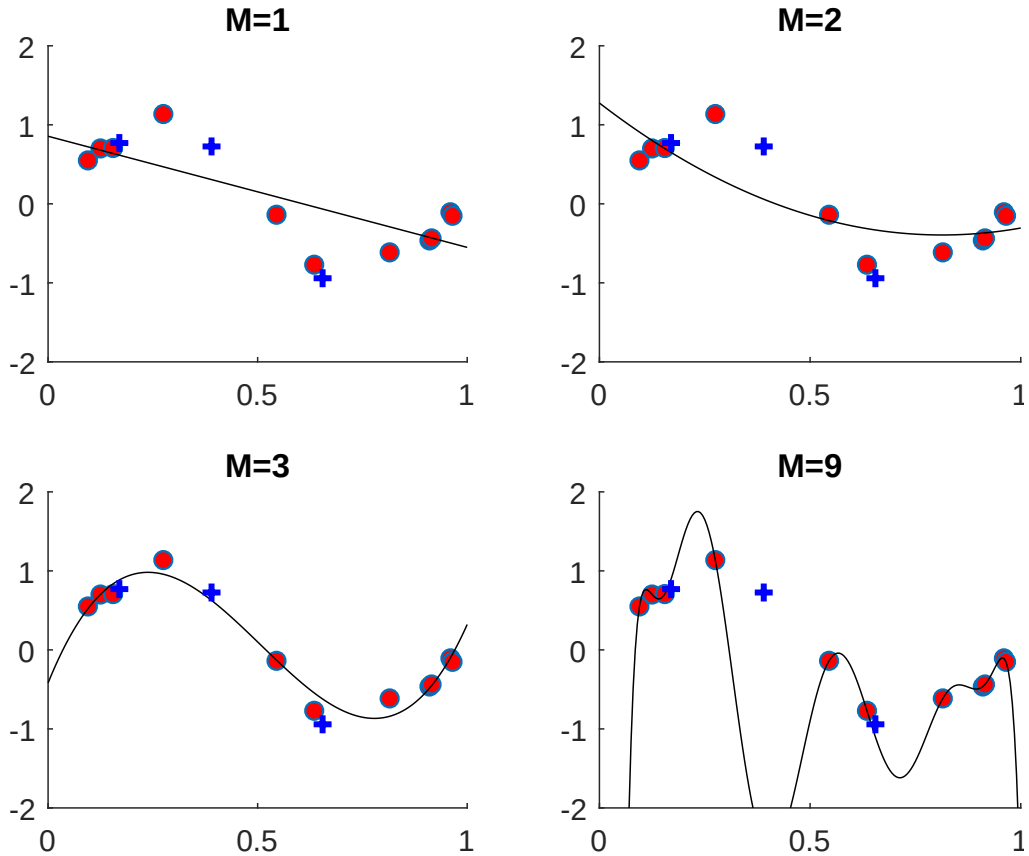
Hyperparameter λ (tuning parameter) determines the size of the penalty.

$$\lambda = \begin{cases} 0 & \text{standard least-squares} \\ (0, \infty) & \text{regularized regression} \\ \infty & \beta_1 = \beta_2 = \cdots = \beta_M = 0 \end{cases}$$

YES,... this IS very Bayesian, with a prior $\beta_i \sim N(0, 1/\lambda)$

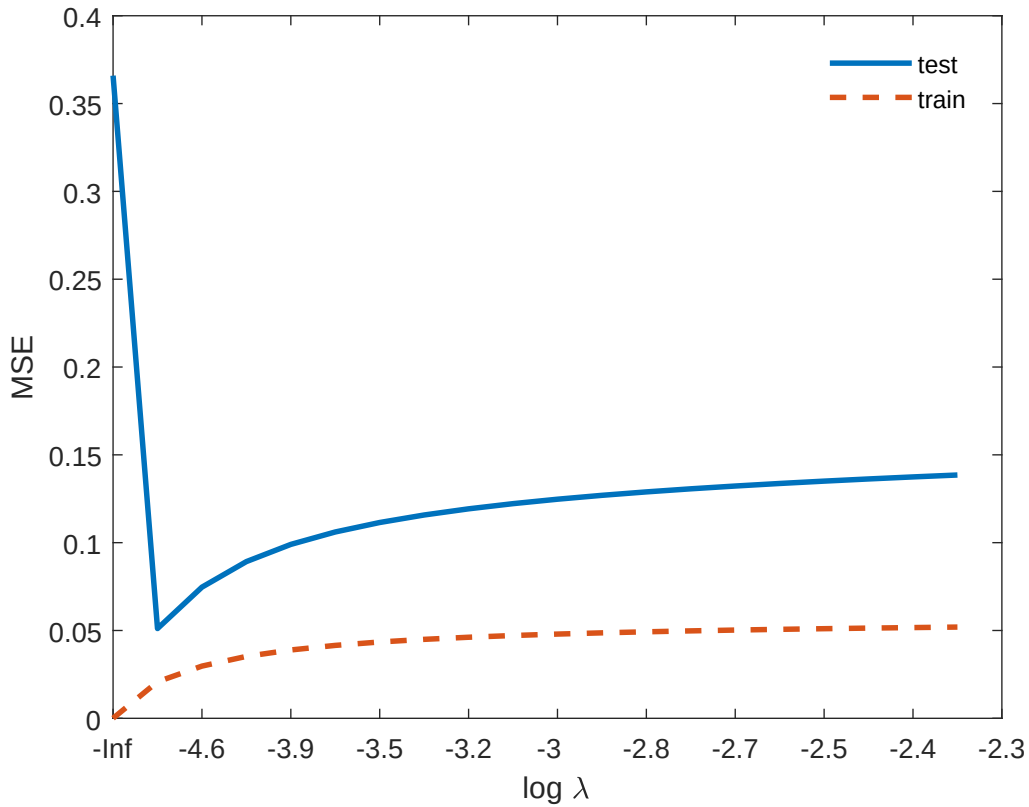
Regularization: Choosing Complexity

We can estimate models with general models with polynomial order $M = 9$ and $\lambda = \{1, \ldots, \lambda_{max}\}$ on the **training data** and evaluate each model on **test data**



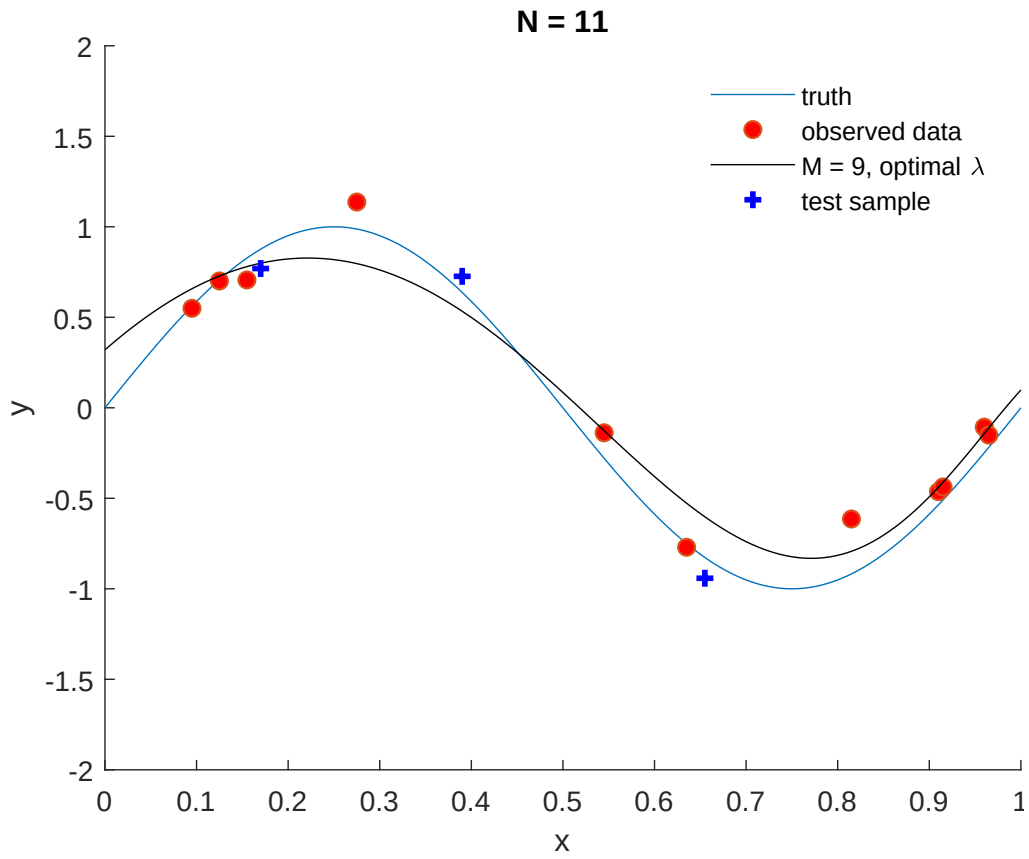
Regularization: Choosing Complexity

We can estimate models with general models with $M = 9$ and $\lambda = \{1, \ldots, \lambda_{max}\}$ on the **training data** and evaluate each model on **test data**



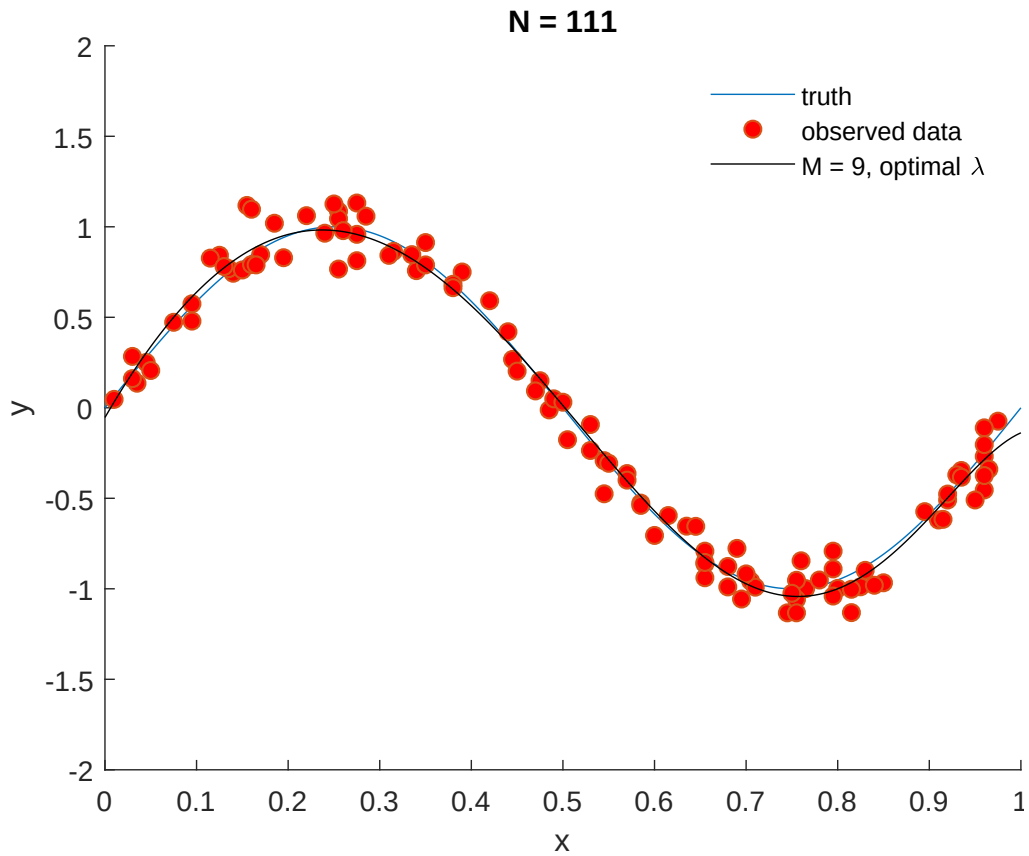
Regularization: Choosing Complexity

Regularized model with order $M = 9$, 11 observations and complexity determined using the test data searching for 'optimal' λ

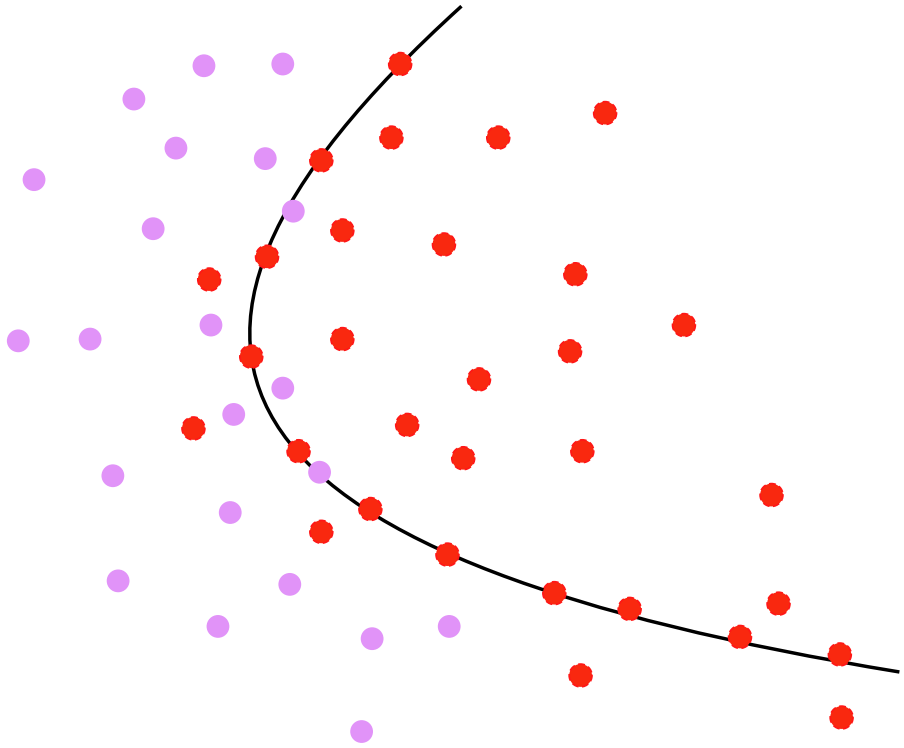


Regularization: Choosing Complexity

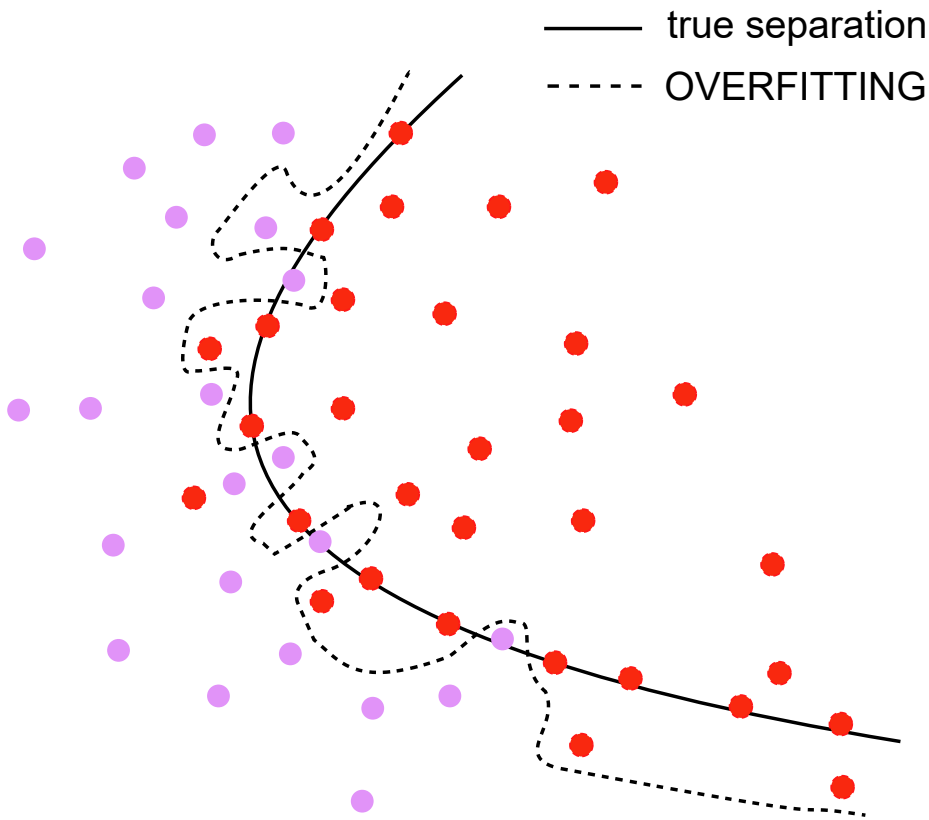
Regularized model with order $M = 9$, **111** observations and complexity determined using the test data searching for 'optimal' λ



Overfitting Classifiers. . .



Overfitting Classifiers. . .



The model must strike a balance between bias and variance over the new samples. . . [from the true, unkown, and hopefully stable DGP]

Regularization (2)– Beyond Parameter Priors

Apart from various parameter priors, is there anything else?

How about estimating many models using **boostrapped** data and **aggregating** their forecast? **BAGGING!**

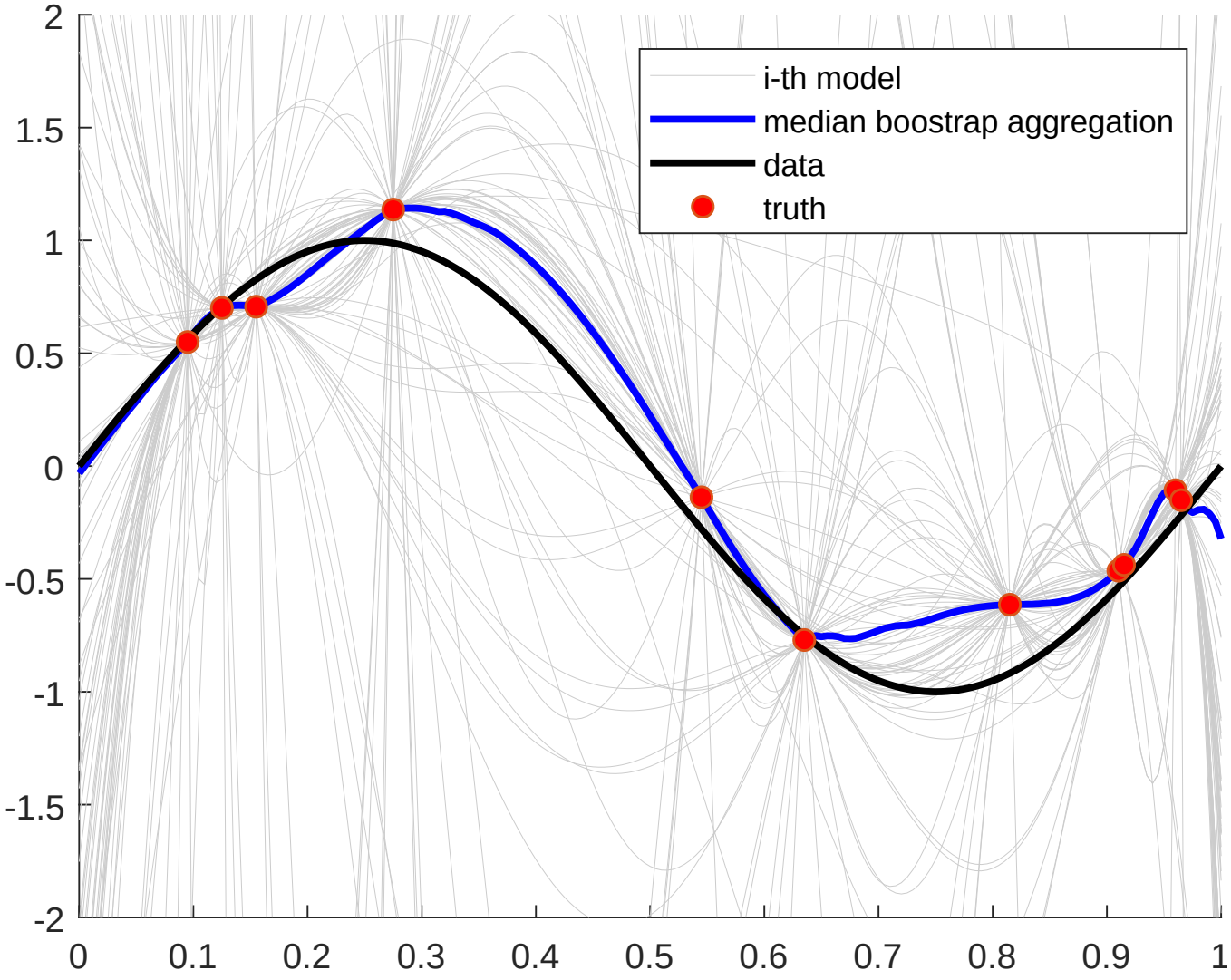
Bootstrap Aggregating:

- Common model ensemble method to lower variance
- Useful for unstable estimators, often used with **trees** (Random Forests)

Our case:

- Re-sample data with replacement R-times, $R = 90$
- Aggregate the model estimates, $\hat{y}_r$ using **median**

Regularization (2b)– Beyond Parameter Priors



Regularization (3)– Beyond Parameter Priors

Can we try something else? How about **data augmentation**?

Data augmentation:
Add modified/corrupted data to the sample to avoid overfitting.

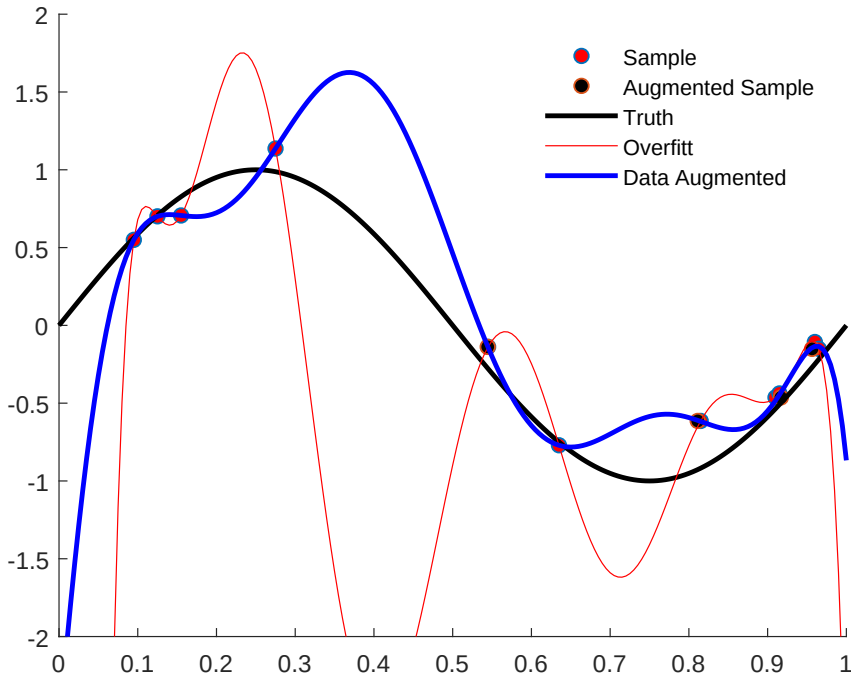
Examples:

- Add rotated images to the sample
- Add images/sounds corrupted by white noise to the sample
- Attempt to address unbalanced datasets for classification (SMOTE, etc.)

Our case:
Select 1/3 of the sample, create new data points by adding $\varepsilon_i \sim N(0, \sigma)$ to input x_i, keeping output, y_i unchanged...

Regularization (3b) – Data Augmentation

Add **noise** to portion of the data and **augment** the dataset. . .



Note: This is not an accident. Bishop (1998) shows training with noise amounts to L2 penalty regularization.

Regularization – Summary

Regularization process is adaptive and can take many forms.

Important examples:

- **Coefficient penalties**, sparsity shrinkage, parameter sharing, . . .
- Bootstrap aggregating (‘**Bagging**’) of models
- **Random perturbations** to model structure, model averaging, early stopping
- Artificial **feature corruption**
  (dropout, data augmenting, adding noise)
- **Theory** – putting a weight on a-priori **theory**, parameter sharing, . . .

Regularization – Hyper Parameter 'Tuning'

Regularization process introduces **hyper parameters.**

Hyper-parameters cannot be optimized using in-sample fit.

Regularization may WORSEN in-sample fit,
to help IMPROVE out-of-sample fit.

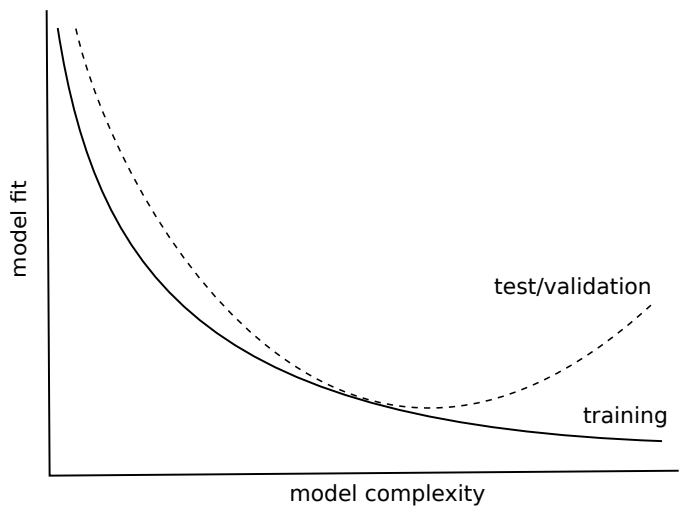
Algorithm	**Hyperparameter Examples**
parametric models	weight-decay penalties, $\sum_{i=1}^{p} \lvert\beta_i - 0\rvert^q$
trees, random forests	depth of trees, number of trees, minimum leaf size, ...
nearest-neighbor	number of neighbors
neural nets	weight-decay, drop-out, layer architecture
...	

Regularization – Hyper Parameter 'Tuning'

How to set the "**right values**" of tuning parameters?

1. **Setting them a priori**, as priors.
2. **Optimizing them** using different data than used for model training. (validation sample, cross-validation, . . .)

Get used to watching charts like this:



Fixing Overfitting

1. **Use more data**

   Given the model, increase in informative data size lowers overfitting

2. **Structural stabilization**

   Lower the number of free parameters, simplify complexity

   - Inflexible, doesn't adjust to data
   - Lowering model-hypothesis space may exclude 'truth'

3. **Regularization**

   Keep using complex model, impose prior restrictions on behavior . . .

   + Keeps large model-hypothesis space, stays flexible
   + Effective representational capacity adapts to information in the data

Bias-Variance Trade-Off

The frequentist view on model complexity is about **bias-variance trade-off**.

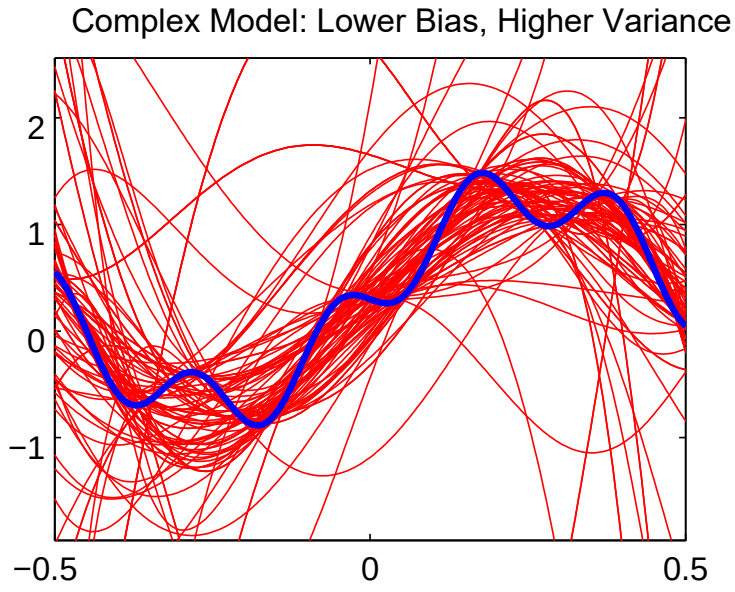
Treating the given data as a draw from an unknown population, the expected mean-square (population) error is a combination of bias and variance.

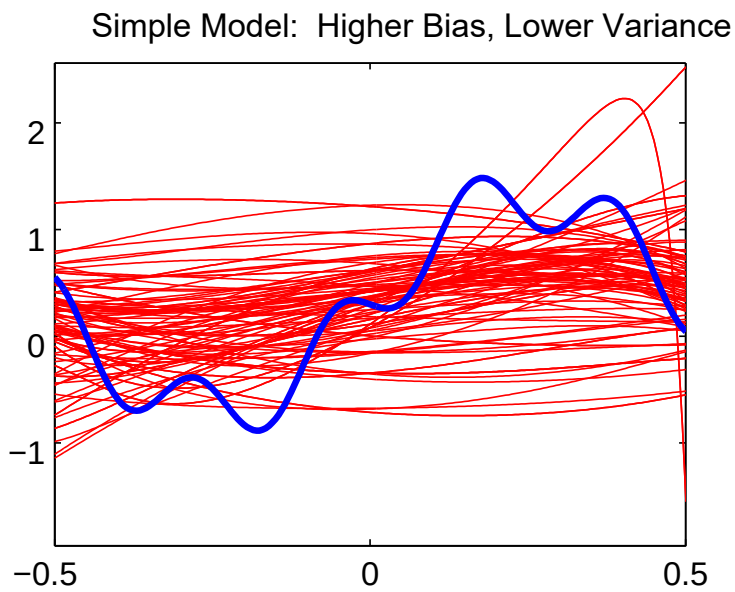
1. More complex model may have lower bias but large variance if re-estimated on new draws from the population

2. Less complex model may have higher bias but lower variance, being less sensitive to new draws from the population

In Bayesian setting when one marginalizes over parameters the over-fitting phenomenon is largely absent. . .

Bias-Variance Trade-Off





Bias-Variance Trade-Off

Decomposing **population** expected mean-square error:

$$\text{expected loss} = \text{bias}^2 + \text{variance} + \text{irreducible noise}$$

To minimize Expected MSE:
Trade lower variance for higher bias,
if possible

By Gauss-Markov theorem, least squares have the smallest variance among all the **unbiased** estimates... If $\hat{\beta}$ has large variance, $\hat{y} = \hat{\beta}' x$ has high variance too

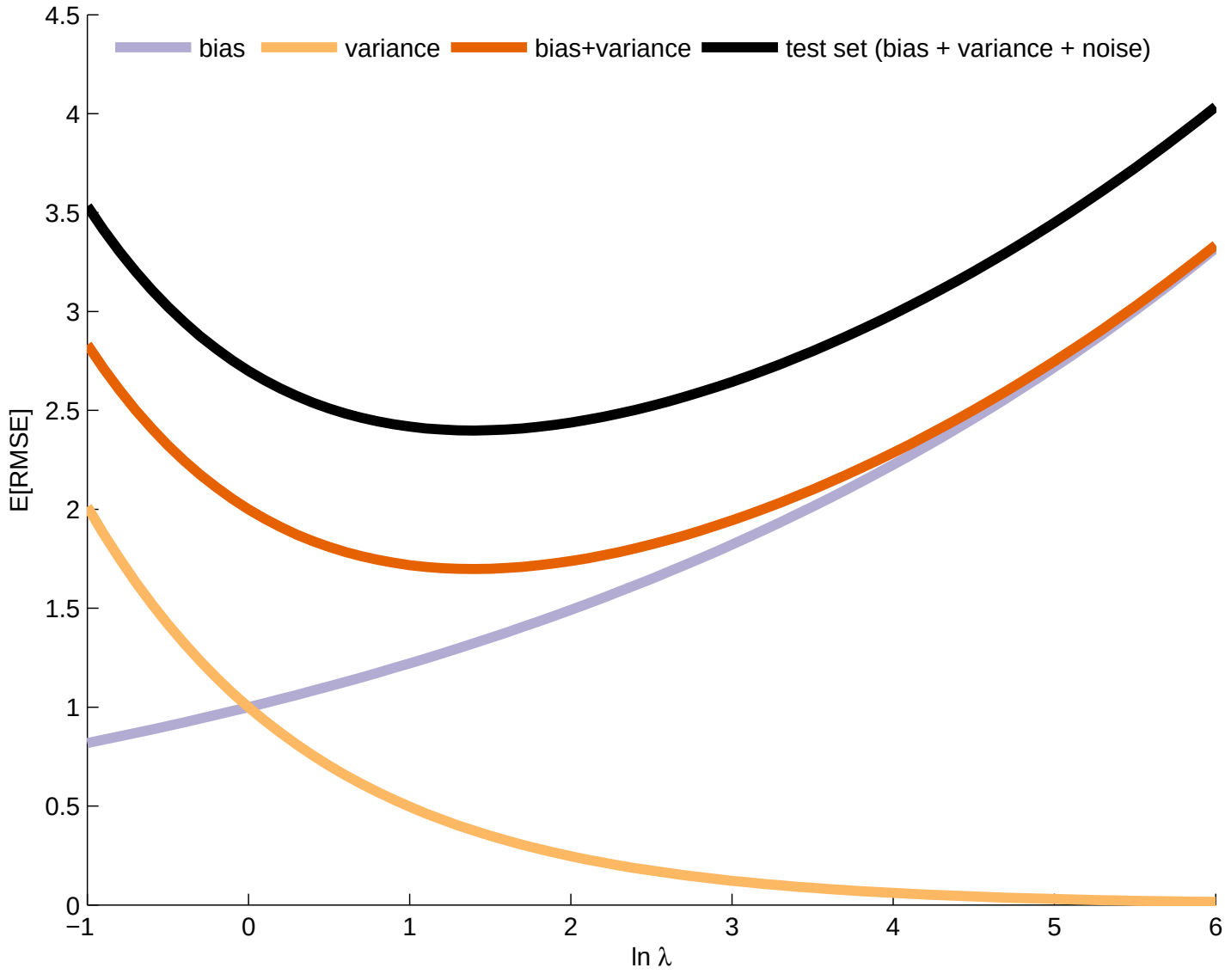
...statistical learning trades bias for lower variance, in order for models to generalize well to new data

Bias-Variance Trade-Off

Let's $y = f(x)$. For a given estimate $\hat{y} = \widehat{f}(x)$, we assess population mean squared error

$$
\begin{aligned}
E[(y - \widehat{f}(x))^2] &= E[y^2] + E[\widehat{f}^2] - 2E[y\widehat{f}] \\
&= (\text{var}[y] + E[y]^2) + (\text{var}[\widehat{f}] + E[\widehat{f}]^2) - 2fE[\widehat{f}] \\
&= \text{var}[y] + \text{var}[\widehat{f}] + (f - E[\widehat{f}])^2 \\
&= \sigma^2 + \text{variance} + \text{bias}
\end{aligned}
$$

Bias-Variance Trade-Off



Wonkish: Bias-Variance Trade-Off for Classification

Say, if using a 0-1 loss, $L(\hat{y}, y) = I(\hat{y} = y)$, bias and variance combine multiplicatively...

With 0-1 loss, estimation errors that leave you making the right 0/1 decision are not painful. Also, when on the wrong side of the decision boundary with a negative bias, it pays off to *increase* variance.

The choice of **hyperparameters** (tuning parameters) and/or model selection thus should rather focus on estimation of **expected loss**.

Regularization – Wonkish...

The regularization examples were meant to provide intuition...

For **LINEAR regression**, the ridge regression, data augmentation, bagged data-augmentation, or feature dropout all imply an L_2 penalty...[if you do all the algebra]

For more complex, nonlinear models, the regularization penalties implied by these techniques start to significantly differ...

Different forms of regularization are suitable for different forms of training, given the problem and the computation tools used

See Wager, Wang, and Liang (Nov 2013)

SUPERVISED LEARNING PROBLEM: SUMMARY

The Elements of Learning

The Problem:

An **unknown** mapping, u, from input space, $\mathcal{X}$, to output space, $\mathcal{Y}$, $u : \mathcal{X} \rightarrow \mathcal{Y}$, such that $y_j = f(x_j)$.

Hypothesis Set:

A set $\mathcal{R}(f)$ containing candidate mappings (models, hypotheses) mean to approximate the unknown mapping u.

Observed Data:

A finite set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$, or $\mathcal{D} = \{(x_1, x_2), \ldots, (x_k, x_k)\}$.

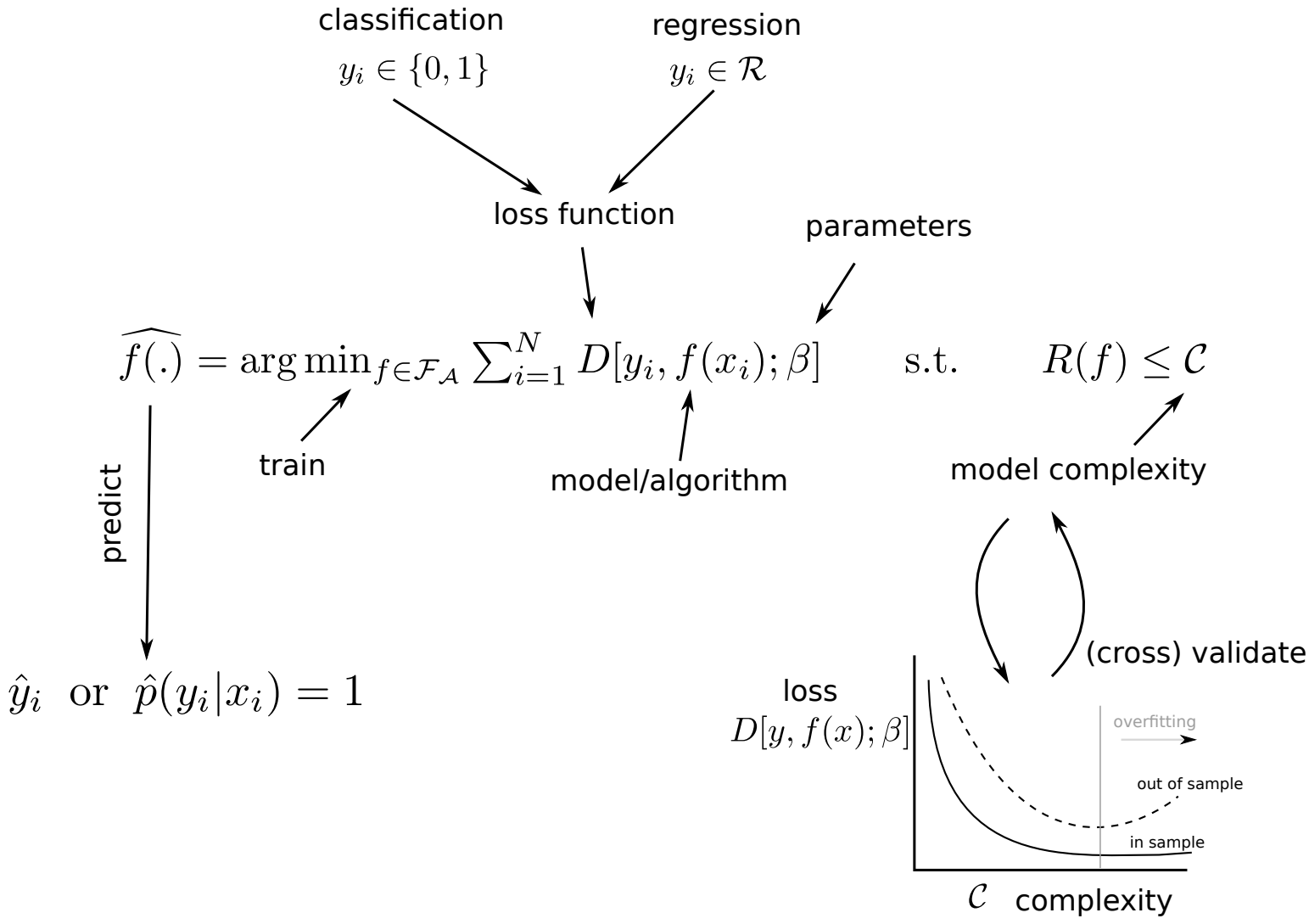
Algorithm:

An algorithm, $\mathcal{A}$, that uses observed data, $\mathcal{D} \in$, to learn a mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$, $f \in \mathcal{R}(f)$.

Error Measure (Distance/Loss/Utility Function)

A metric $\mathcal{E}$ used by the algorithm $\mathcal{A}$ to choose a mapping g.

Machine Learning



Supervised Learning = Training + Validation

1. Using **training data** optimize for $f(.)$
2. Using **validation set** search hyper-parameters to optimize out-of-sample fit

Choose $\widehat{f(.)}$ such that

$$\begin{aligned}\widehat{f(.)} &= \arg\min_{f \in \mathcal{F}_A} \sum_{i=1}^{N} D[y_i, f(x_i)] \qquad s.t. \qquad R(f) \leq c \\ &= \arg\min_{f \in \mathcal{F}_A} \sum_{i=1}^{N} D[y_i, f(x_i)] - \lambda\{R(f) - c\}\end{aligned}$$

$D(.)$	loss function due to difference between y_i and $f(x_i)$
β	coefficients to be estimated
$\mathcal{F}_A$	function space with architecture A
$R(f)$	regularizer of the function $f(.)$
λ	regularization tightness, "shadow price" ...

Supervised Learning Problem Defined (2)

Example: **Bayesian [V]AR** (ridge-regression with K lags)

$$\widehat{f(.)} \quad = \quad \underset{f \in \mathcal{F}_A}{\arg\min} \sum_{i=1}^{N} D[y_i, f(x_i)] - \lambda\{R(f) - c\}$$

$D(.)$	Quadratic loss, $[y_i - f(x_i)]^2$
$\mathcal{F}_A$	Linear model with coefs β, K lags: $y_v = f(x) = \beta_0 + \beta_1 y_{t-1} + \cdots + \beta_k y_{t-k}$
$R(f)$	Ridge (Normal prior): $\sum_{j=1}^{K} (\beta_{ij} - 0)^2 \times j^{\lambda_1}$
λ	A-Priori Regularization tightness,
	λ_0 – overall tightness, λ_1 – higher lag, larger shrinkage to zero

LEARNING:
1) For **given** $D, \mathcal{F}_A, R(f)$ and λ use **training data** to find β.
2) Use **validation data set** to search for λ, number of lags, etc.

Thank you for your patience. . .