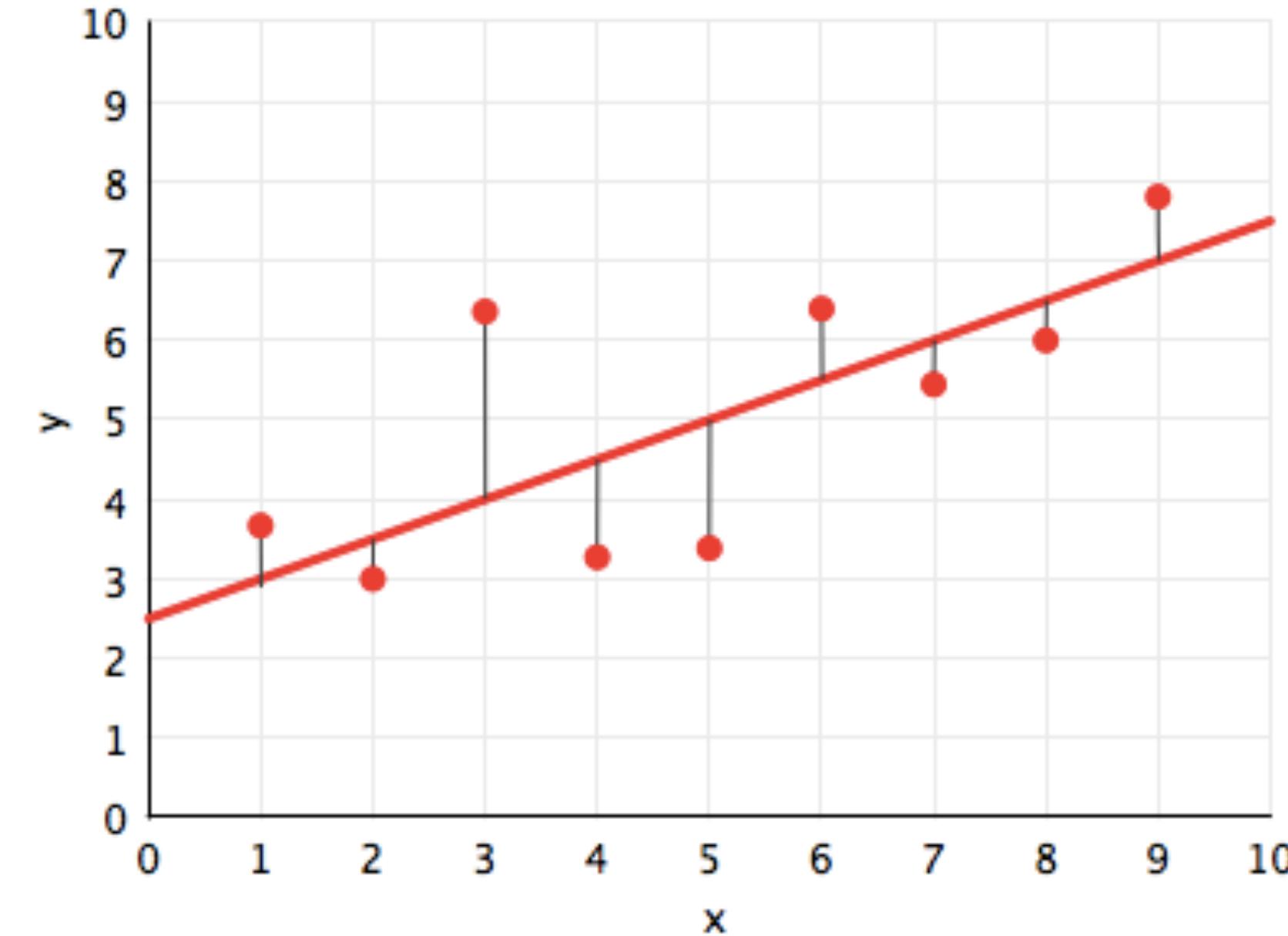


univ.AI

A. Learning a Model

(of the right complexity)

RISK: What does it mean to FIT?



Minimize distance from the line?

$$R_{\mathcal{D}}(h_1(x)) = \frac{1}{N} \sum_{y_i \in \mathcal{D}} (y_i - h_1(x_i))^2$$

Minimize squared distance from the line.

$$g_1(x) = \arg \min_{h_1(x) \in \mathcal{H}_1} R_{\mathcal{D}}(h_1(x)).$$

Get intercept w_0 and slope w_1 .

HYPOTHESIS SPACES

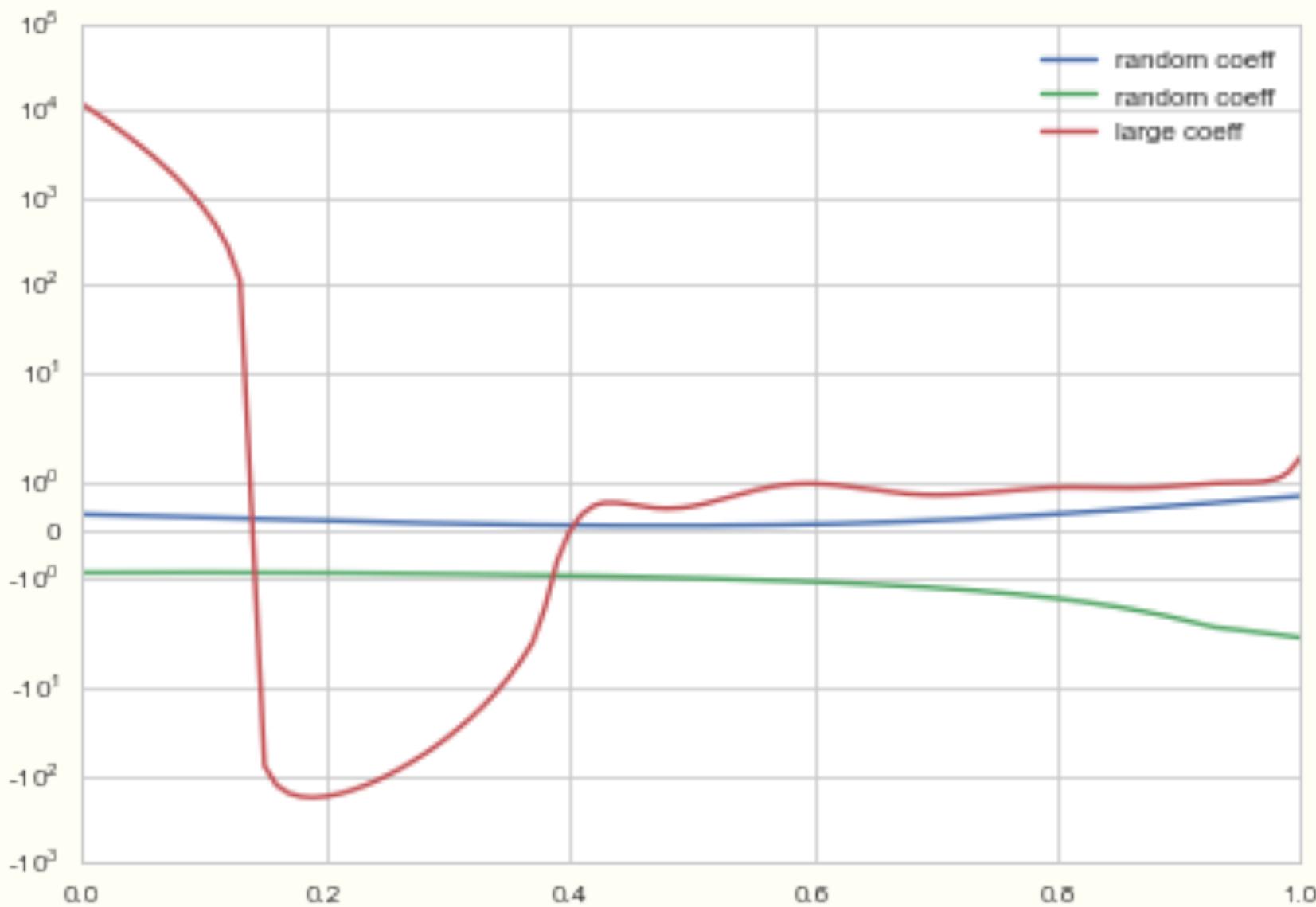
For example, a polynomial looks so:

$$h(x) = \theta_0 + \theta_1 x^1 + \theta_2 x^2 + \dots + \theta_n x^n = \sum_{i=0}^n \theta_i x^i$$

All polynomials of a degree or complexity d constitute a hypothesis space.

$$\mathcal{H}_1 : h_1(x) = \theta_0 + \theta_1 x$$

$$\mathcal{H}_{20} : h_{20}(x) = \sum_{i=0}^{20} \theta_i x^i$$



1. Approximation

Learning Without Noise...

Constructing a sample from a population

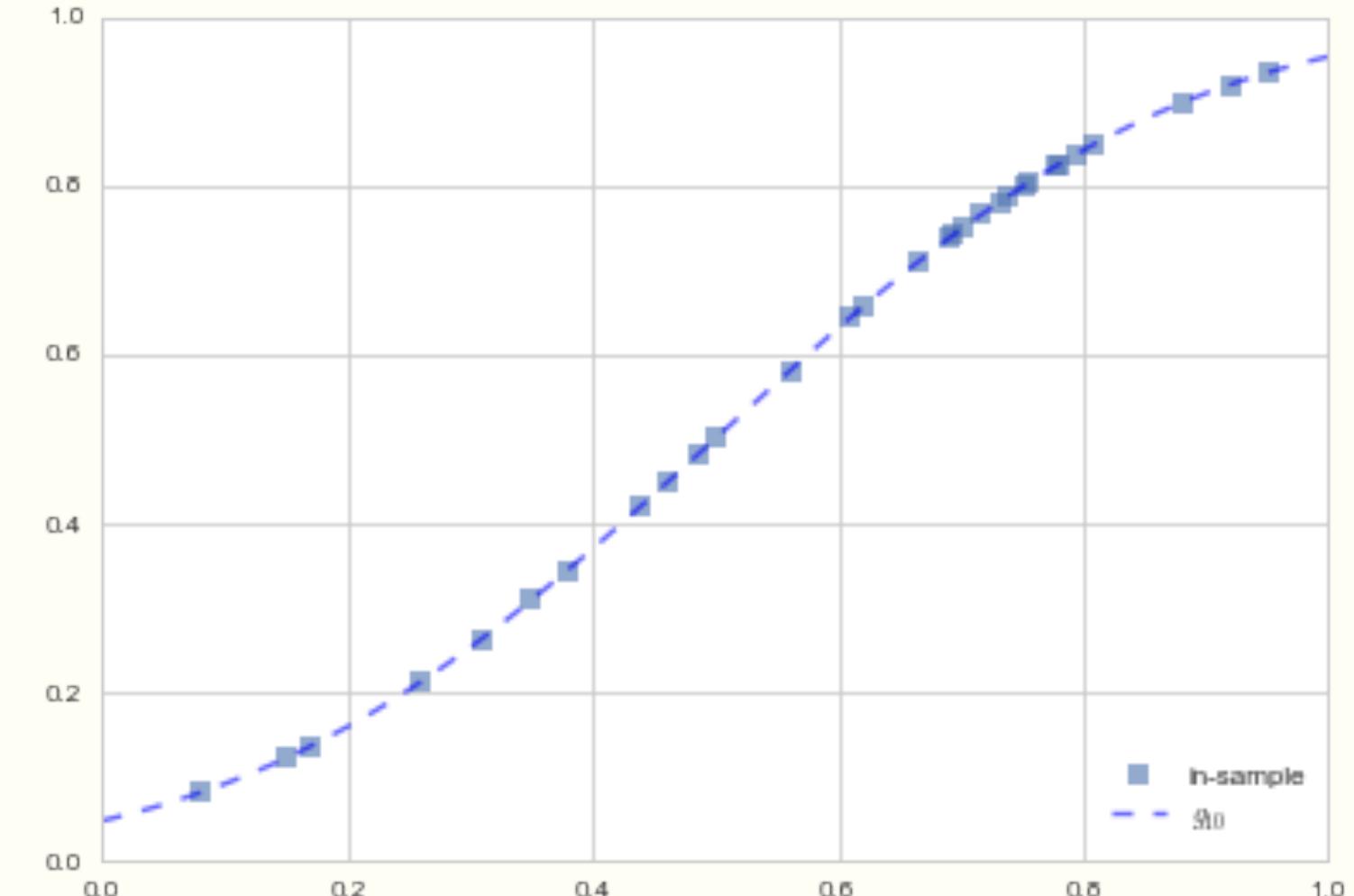
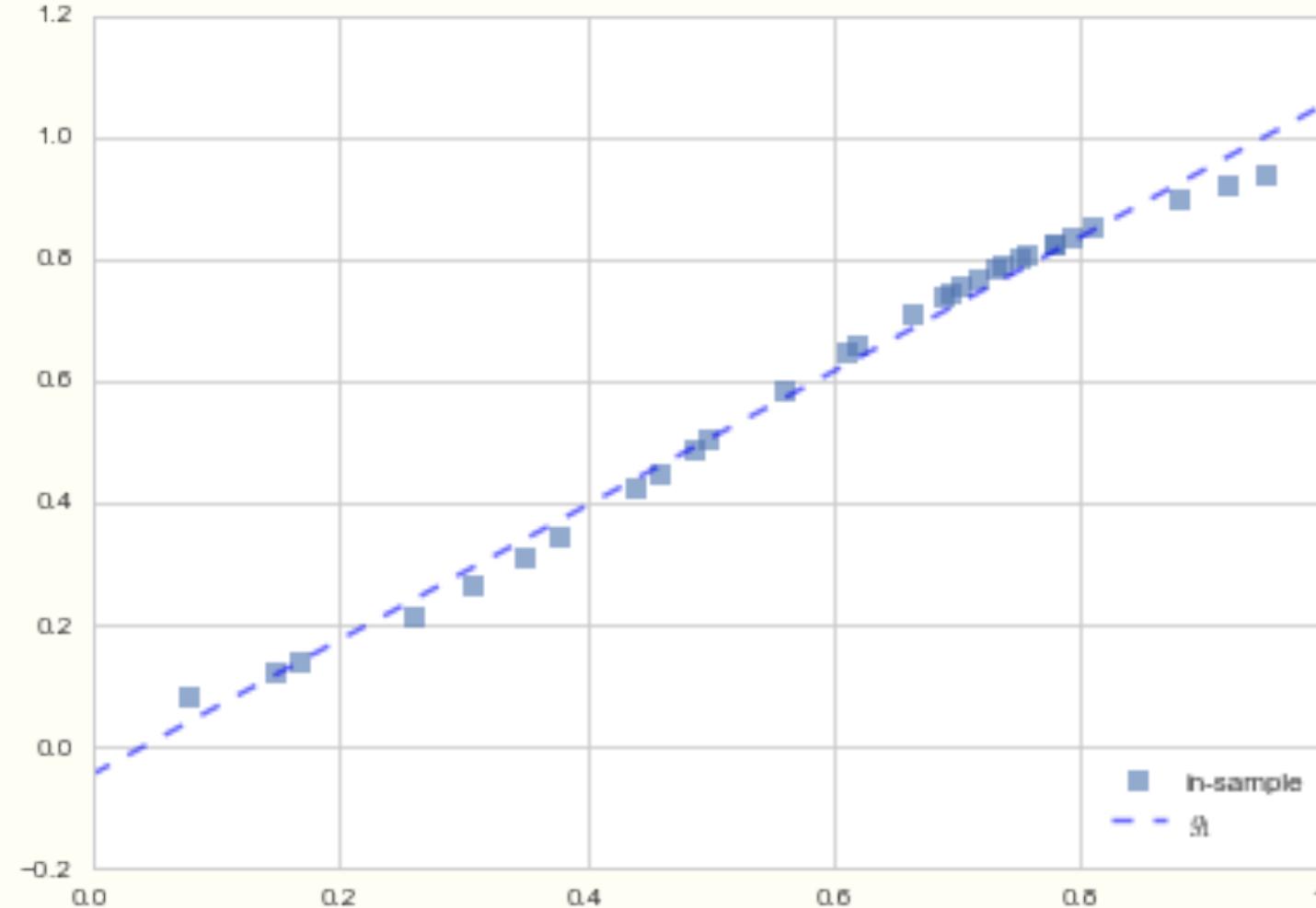
Well usually you are only given a sample. What is it?

Its a set of (x, y) points chosen from the population.

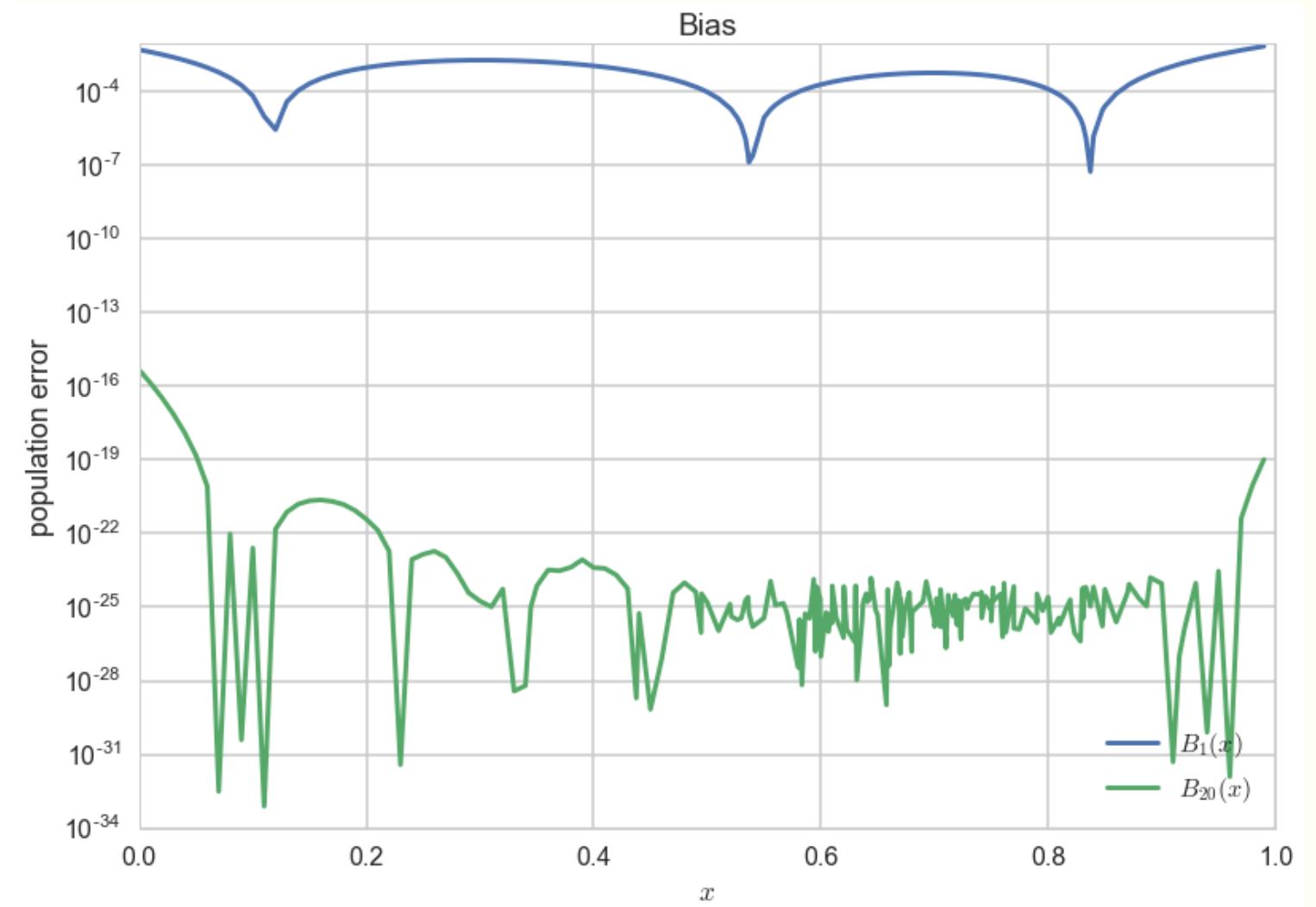
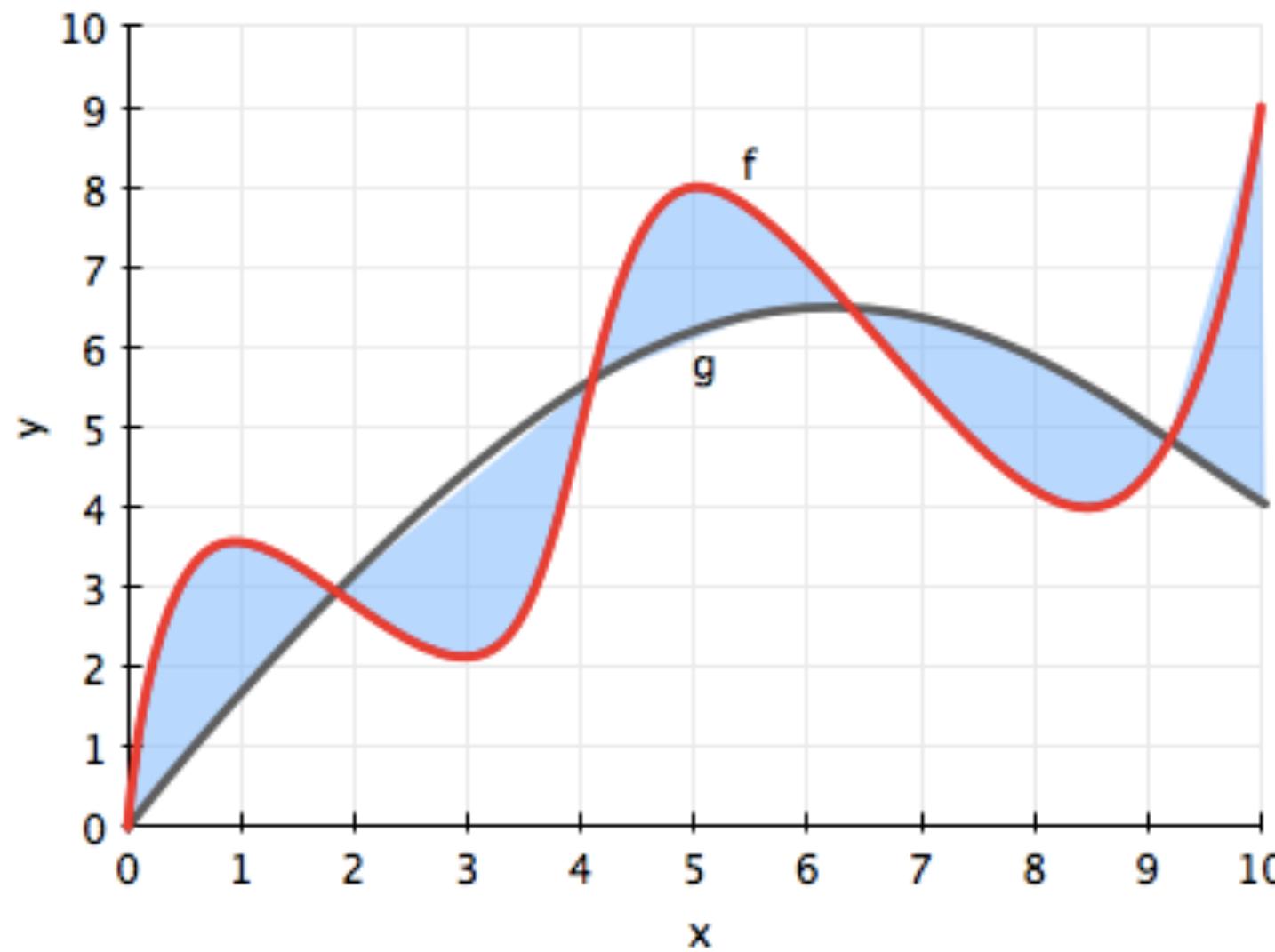
If you had the population you could construct many samples of a smaller size by randomly choosing subsamples of such points.

This is a game we will play...

A sample of 30 points of data. Which fit is better? Line in \mathcal{H}_1 or curve in \mathcal{H}_{20} ?

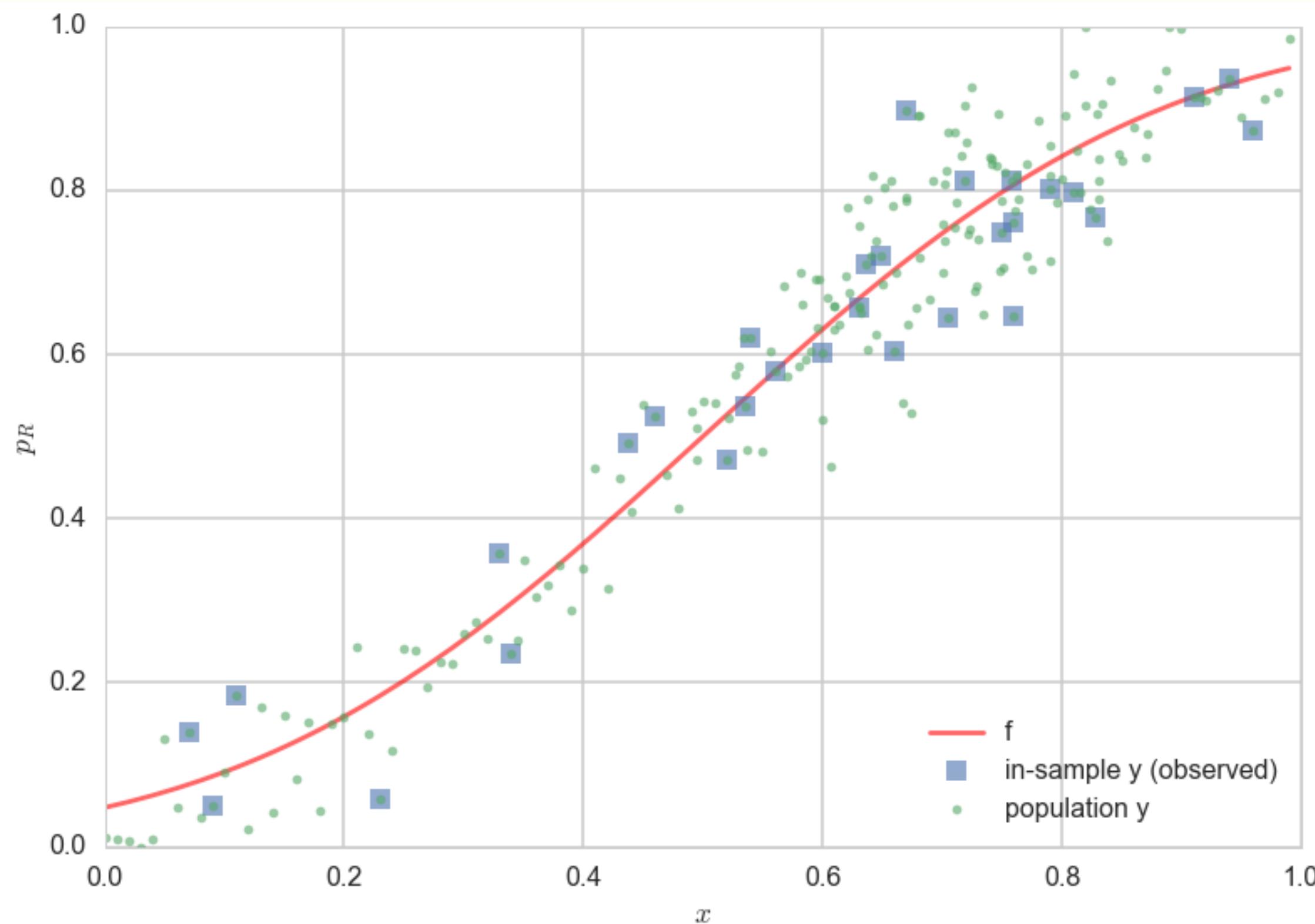


Bias or Mis-specification Error



2. THE REAL WORLD HAS NOISE

(or finite samples, usually both)

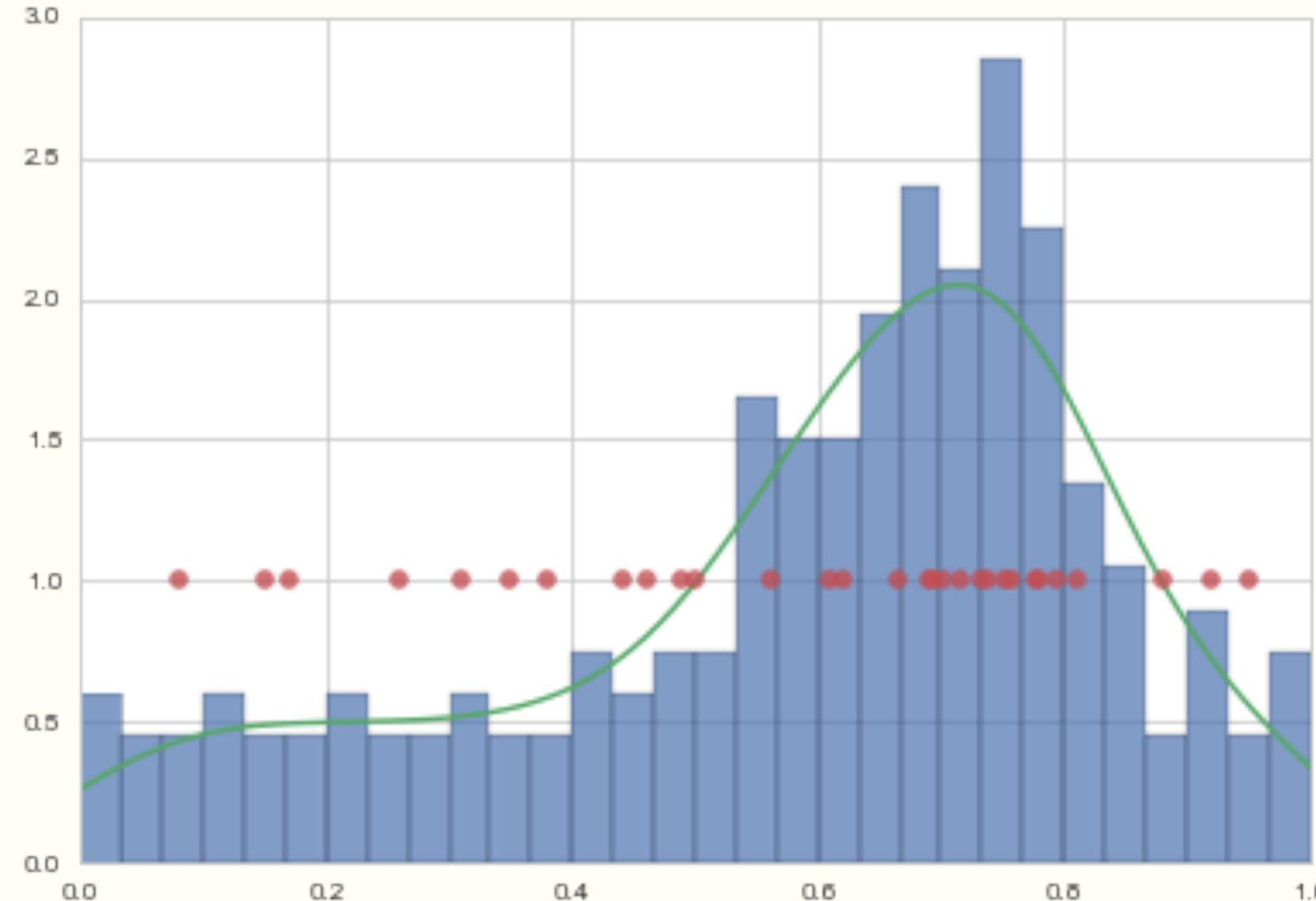


Statement of the Learning Problem

The sample must be representative of the population!

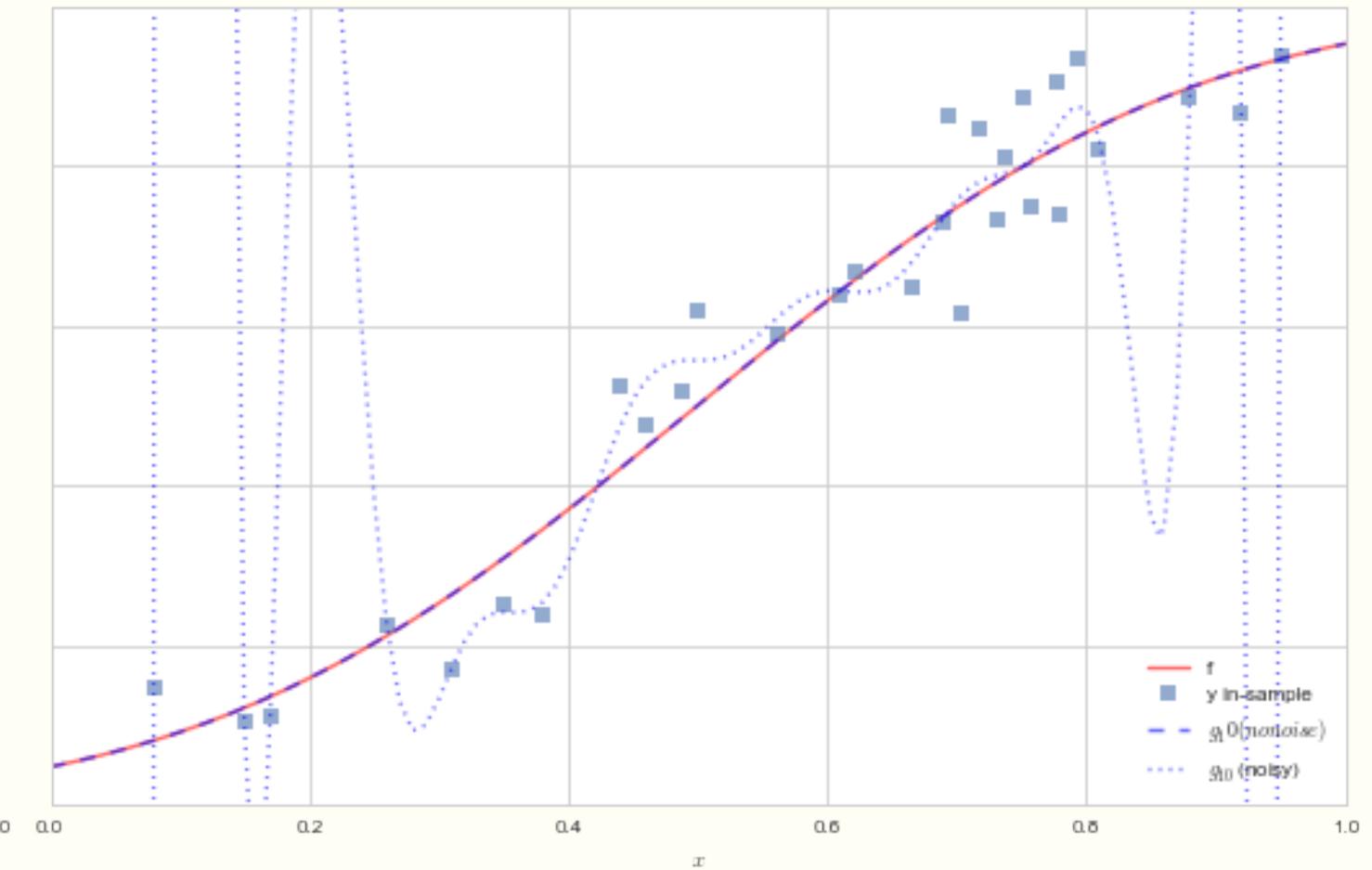
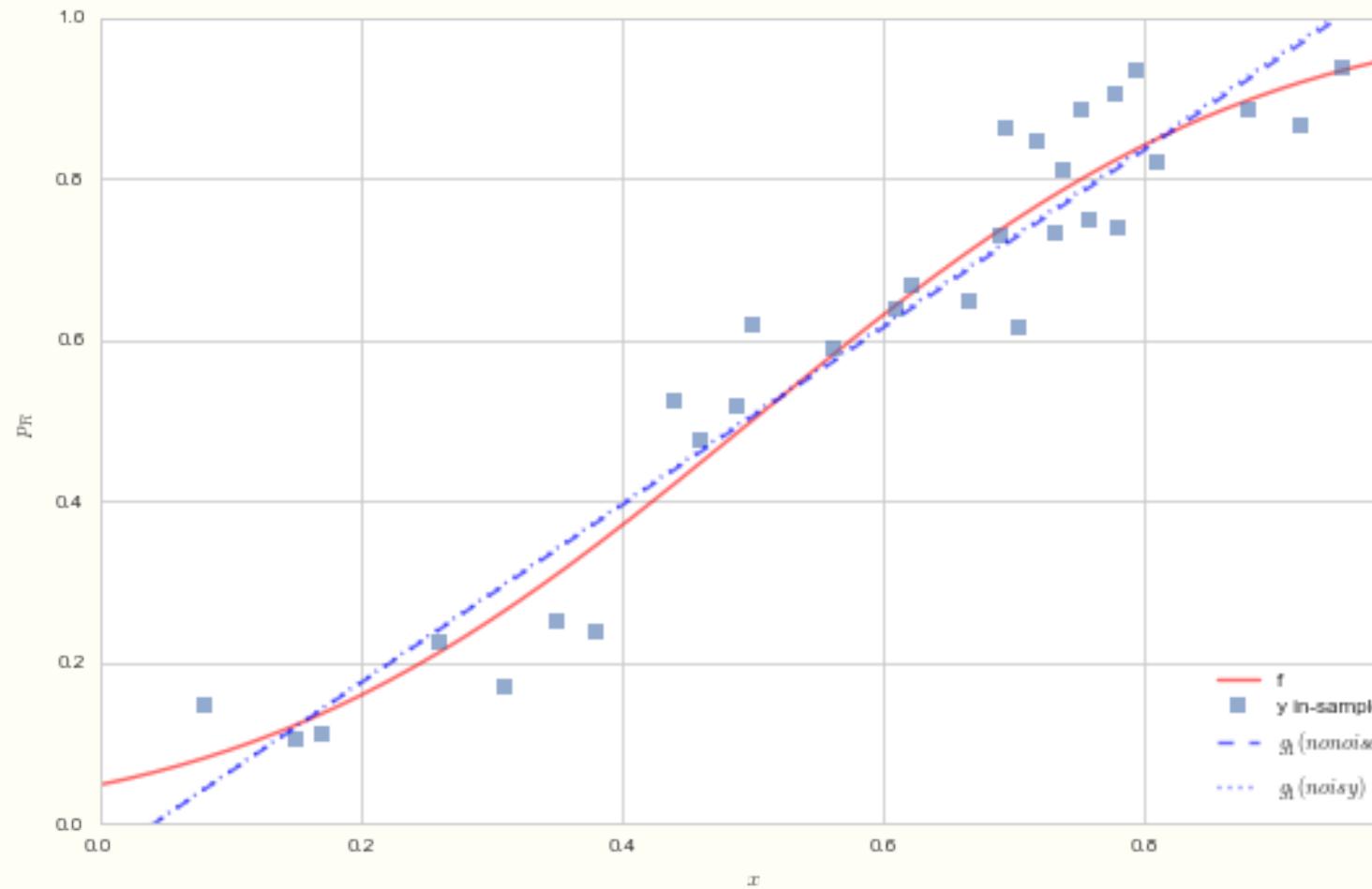
$$\begin{aligned} A : & R_{\mathcal{D}}(g) \text{ smallest on } \mathcal{H} \\ B : & R_{out}(g) \approx R_{\mathcal{D}}(g) \end{aligned}$$

- A: In-sample risk is small
B: Population, or out-of-sample risk is WELL estimated by in-sample risk. Thus the out of sample risk is also small.



Which fit is better now?

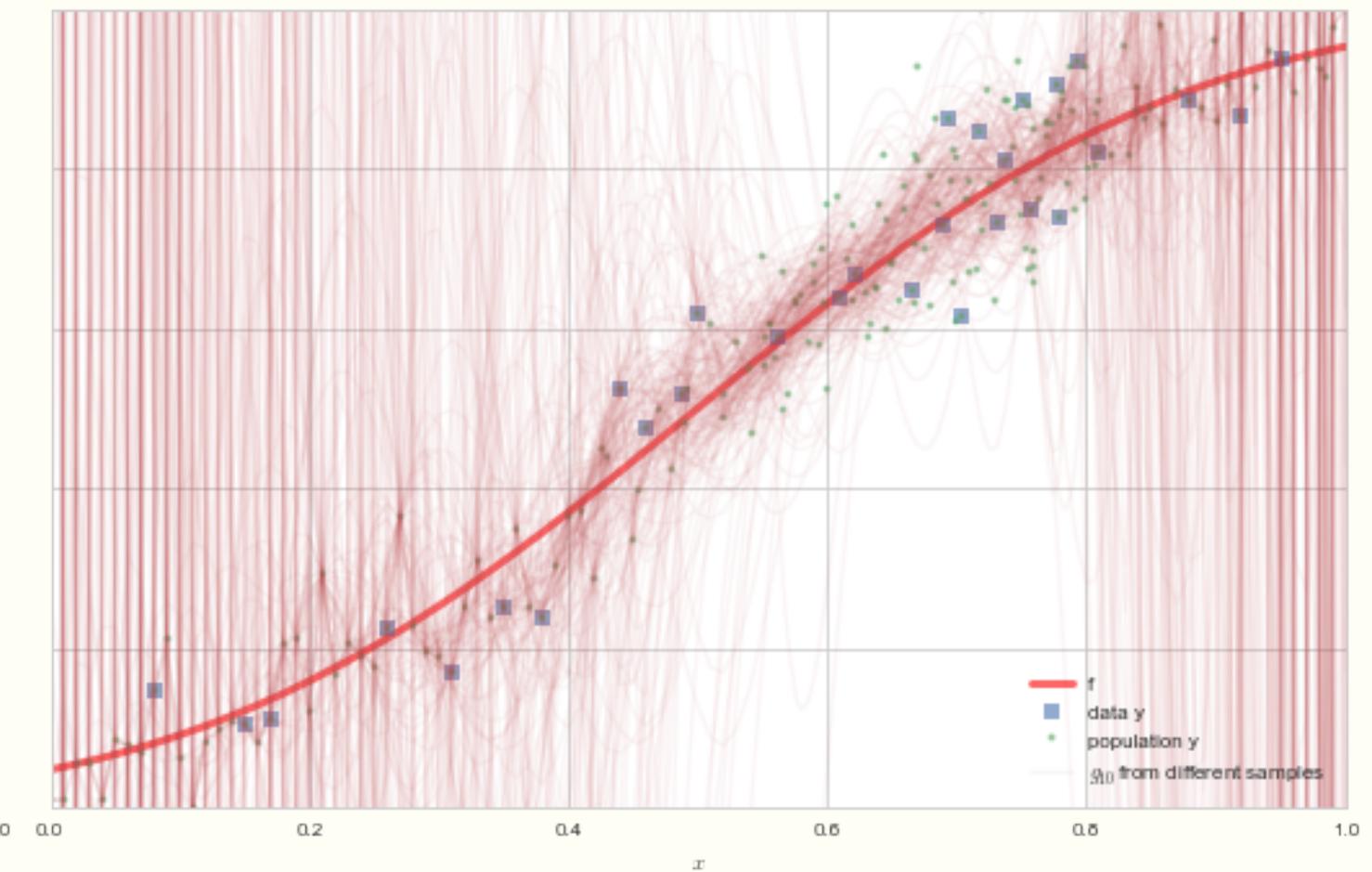
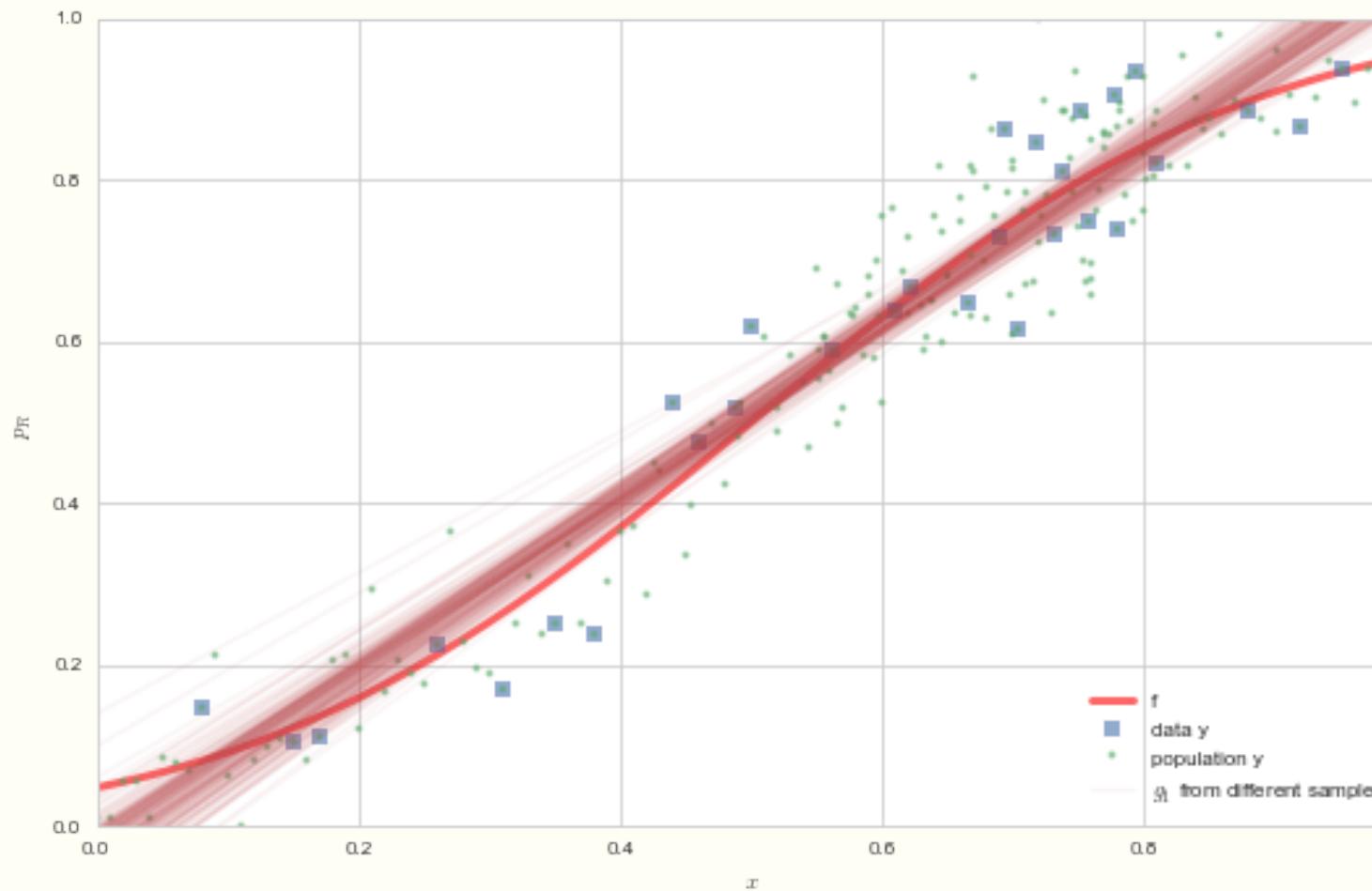
The line or the curve?



Training sets

- look at fits on different "training sets \mathcal{D} "
- in other words, different samples
- in real life we are not so lucky, usually we get only one sample
- but lets pretend, shall we?

UNDERFITTING (Bias) vs OVERFITTING (Variance)

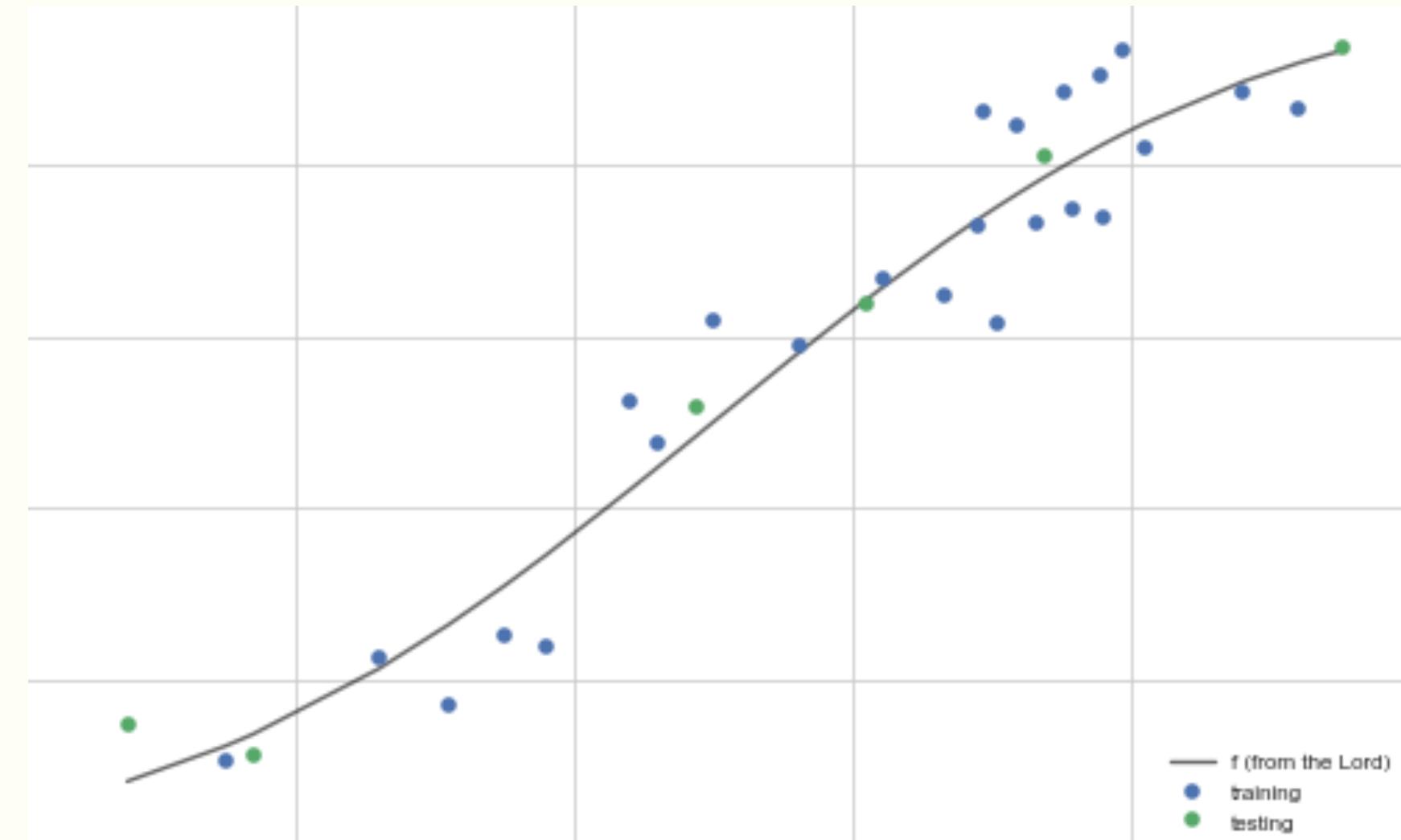
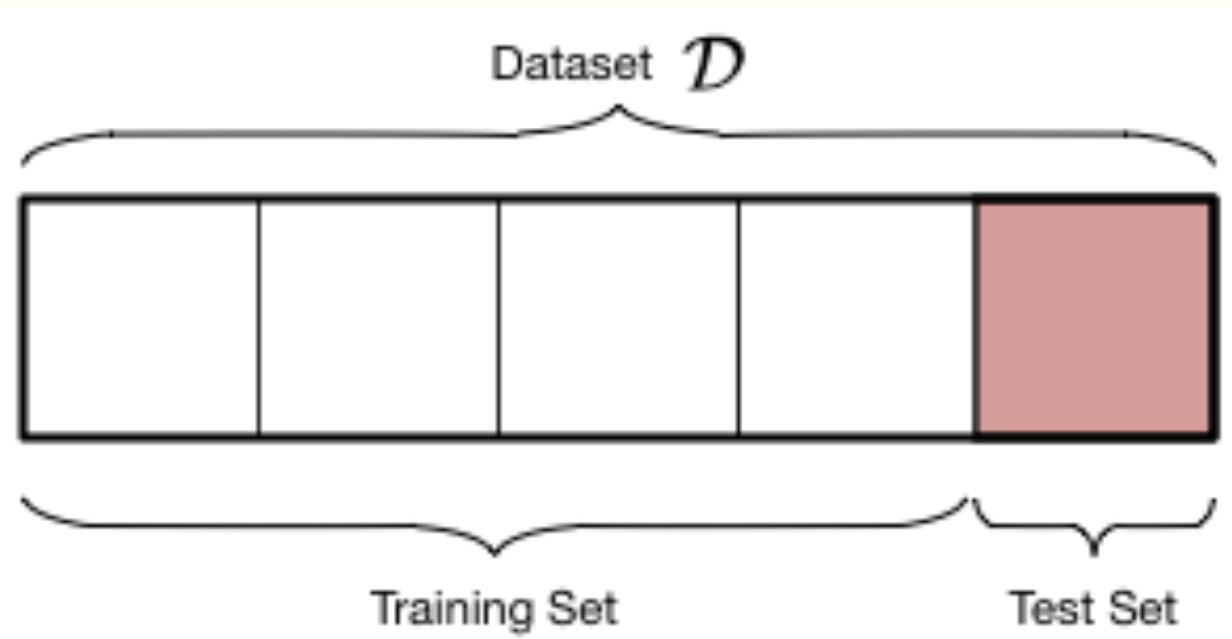


3. Complexity amongst Models

How do we estimate

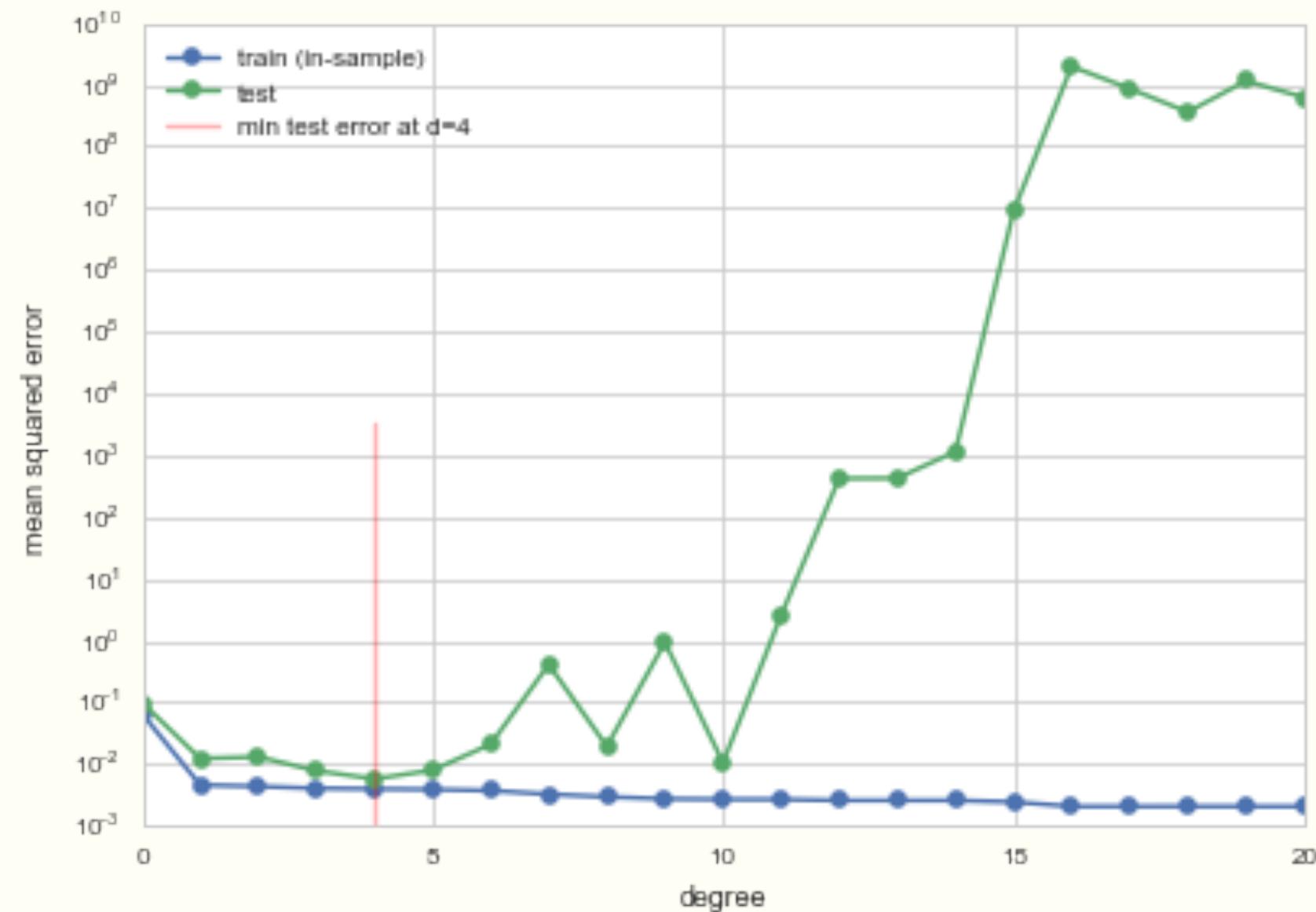
out-of-sample or
population error R_{out}

TRAIN AND TEST

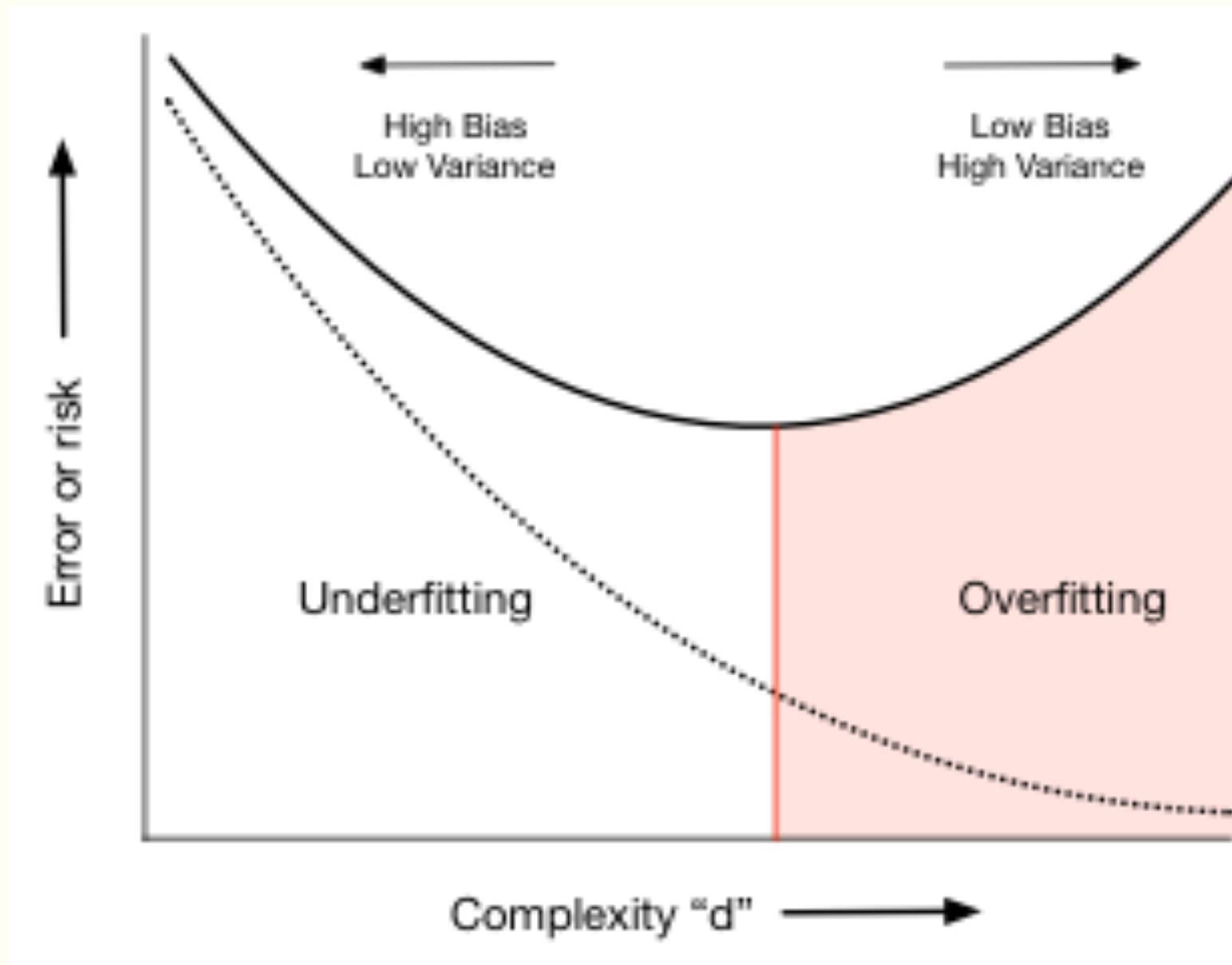


MODEL COMPARISON: A Large World approach

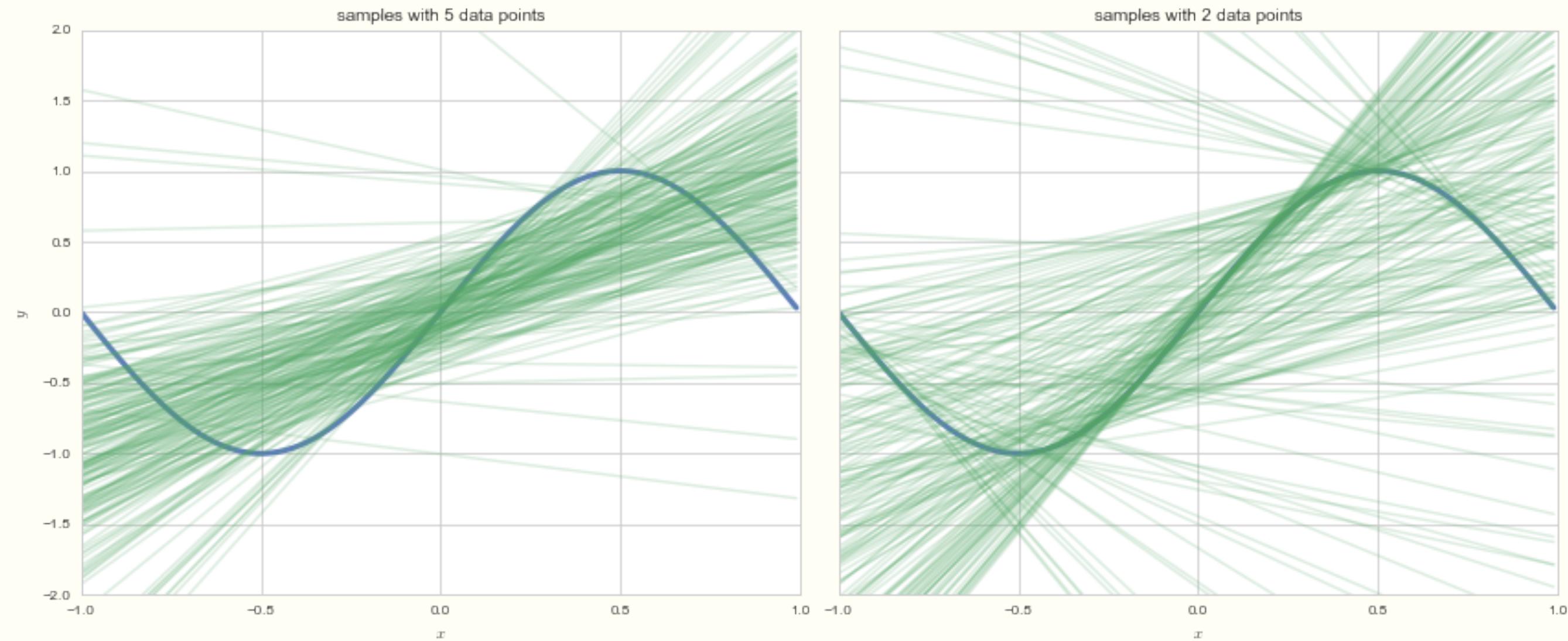
- want to choose which Hypothesis set is best
- it should be the one that minimizes risk
- but minimizing the training risk tells us nothing: interpolation
- we need to minimize the training risk but not at the cost of generalization
- thus only minimize till test set risk starts going up



Complexity Plot



DATA SIZE MATTERS: straight line fits to a sine curve



Corollary: Must fit simpler models to less data! This will motivate the analysis of learning curves later.

4. Validation

Do we still have a test set?

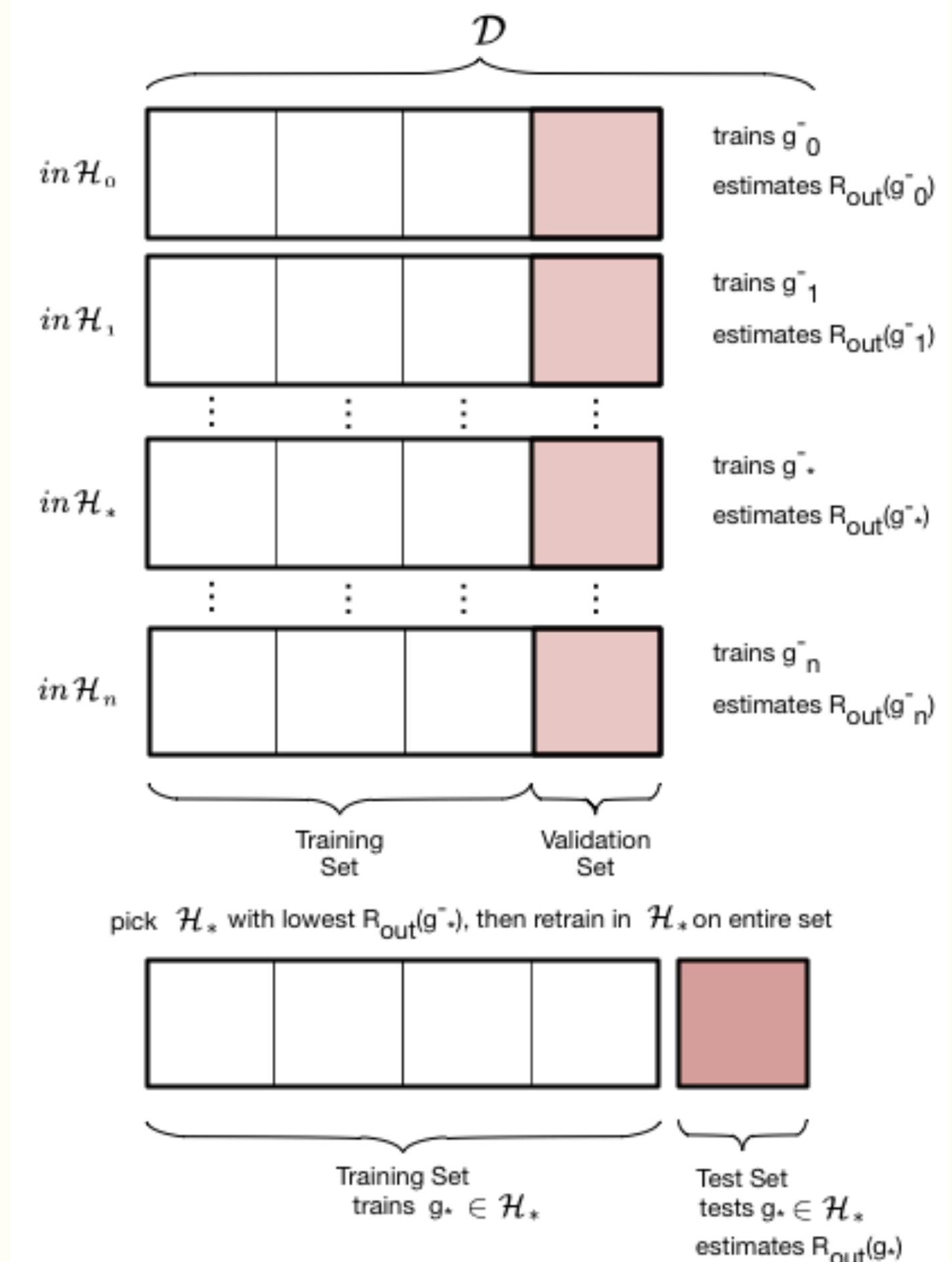
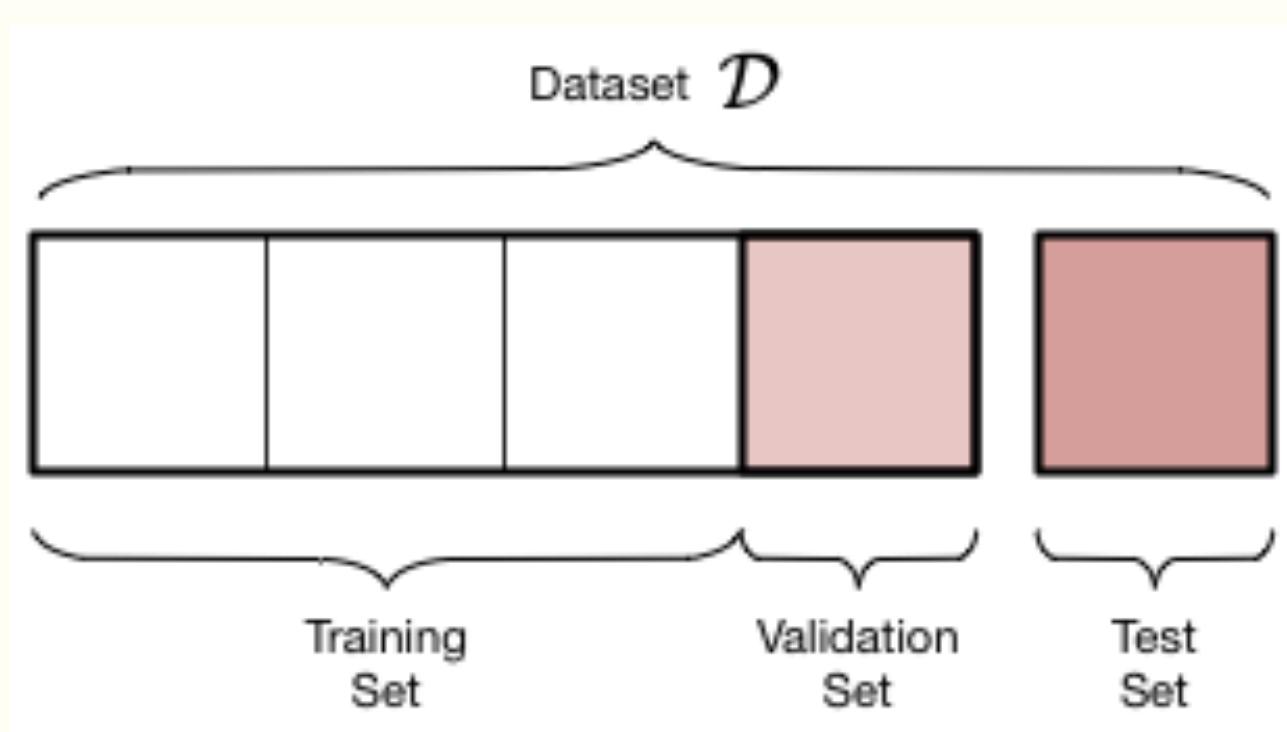
Trouble:

- we dont have the error bars on our risk estimate
- "visually fitting" a value of d \implies **contaminated test set.**

The moment we **use a set in the learning process, it is not a test set.** The risk estimated on the test set now underestimates the risk on the population.

VALIDATION

- train-test not enough as we *fit* for d on test set and contaminate it
- thus do train-validate-test



We want to find complexity (a hyperparameter):

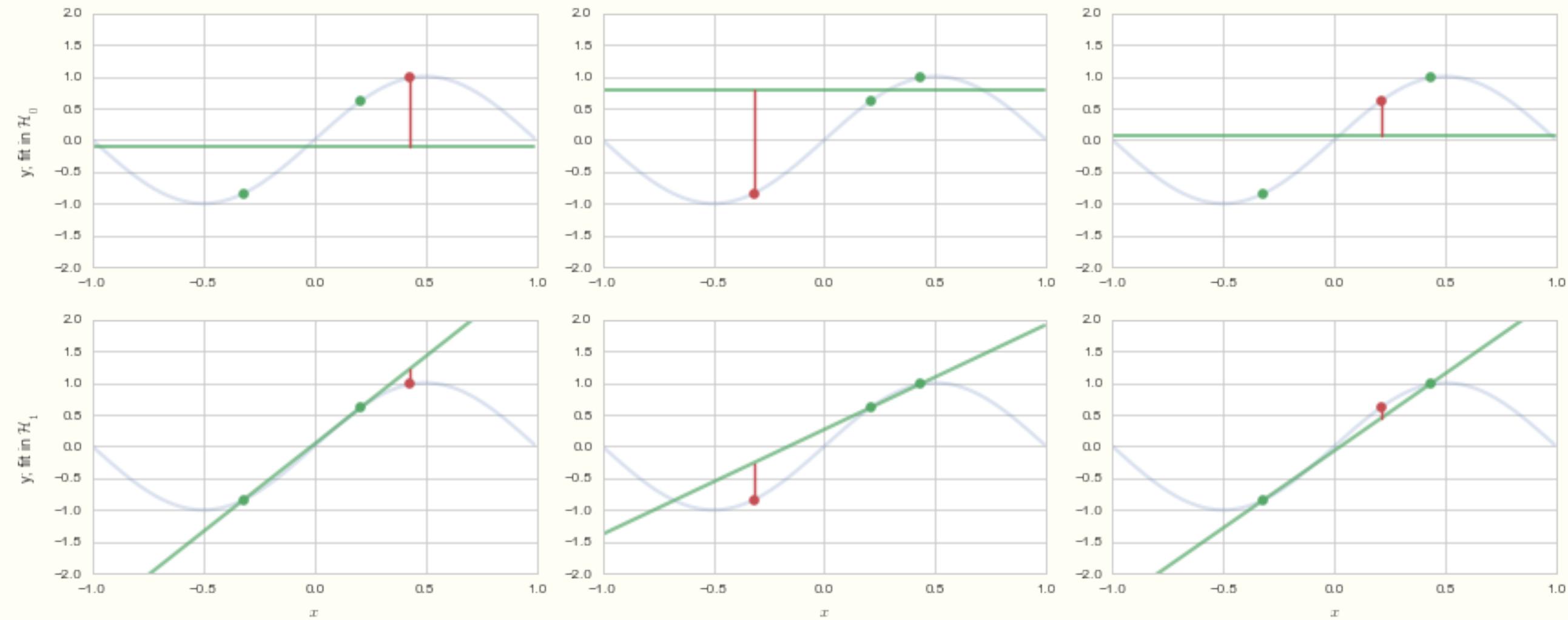
- we **wrongly** already attempted to fit d on our previous test set.
- choose instead the d^*, g^{-*} combination with the lowest validation set risk.
- $R_{val}(g^{-*}, d^*)$ has an optimistic bias since d effectively fit on validation set

Then Retrain on entire set!

- finally retrain on the entire train+validation set using the appropriate d^* . This gives us g^* (its a different model!). Why? We want to finally compute error on the held out test set and..
- works as training for a given hypothesis space with more data typically reduces the risk even further.

Whats the problem?

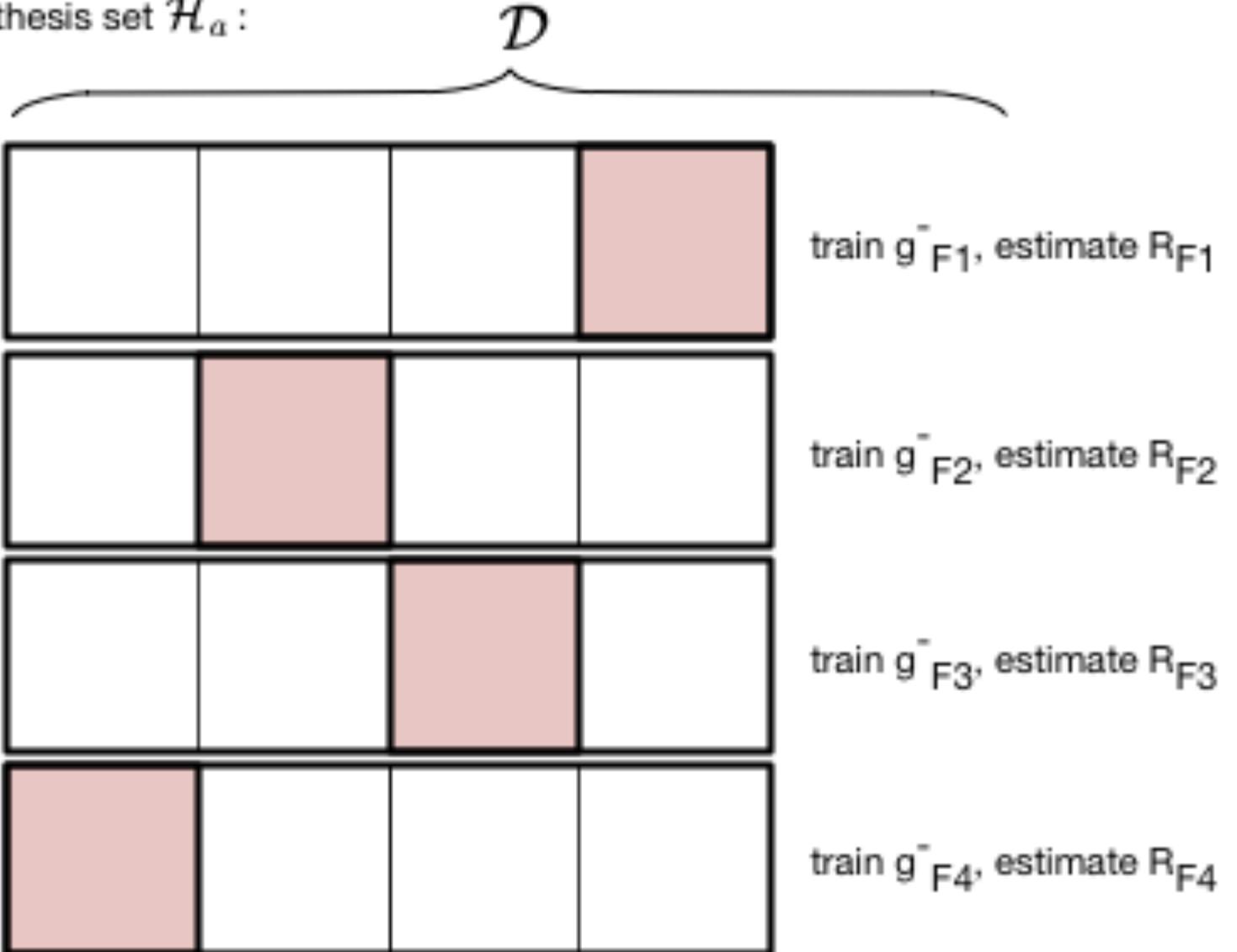
What if we, just by chance had an iffy validation set? What if the data was spread out?



We then do

5. Cross-Validation

For hypothesis set \mathcal{H}_a :

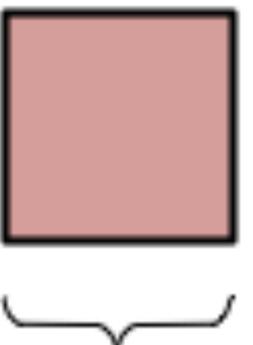


Calculate total error or risk over folds:

$$R_{CV} = \frac{R_{F1} + R_{F2} + R_{F3} + R_{F4}}{4}$$

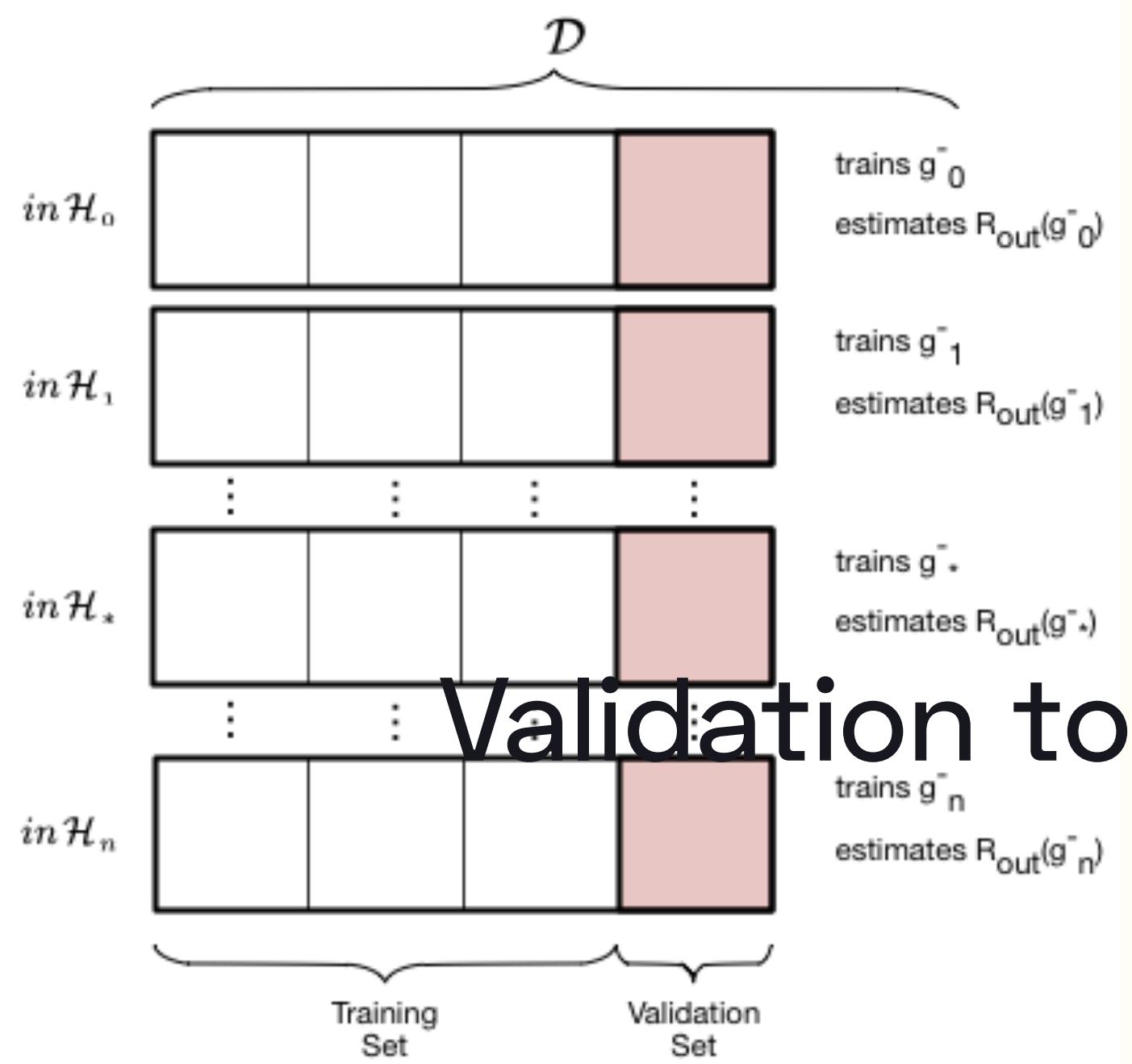
For hypothesis set \mathcal{H}_a , report R_{CV}

Univ.AI

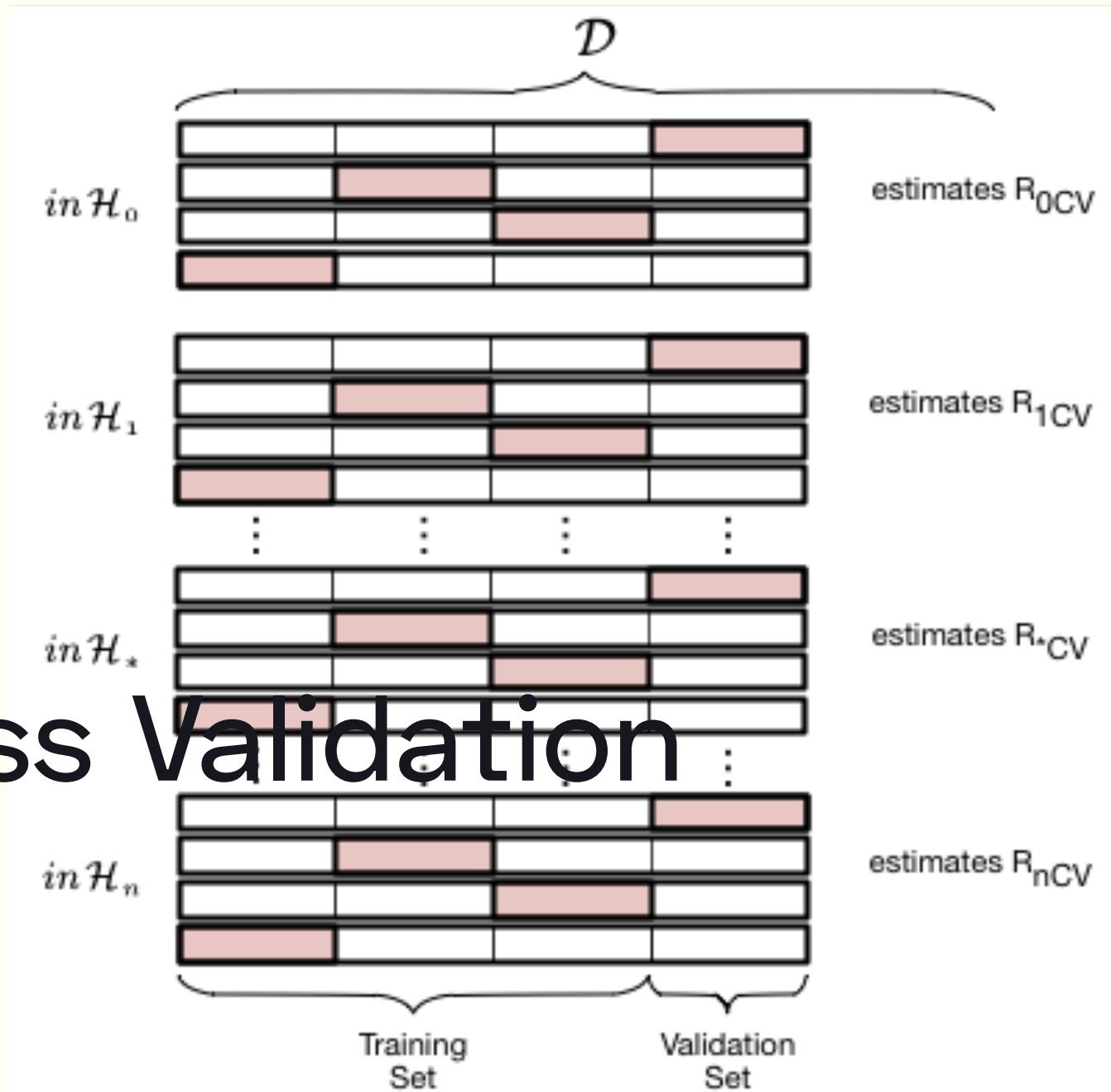


Cross Validation

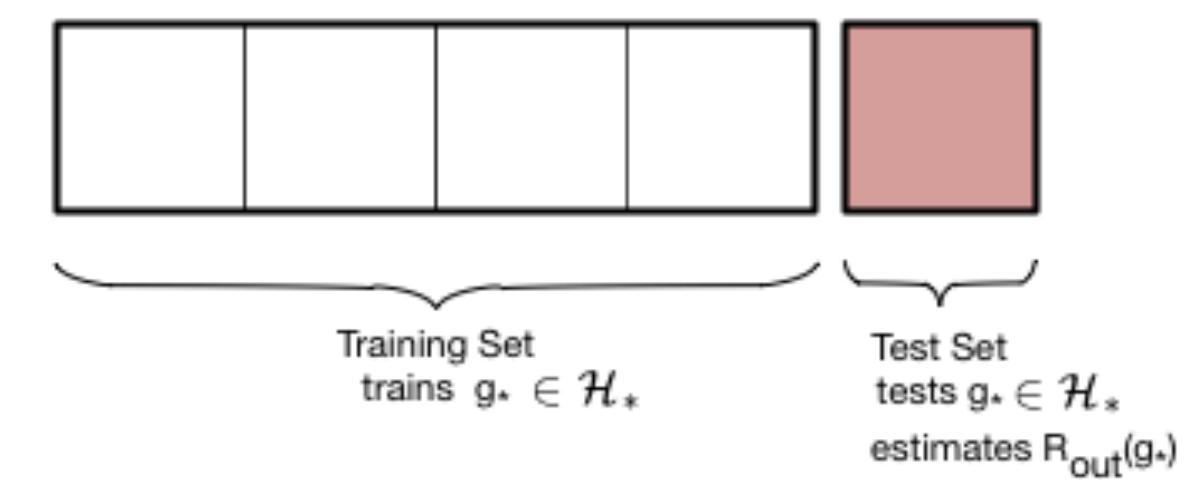
Key Idea: Repeat the validation process on different pieces of left out data. Make these left-out parts not overlap so that the risks/errors/mse calculated on each are not correlated. Average the multiple validation errors. Use this error estimate like before...to choose the right complexity.



pick \mathcal{H}_* with lowest $R_{out}(\bar{g}_*)$, then retrain in \mathcal{H}_* on entire set

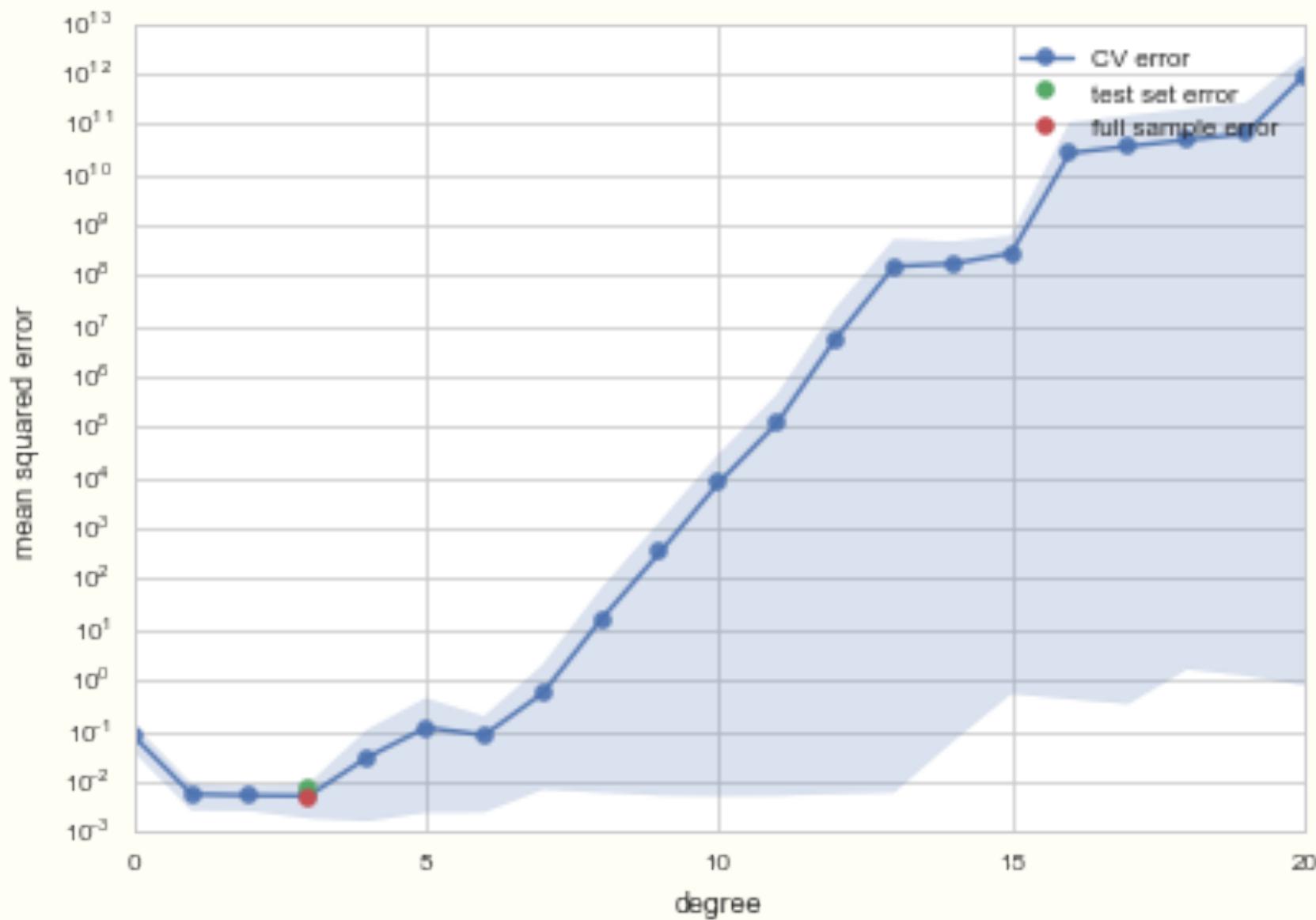


pick \mathcal{H}_* with lowest R_{CV} , then retrain in \mathcal{H}_* on entire set



CROSS-VALIDATION

- robust to outlier validation set
- allows for computing standard deviation on risk estimates
- validation process as one that estimates R_{out} directly, on the validation set. It's critical use is in the model selection process.
- once you do that you can estimate R_{out} using the test set as usual, but now you have also got the benefit of a robust average and error bars.
- key subtlety: in the risk averaging process, you are actually averaging over different g^- models, with different parameters.



Consider instead choosing a Hypothesis set that includes very complex models, and then find a way to subset this set. This method is called

B. Regularization

REGULARIZATION

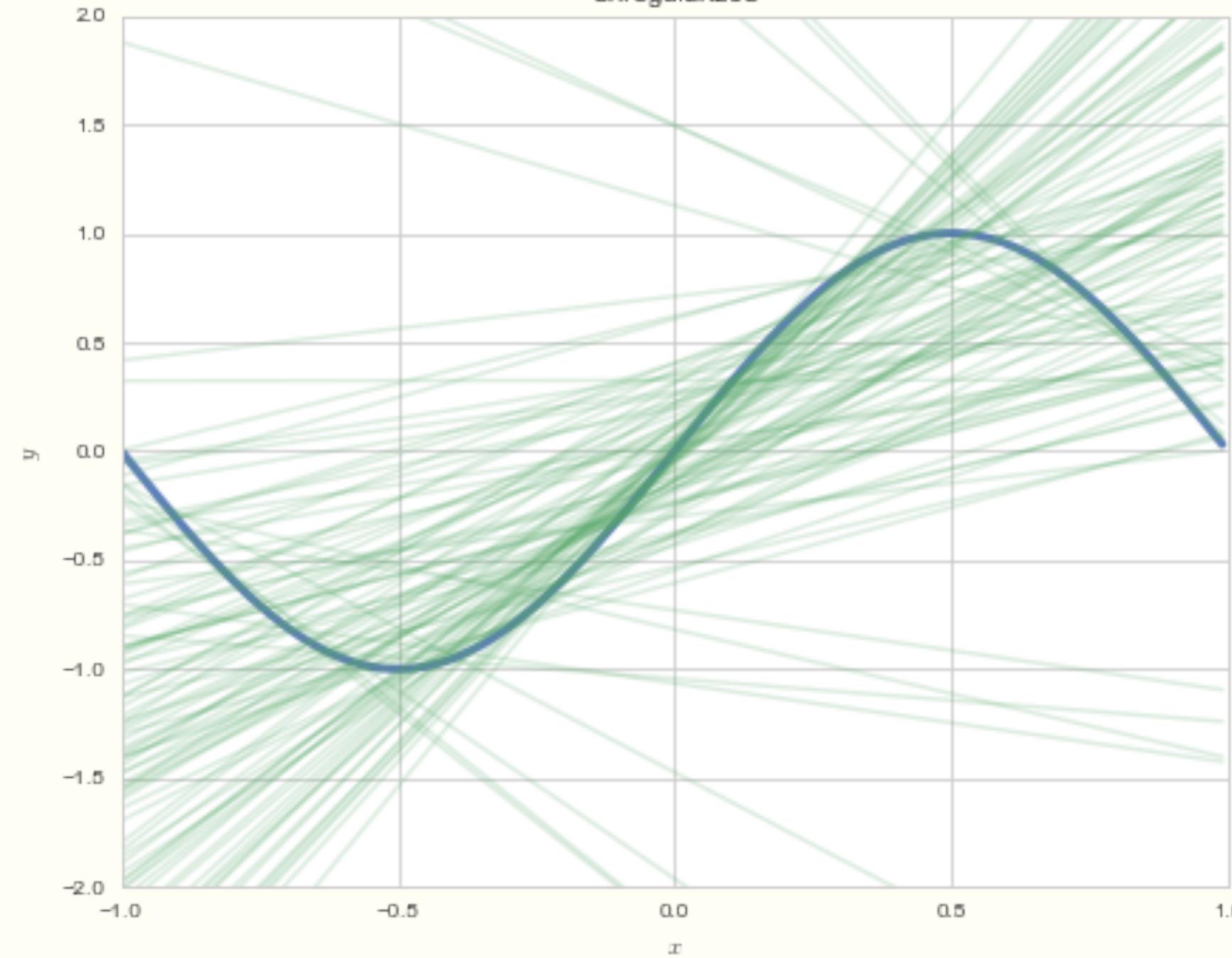
Keep higher a-priori complexity and impose a

complexity penalty

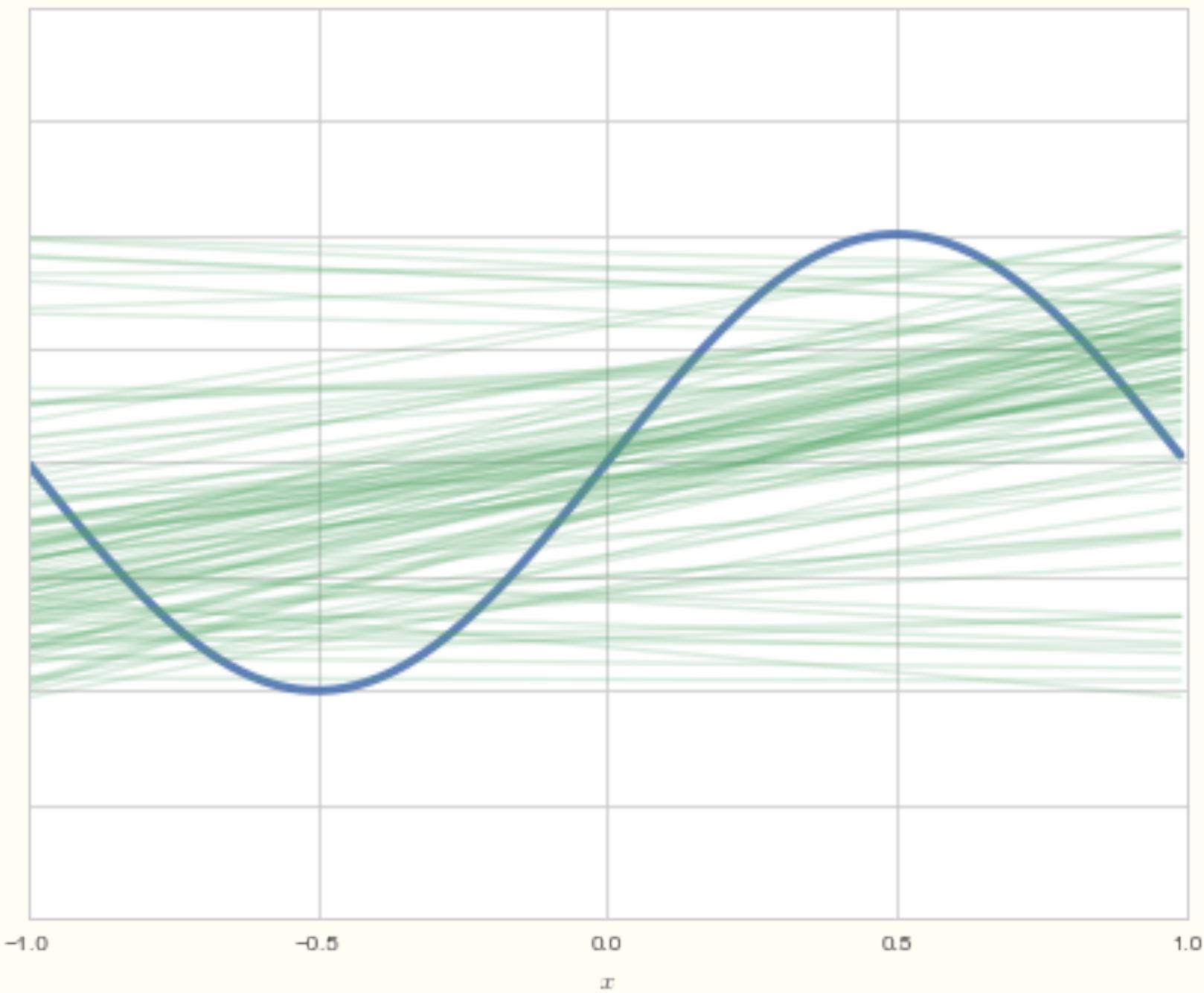
on risk instead, to choose a SUBSET of \mathcal{H}_{big} .

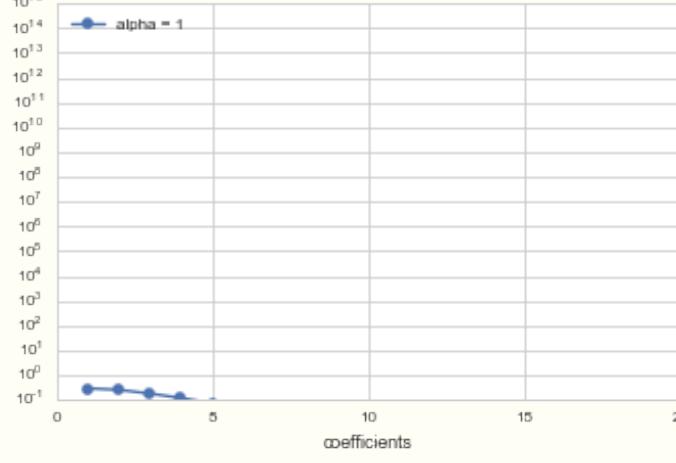
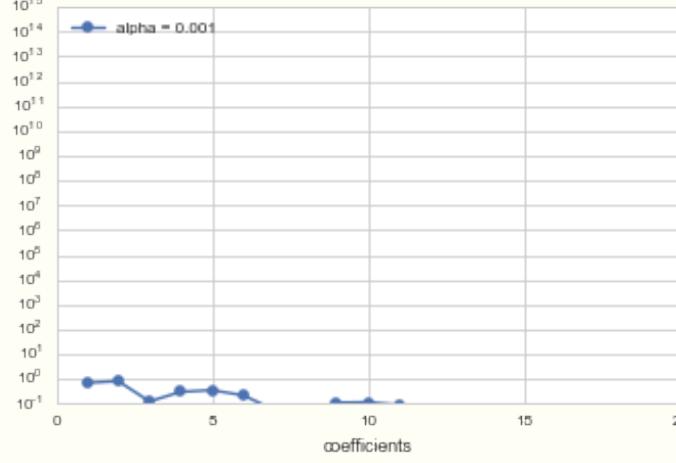
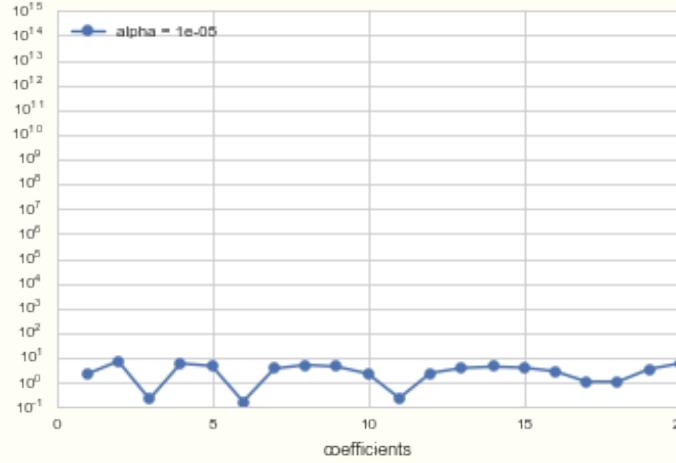
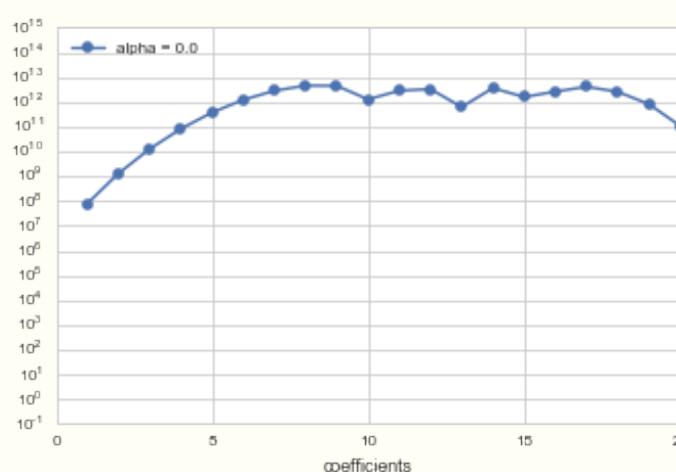
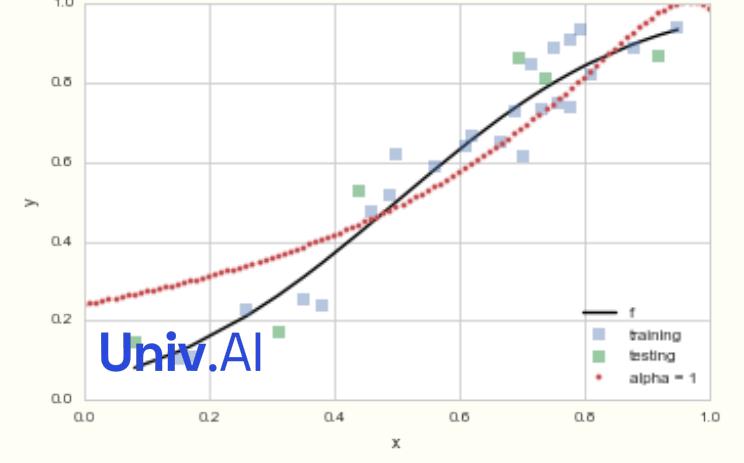
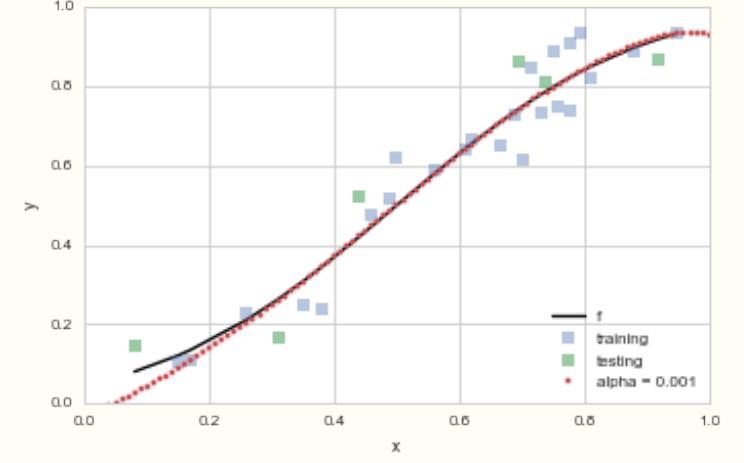
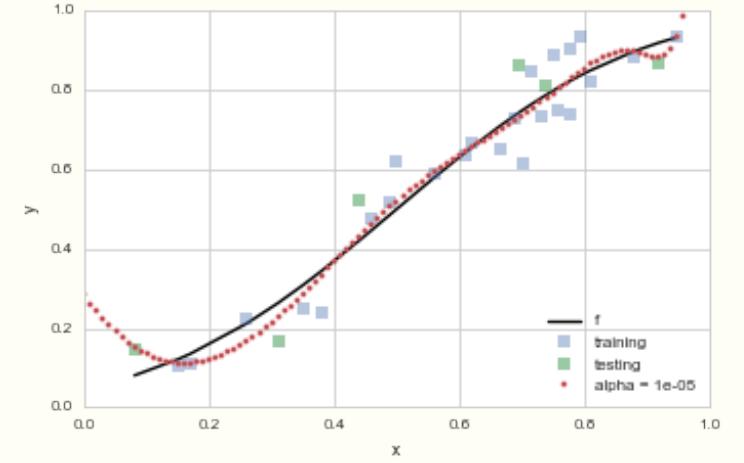
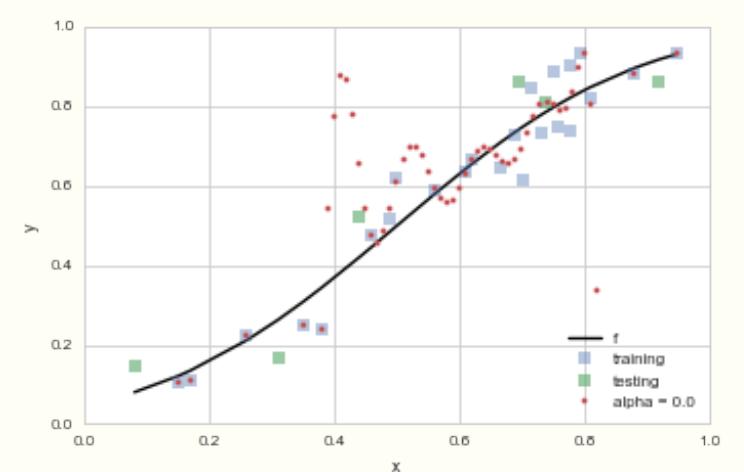
We'll make the coefficients small: $\sum_{i=0}^m \theta_i^2 < C$.

Unregularized



Regularized with $\alpha = 0.2$





ok so now how do we do

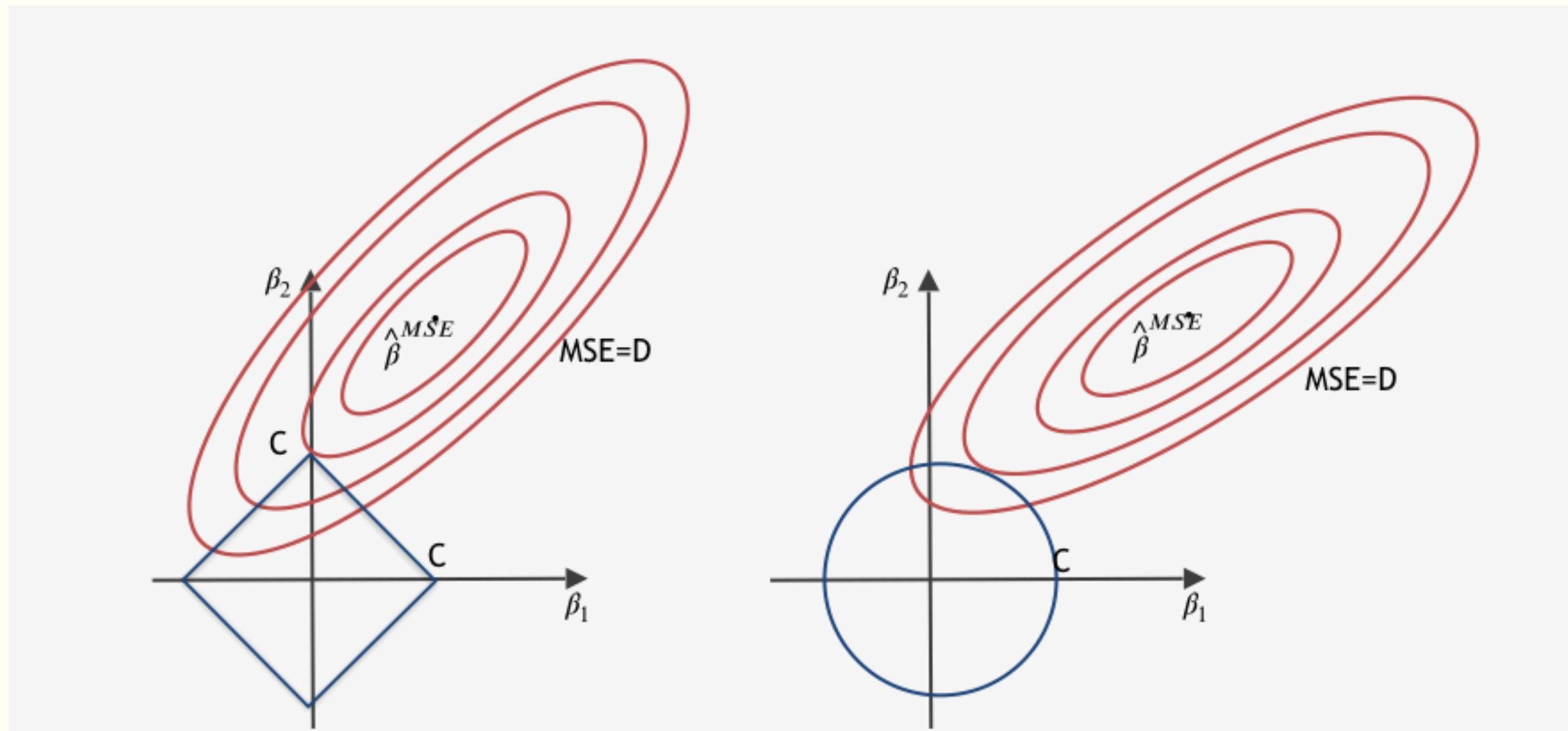
REGULARIZATION

$$\mathcal{R}(h_j) = \sum_{y_i \in \mathcal{D}} (y_i - h_j(x_i))^2 + \lambda \sum_{i=0}^j \theta_i^2.$$

As we increase λ , coefficients go towards 0.

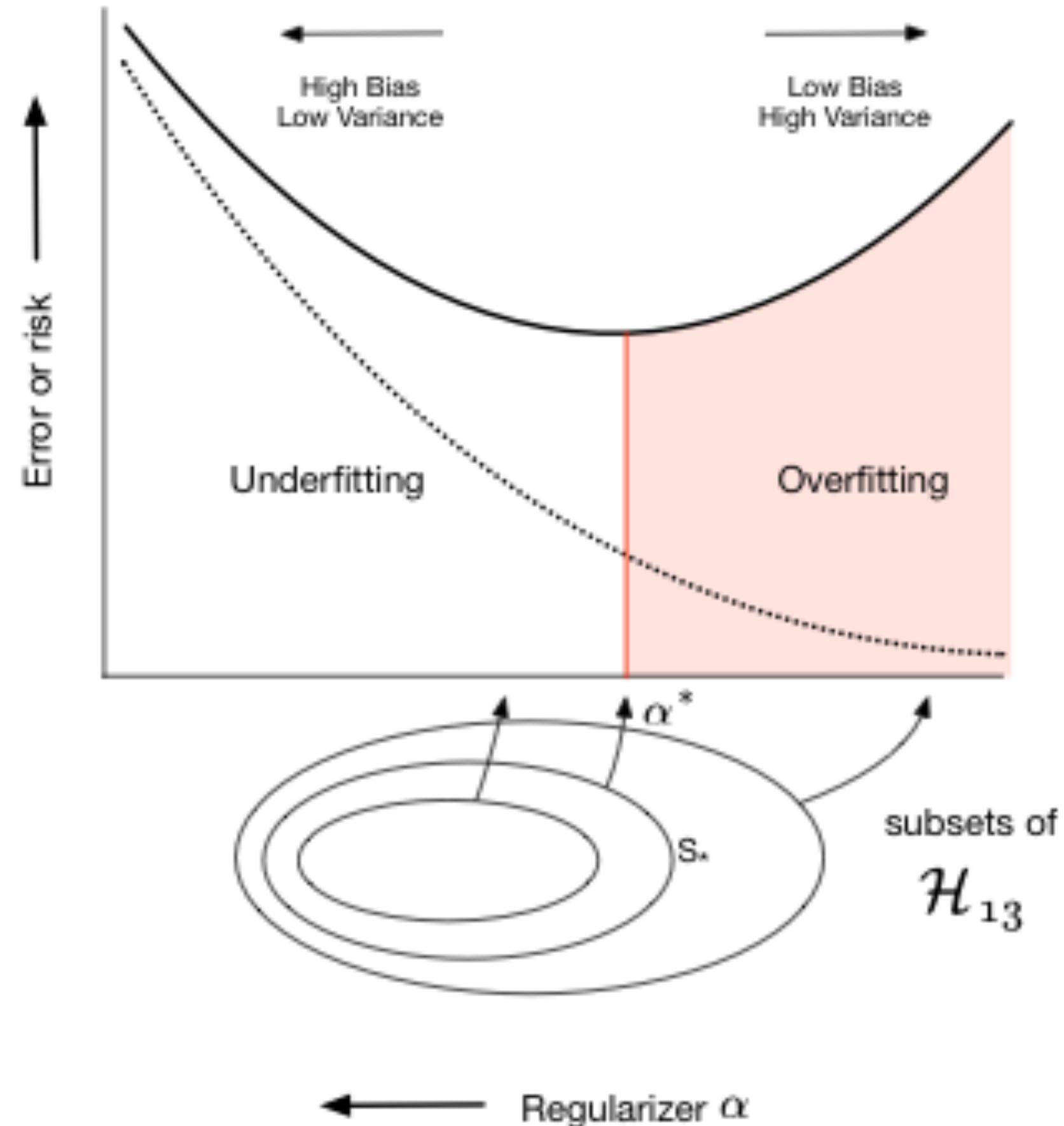
Lasso uses $\lambda \sum_{i=0}^j |\theta_i|$, sets coefficients to exactly 0.

Lasso vs Ridge Geometry

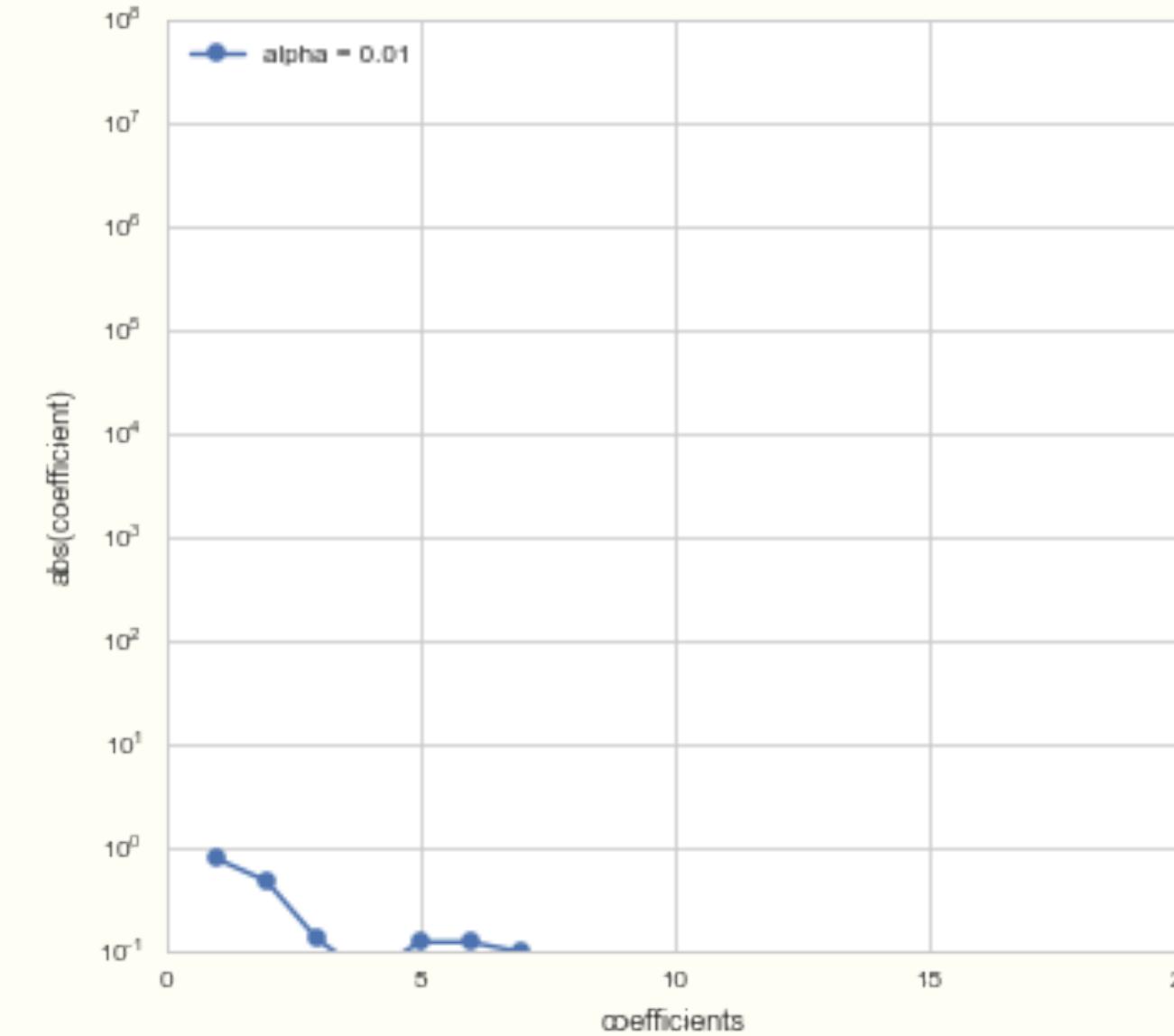
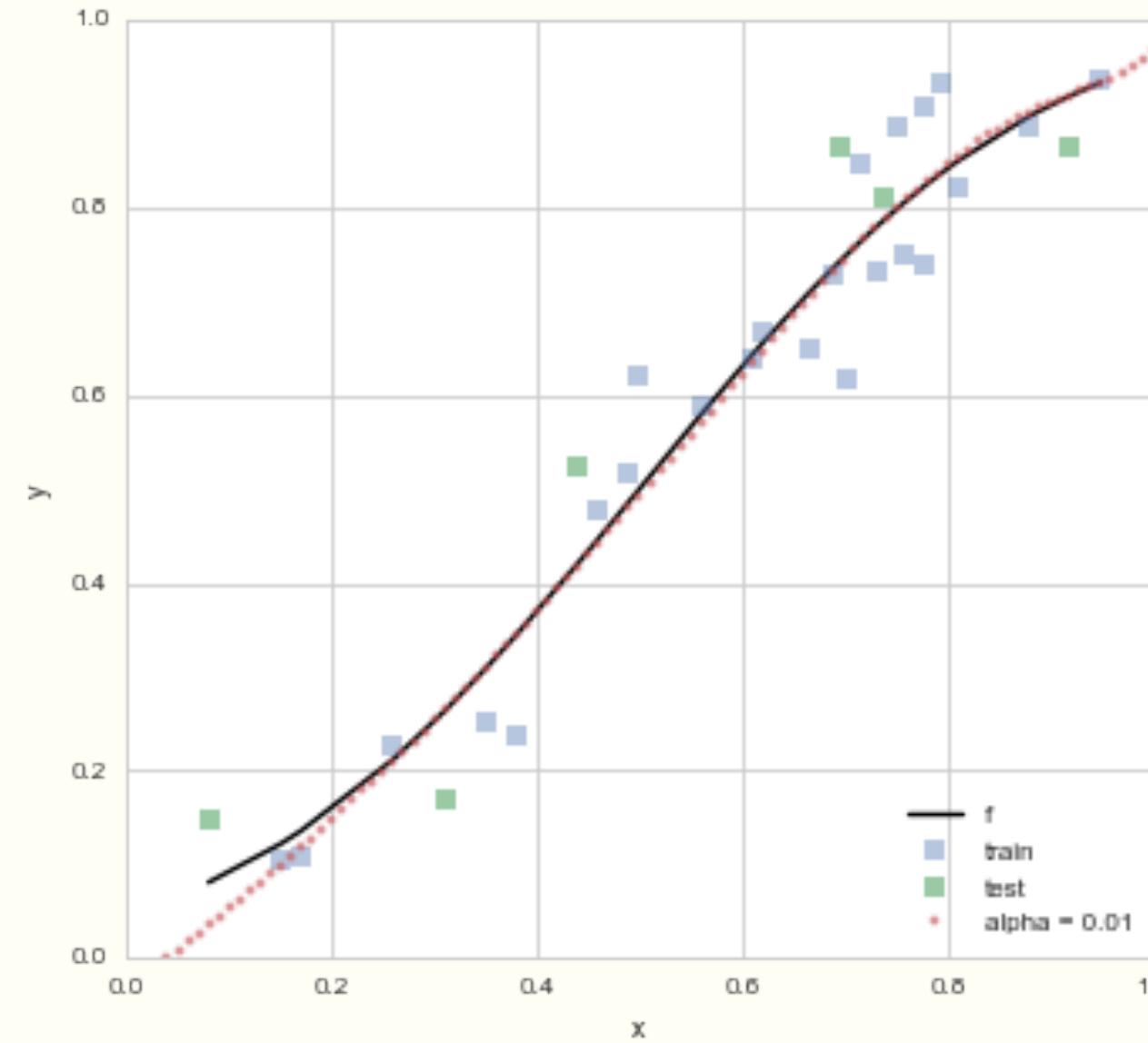


Structural Risk Minimization

- Regularization is a subsetting now,
- of a complex hypothesis set.
- If you subset too much, you underfit
- but if you do not do it enough, you overfit



Regularization with Cross-Validation



C. Lots of features

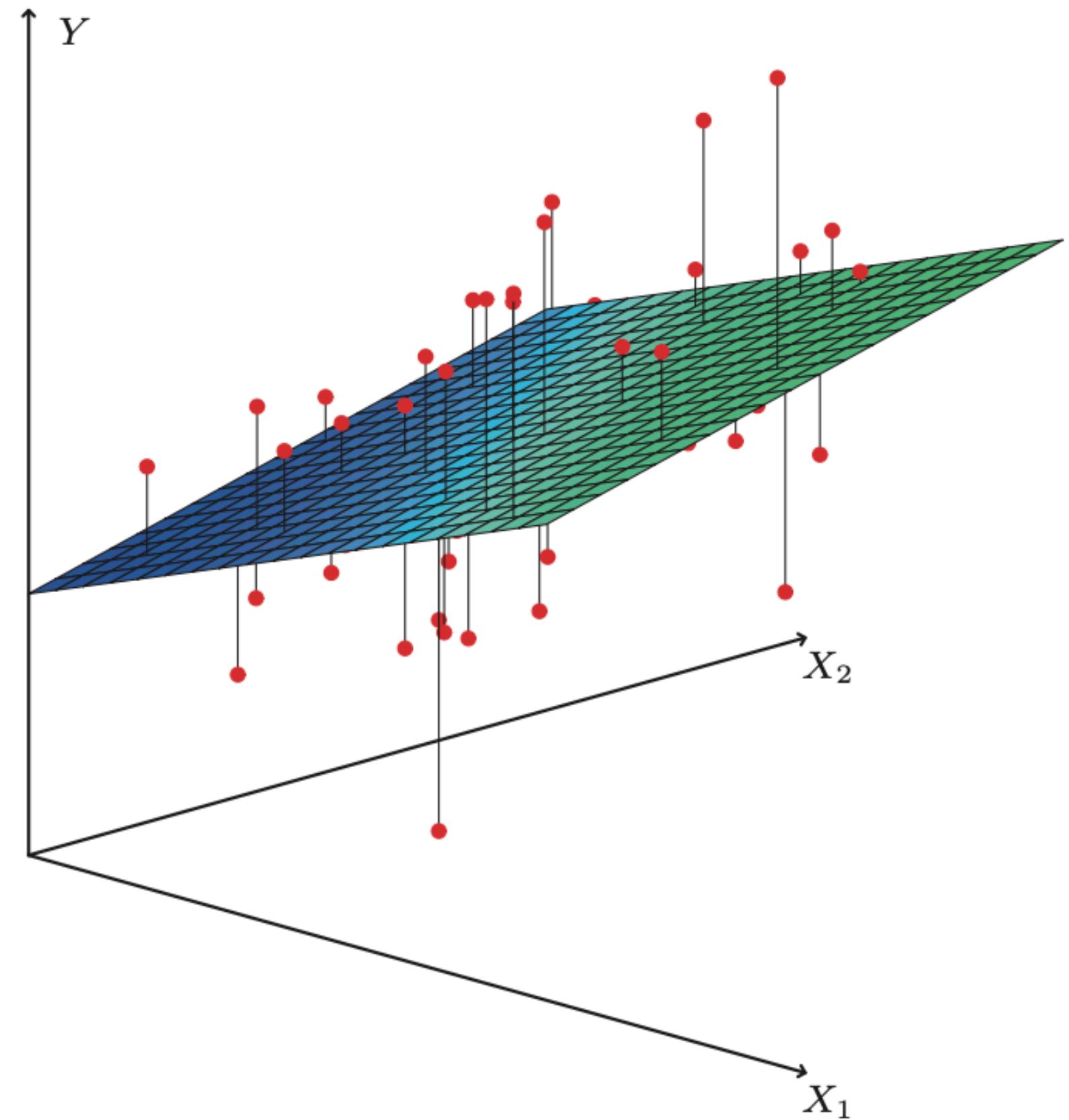
AKA: features are not just polynomial powers

Powers in a polynomial as separate features

- imagine you put x, x^2, x^3 and so on as different axes
- if you have a high degree polynomial then this space looks like a high dimensional hypercube
- and in this high dimensional hypercube, you are looking to fit a hyperplane to the data

Here $X_1 = x = X_2 = x^2$, and so on...

So Polynomial Regression is linear regression in a high dimensional space.



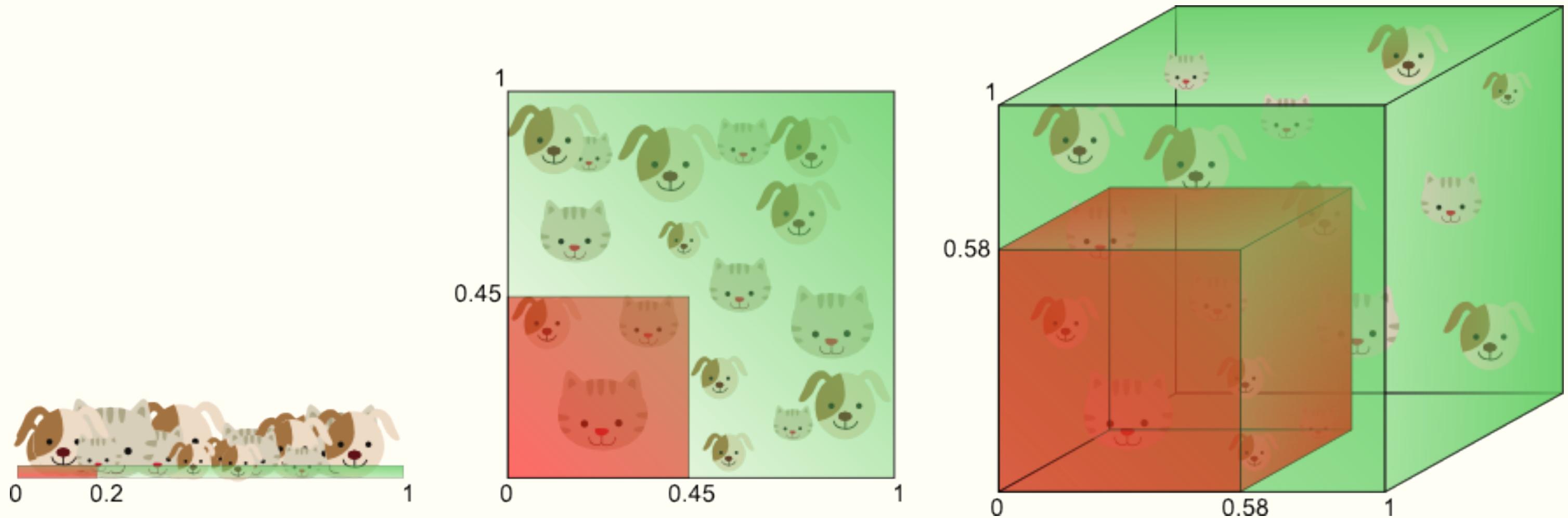
Curse of Dimensionality

As you can see, the number of features can balloon. In many modern problems: startup with few customers but lots of data on them, there are already more predictors than members in your sample. And you probably want polynomials to model interactions and higher order behavior. You get tons and tons of dimensions,

But then :

- data is sparser in higher dimensions
- volume moves to the outside

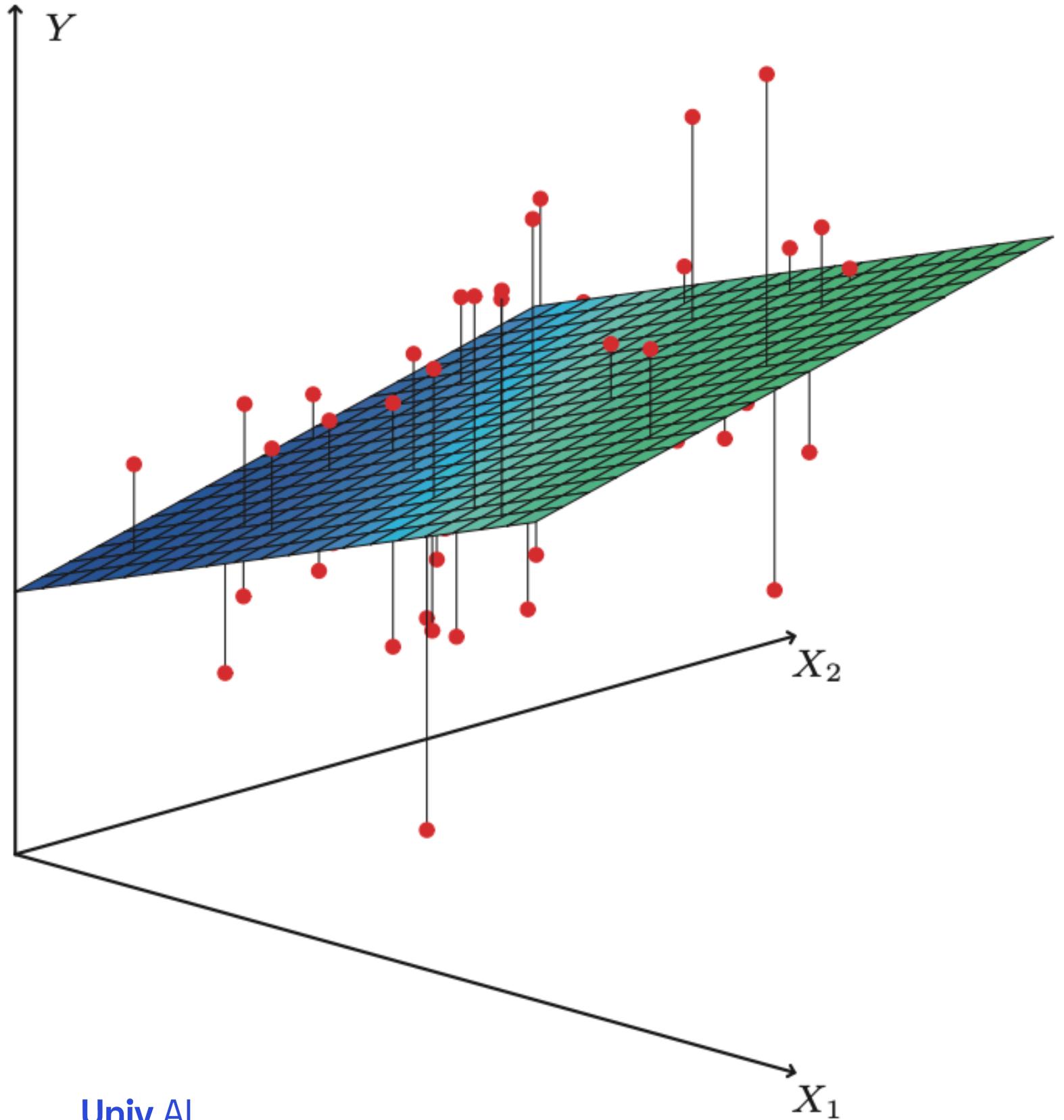
To cover same fractional volume, you need to go bigger on length in higher dims. Data points get sparser.



Overfitting and the curse

- remember dimensionality d in our problems refers to the number of features we have
- each feature (or feature combination which we shall just call a new feature) is a dimension
- thus each member/point of our sample is a point in this feature space, and we have gained a new parameter/slope for this feature
- distance between samples increases: they become sparser. Or you need more points (data) to keep the same distance

What regression looks like now?



See how close the points are to the plane here?

In high dimensions the points are much further since they are sparser.

Thus replacing even one data point by another in a new sample can lead to a radically different regression plane (radically different slopes/weights). This is overfitting.

Thus if we have lots of features, we will have instability, and must use complexity control/regularization.