
Univ.AI

Learning a Model

Part2

Validation and Regularization

Univ.AI

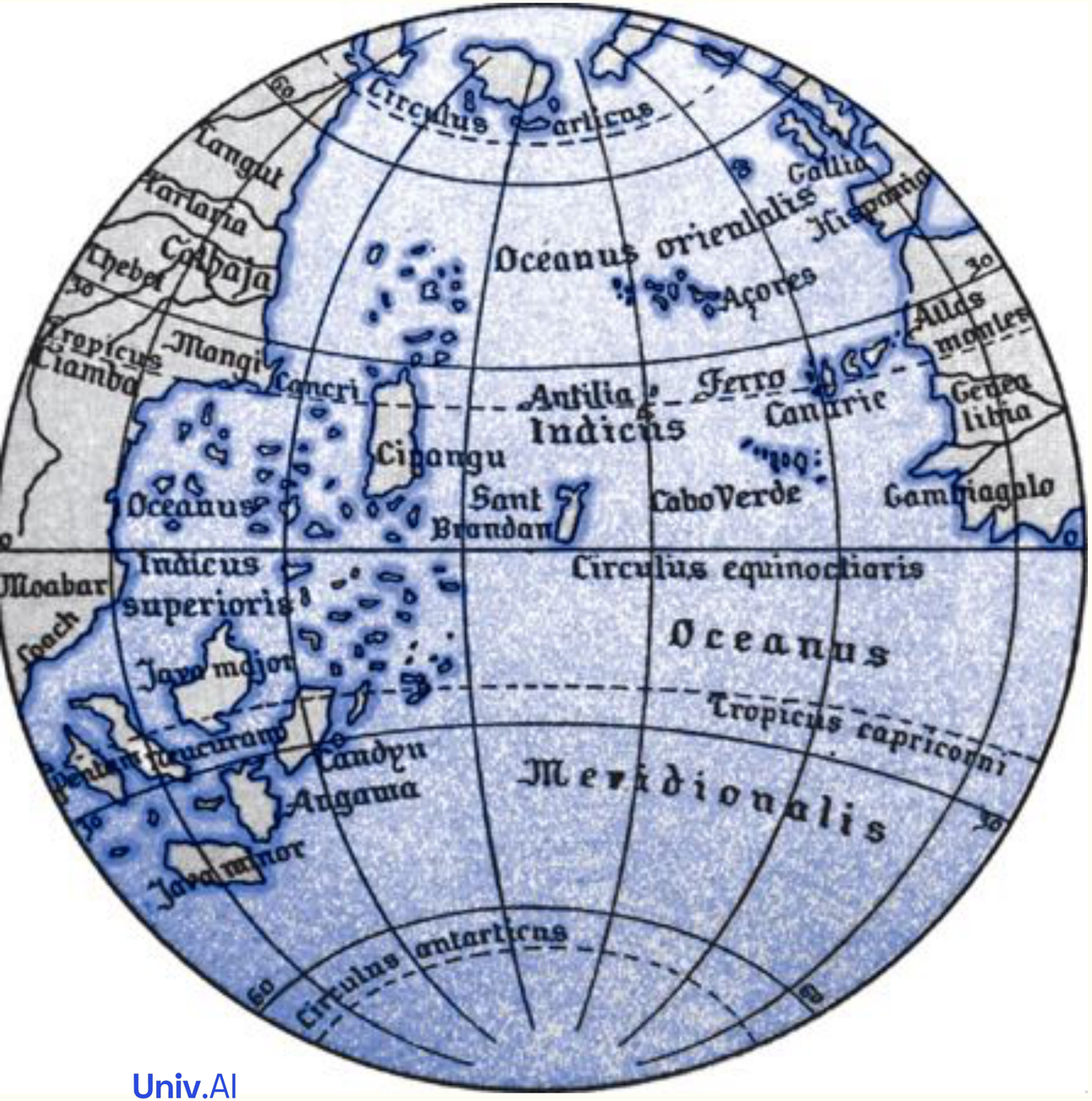
Last Time

1. SMALL World vs BIG World
2. Approximation
3. THE REAL WORLD HAS NOISE
4. Complexity amongst Models
5. Validation

Today

- (0) Recap of key concepts from earlier
- (1) Validation and Cross Validation
- (2) Regularization
- (3) Multiple Features

0. From before



Small World vs Big World

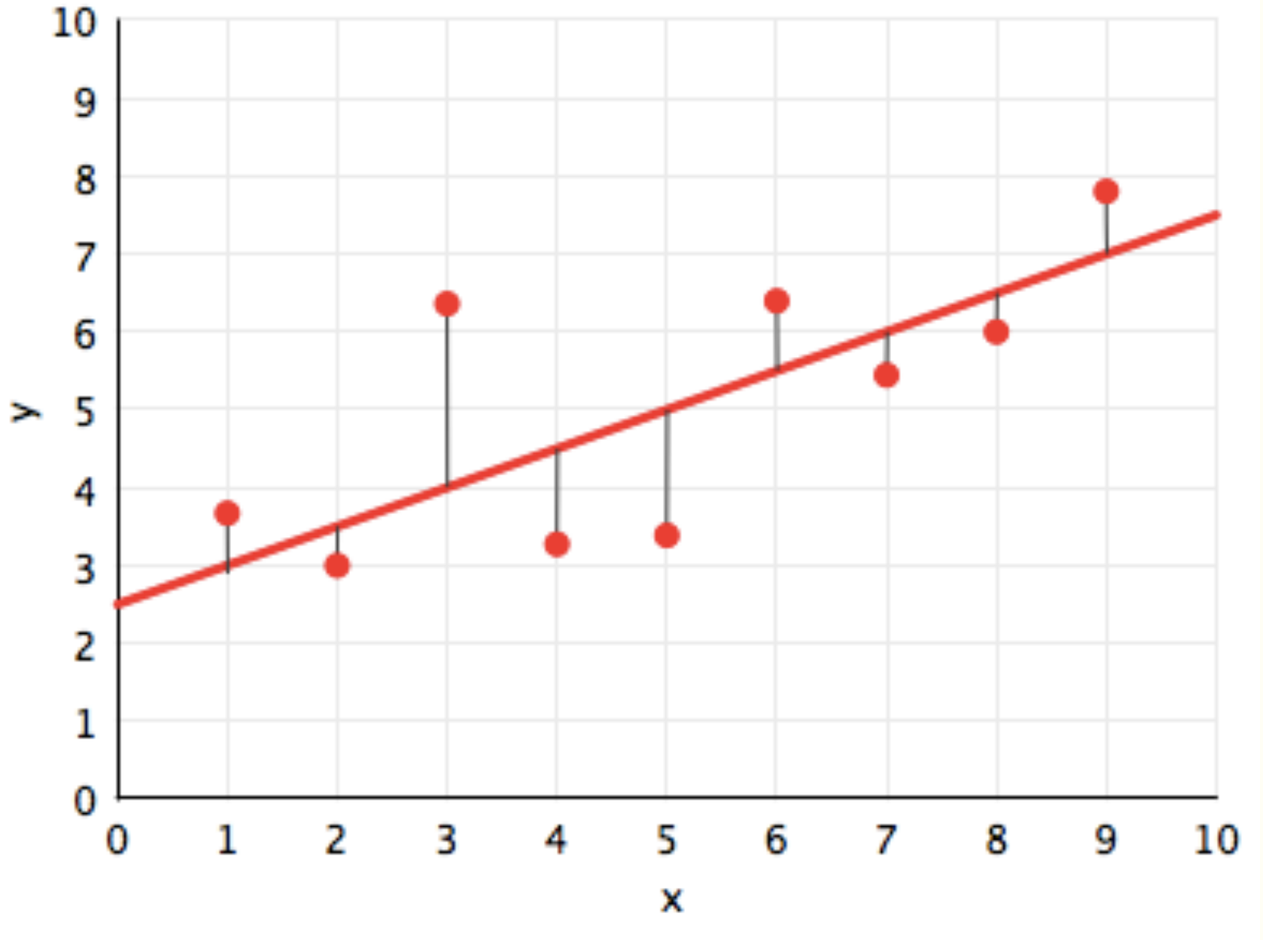
Small World given a map or model of the world, how do we do things in this map?

Answers the question: given a model class (i.e. a Hypothesis space, whats the best model in it). Thus its looking for a particular $h(x)$ in a particular $\mathcal{H}$.

BIG World compares maps or models. Asks: whats the best map?

Compares model spaces. Wants to find the true f(x), or at least the **best** $h(x)$ in the best $\mathcal{H}$ amongst the Hypothesis spaces we test.

Finding the best fit



Minimize distance from the curve

$$h_m(x) = \sum_{i=0}^{m} \theta_i x^i \in \mathcal{H}_m$$

Compute the risk/cost/error/distance:

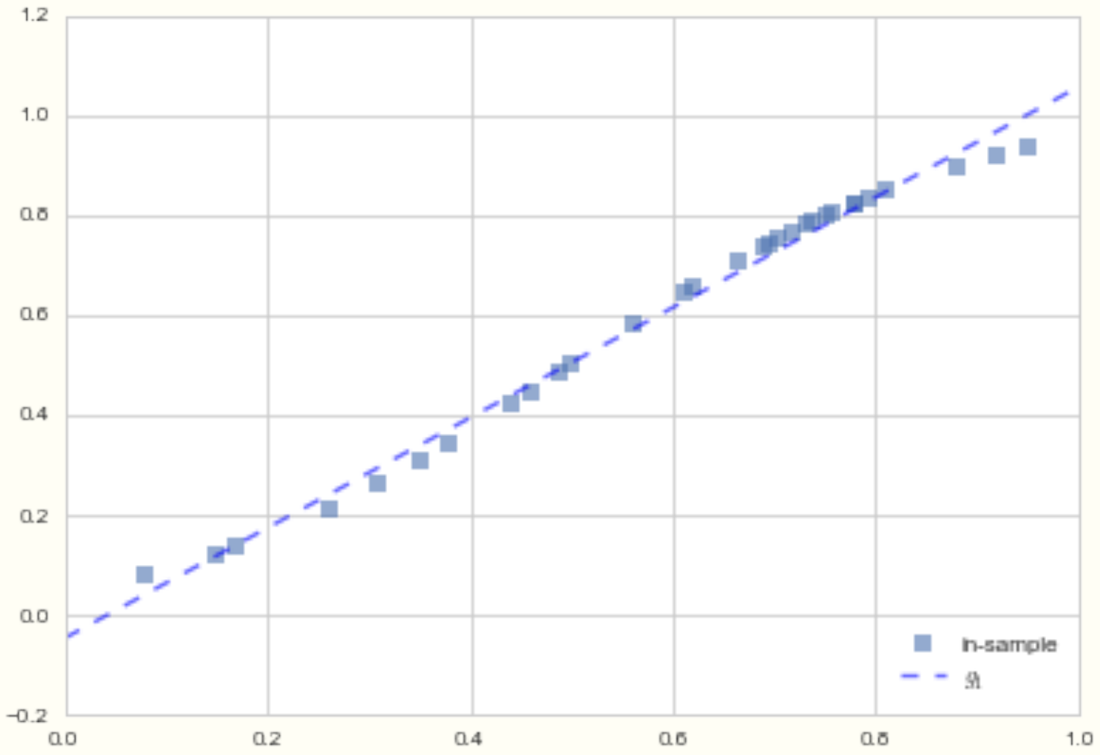
$$R_{\mathcal{D}}(h_m(x)) = \frac{1}{N} \sum_{y_i \in \mathcal{D}} (y_i - h_m(x_i))^2$$

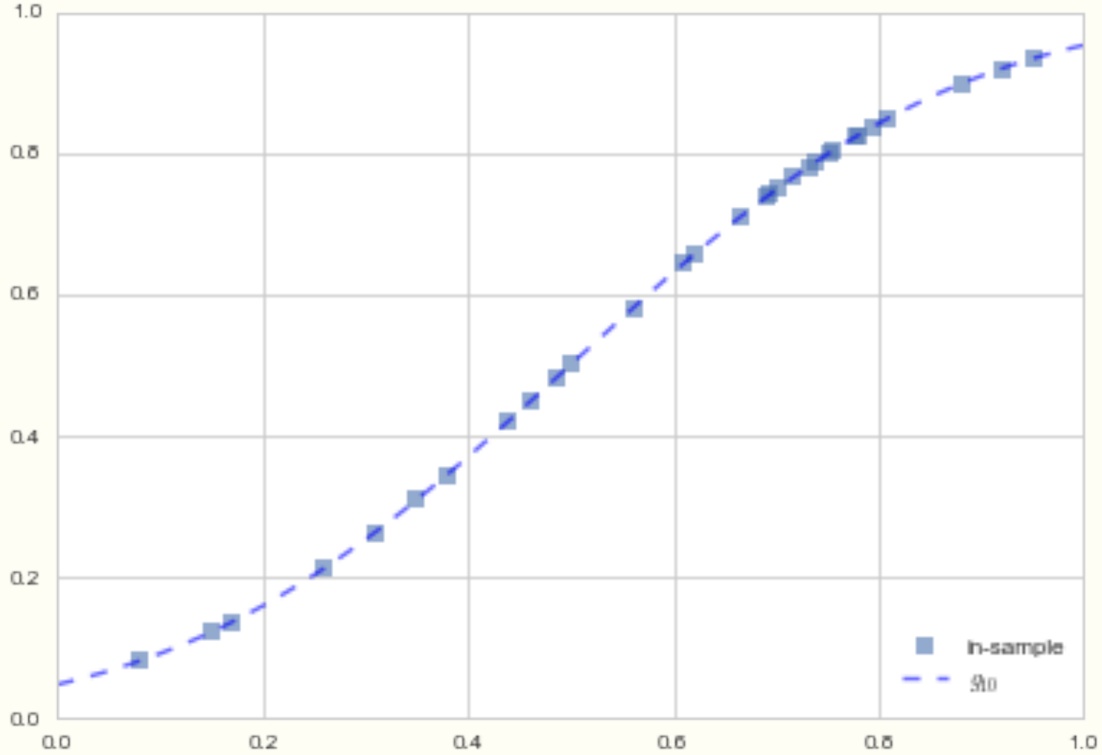
Now minimize squared distance from the curve.

$$g_m(x) = \arg \min_{h_m(x) \in \mathcal{H}_m} R_{\mathcal{D}}(h_m(x)).$$

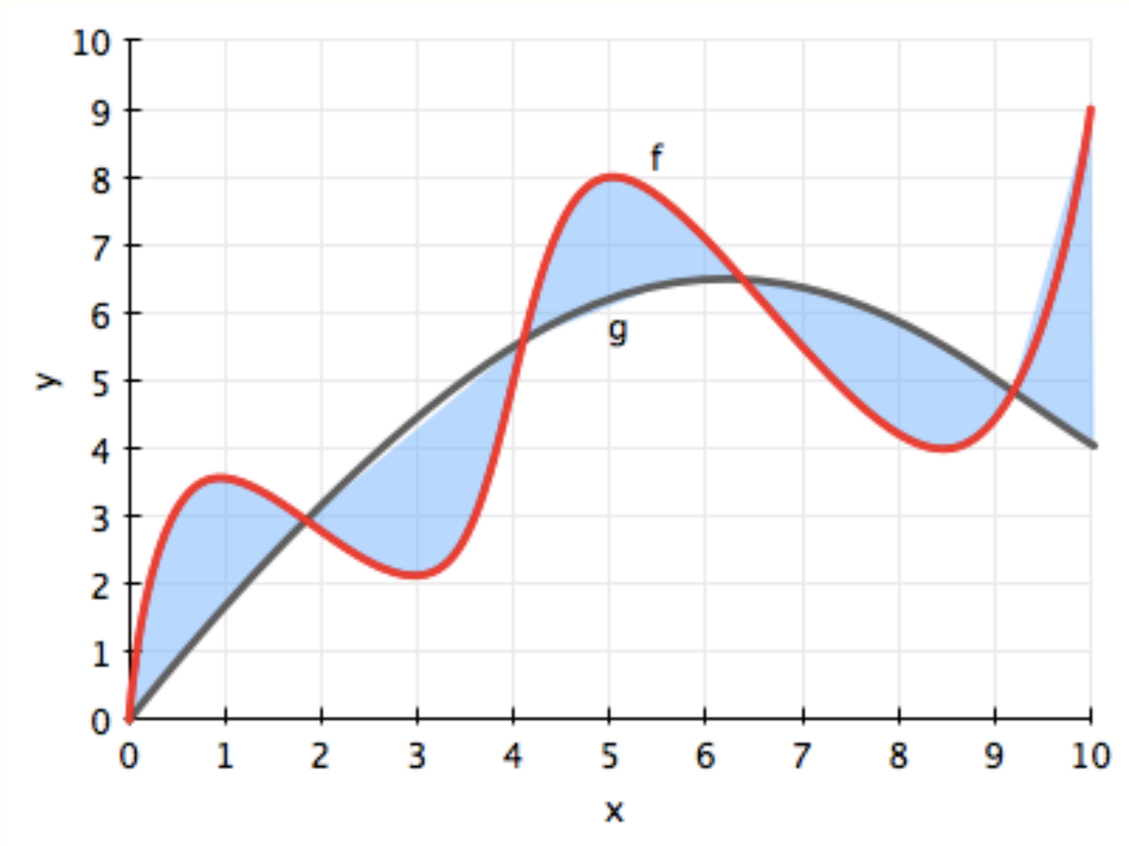
Read: find the best fit curve g_m that minimizes risk amongst all the curves in H_m.

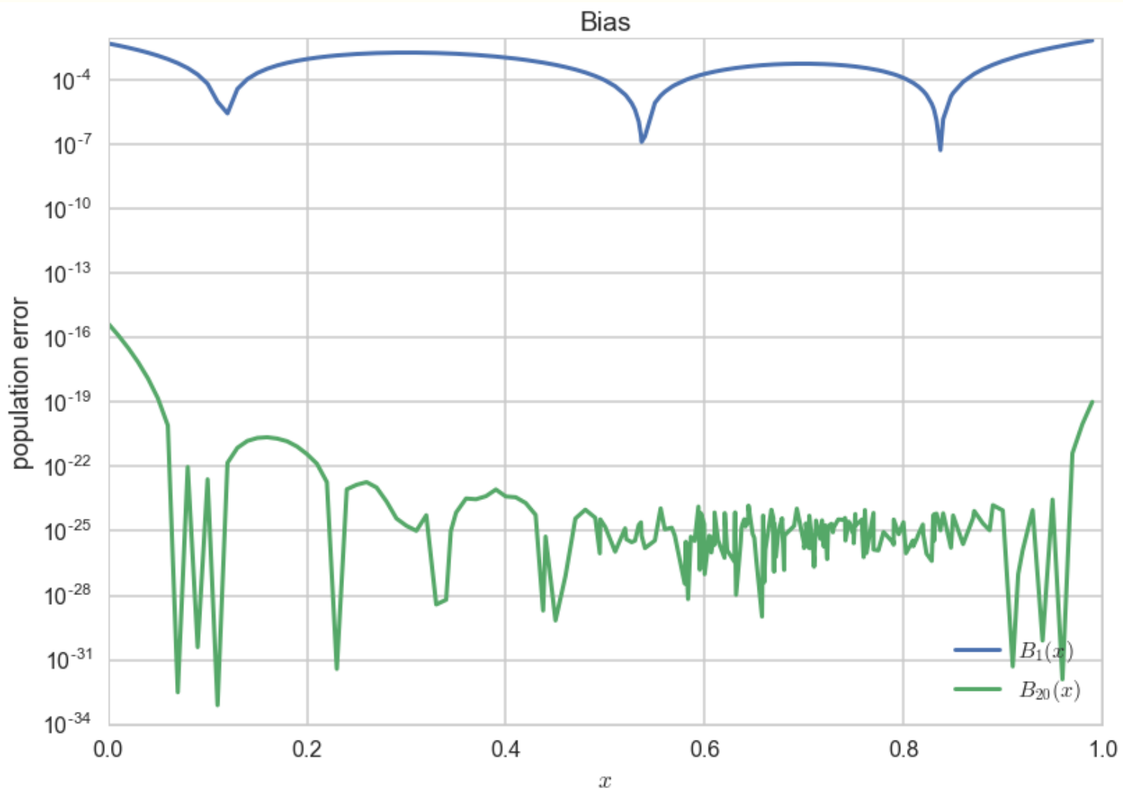
NO NOISE: A sample of 30 points of data. Which fit is better? Line in $\mathcal{H}_1$ or curve in $\mathcal{H}_{20}$?



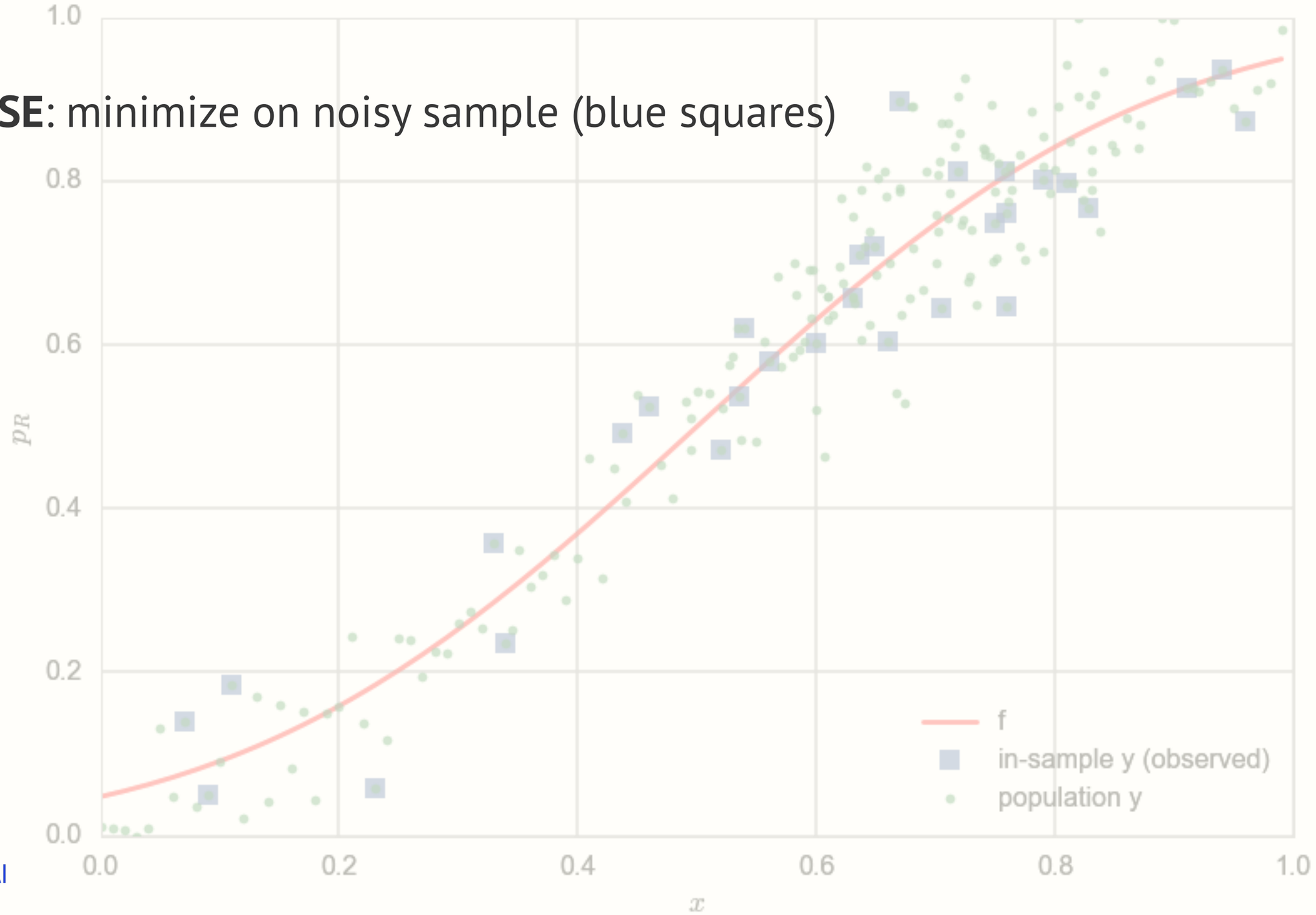


Bias or Mis-specification Error

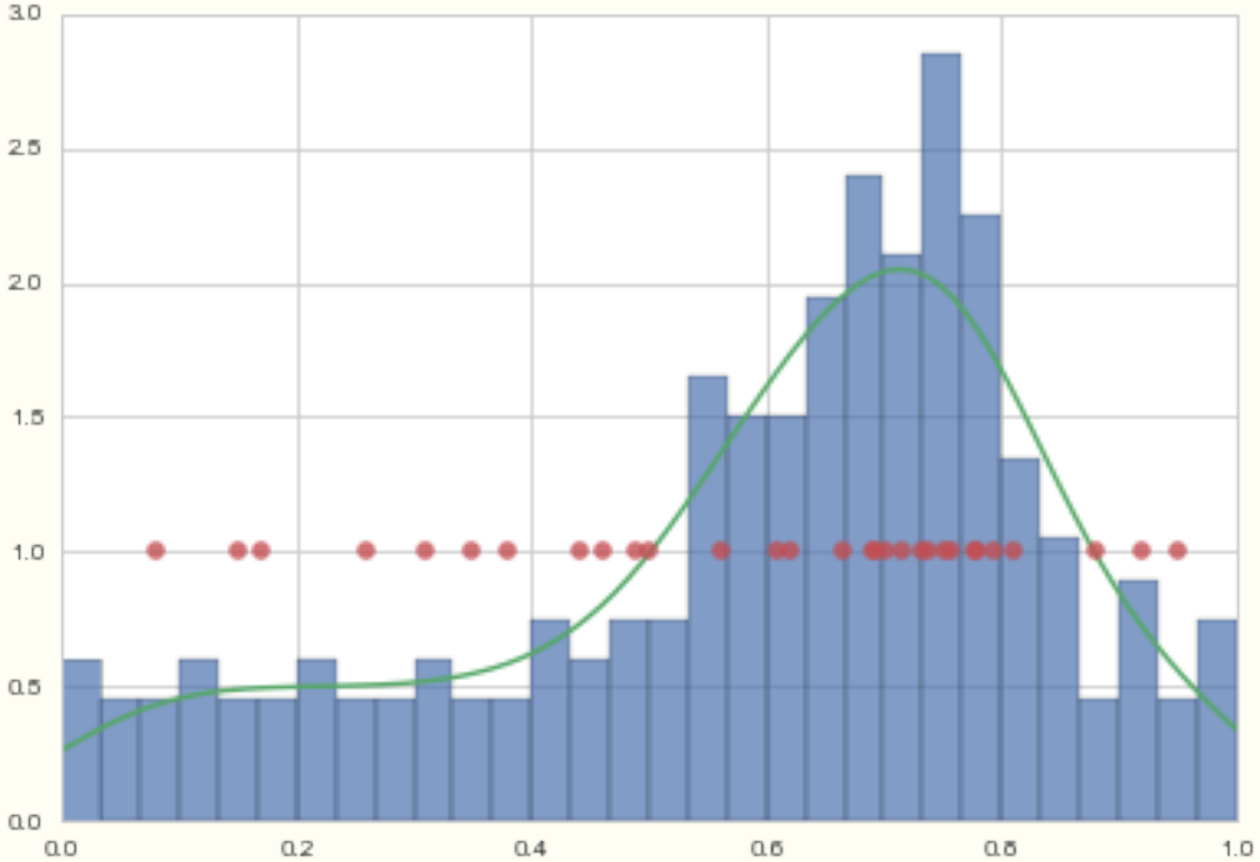




NOISE: minimize on noisy sample (blue squares)



Statement of the Learning Problem



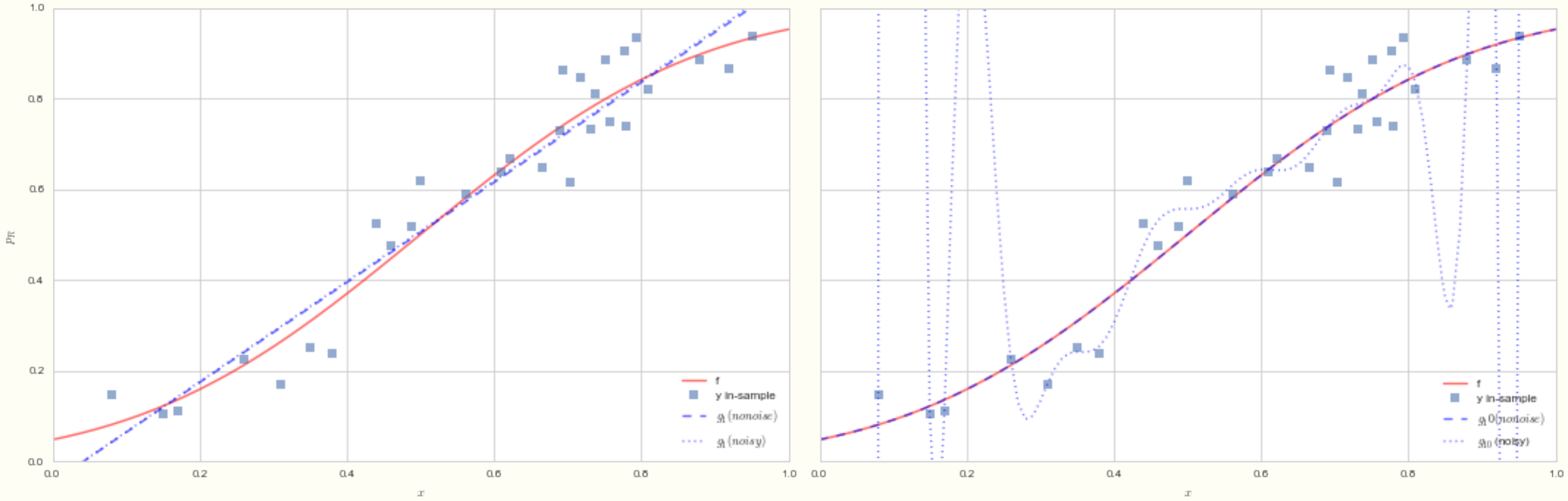
The sample must be representative of the population!

$A : R_{\mathcal{D}}(g)\ smallest\ on\ \mathcal{H}$
$B : R_{out}(g) \approx R_{\mathcal{D}}(g)$

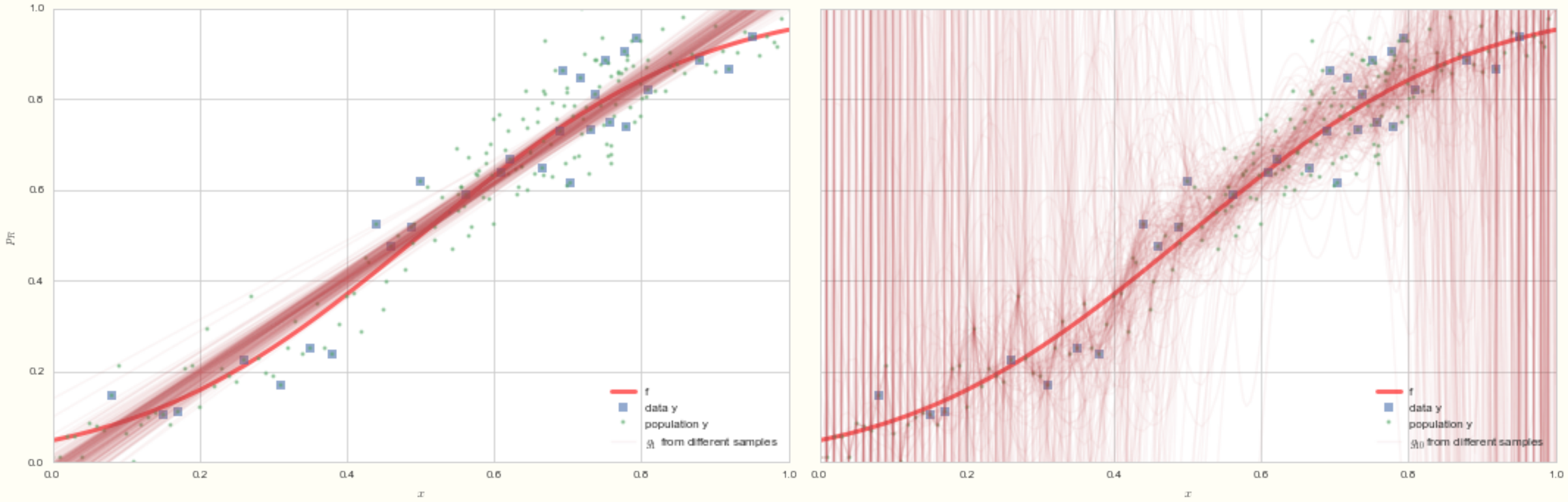
A: In-sample risk is small
B: Population, or out-of-sample risk is WELL estimated by in-sample risk. Thus the out of sample risk is also small.

Which fit is better now?

The line or the curve?



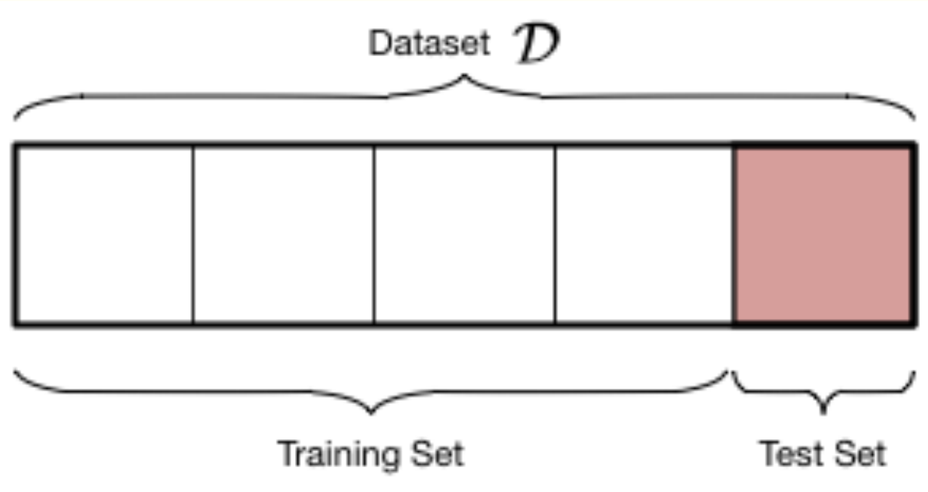
UNDERFITTING (Bias) vs OVERFITTING (Variance)

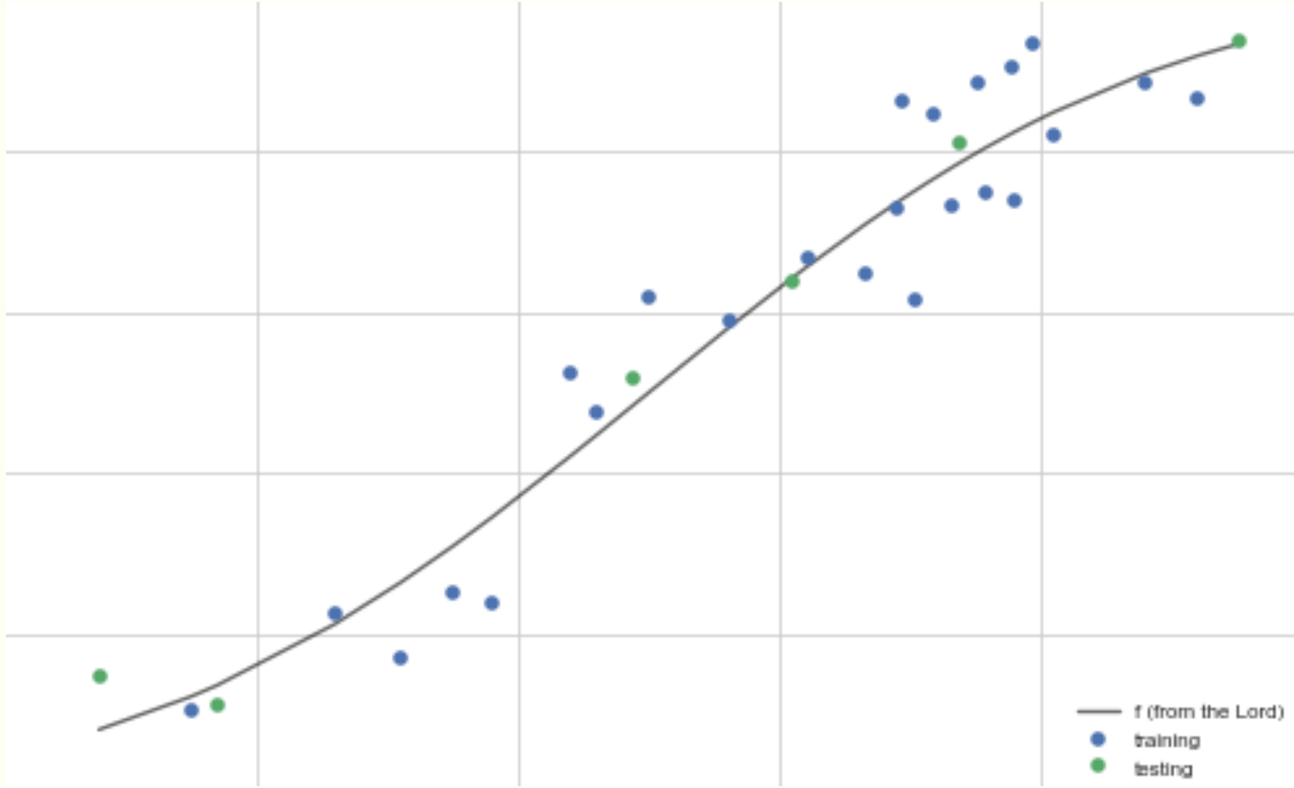


How do we estimate

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

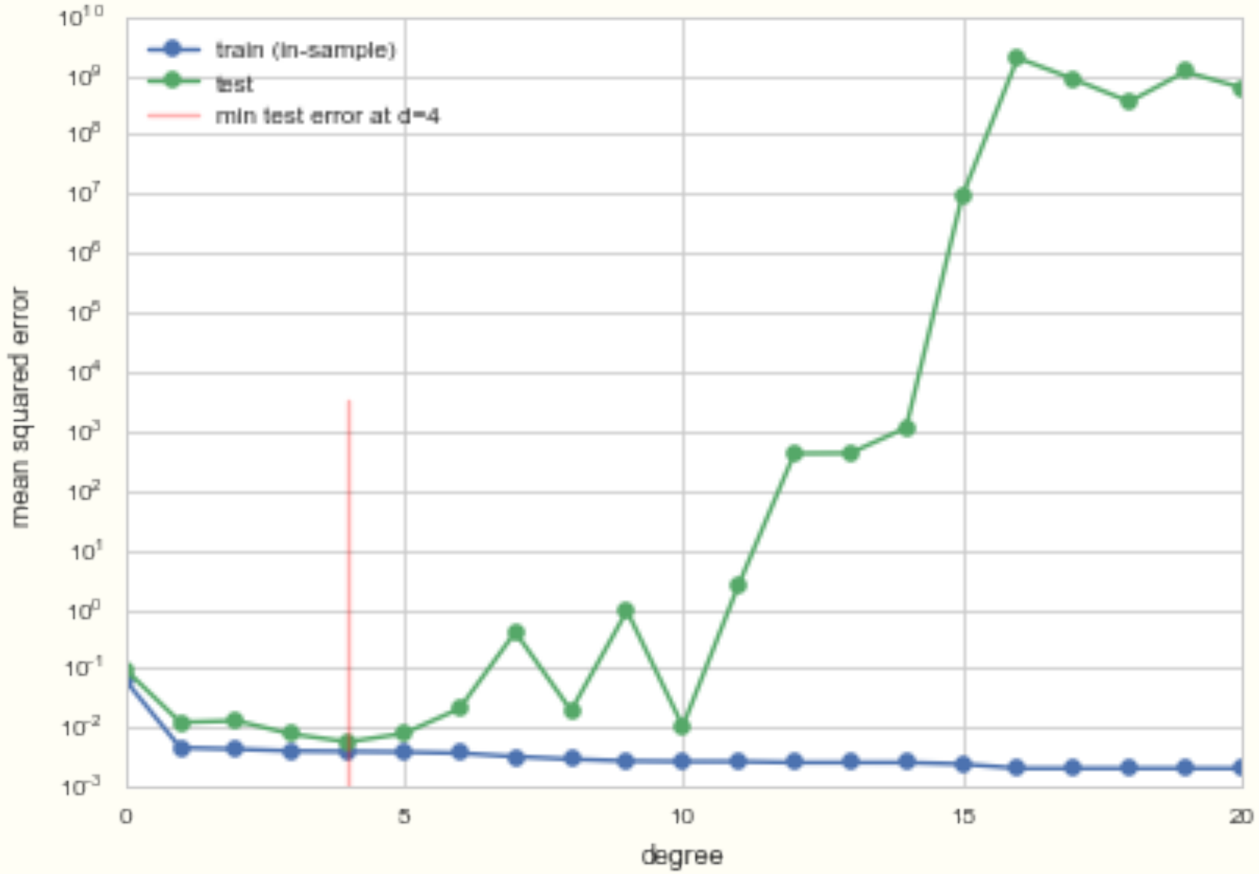
TRAIN AND TEST



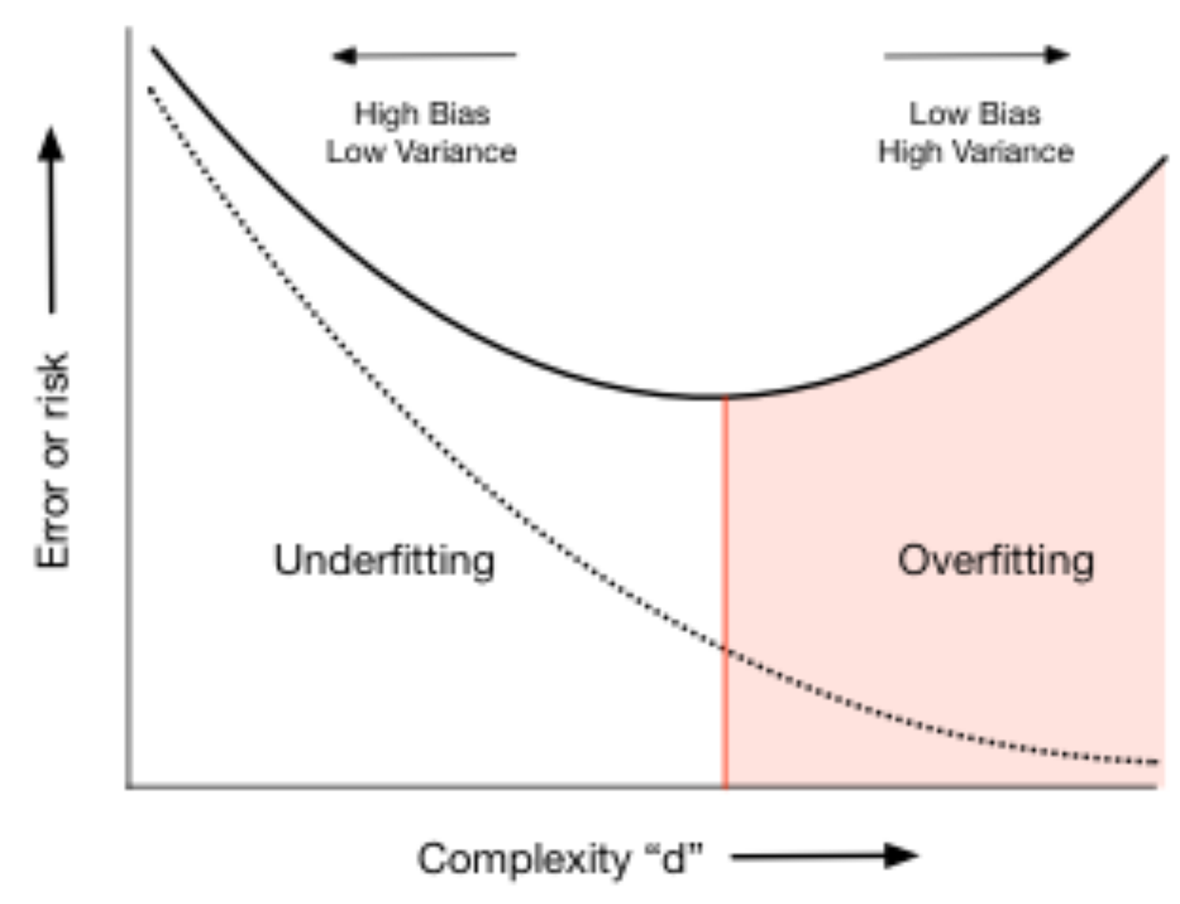


MODEL COMPARISON: A Large World approach

- 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



1. Validation and Cross Validation

Do we still have a test set?

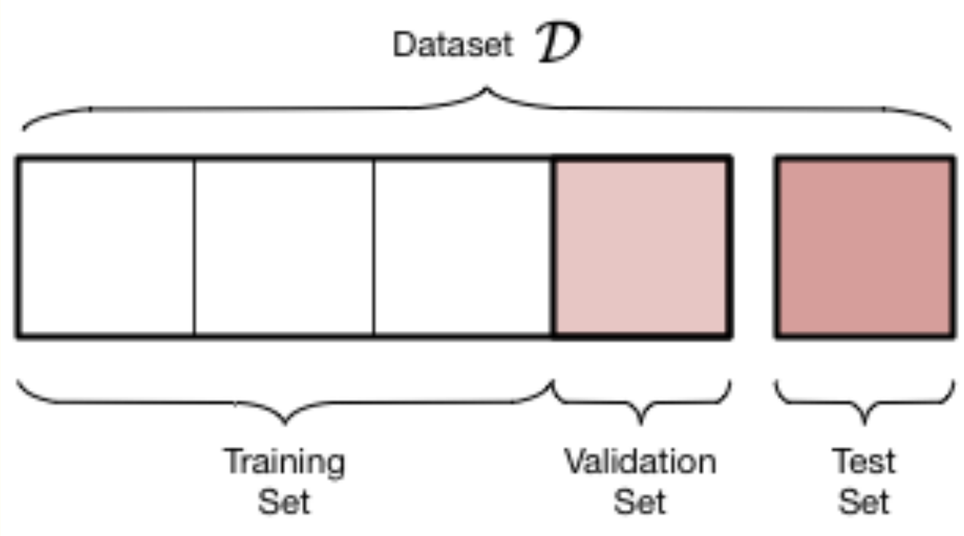
Trouble:

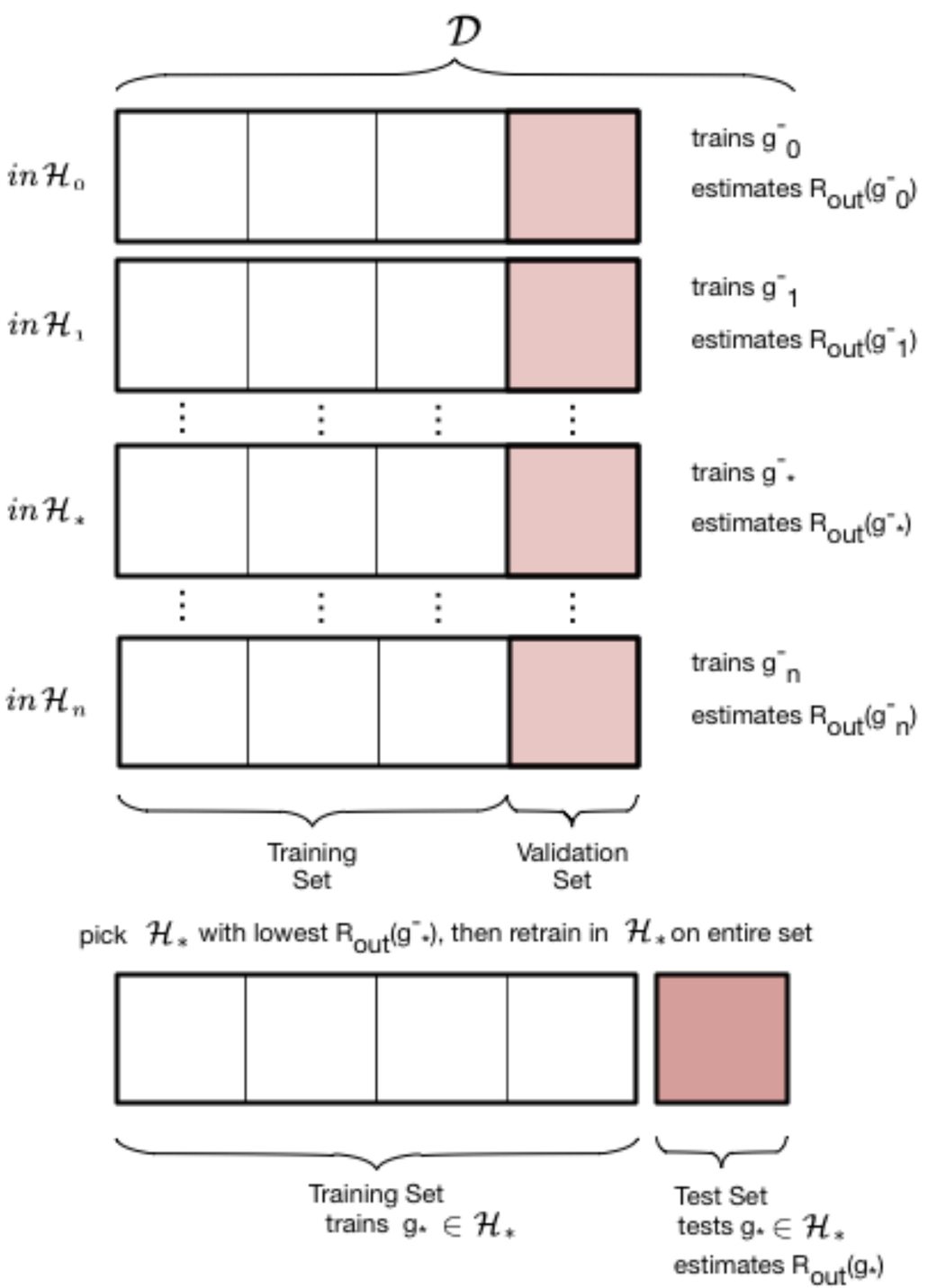
- no discussion on the error bars on our error estimates
- "visually fitting" a value of $d \implies$ contaminated test set.

The moment we **use it in the learning process, it is not a test set**.

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:

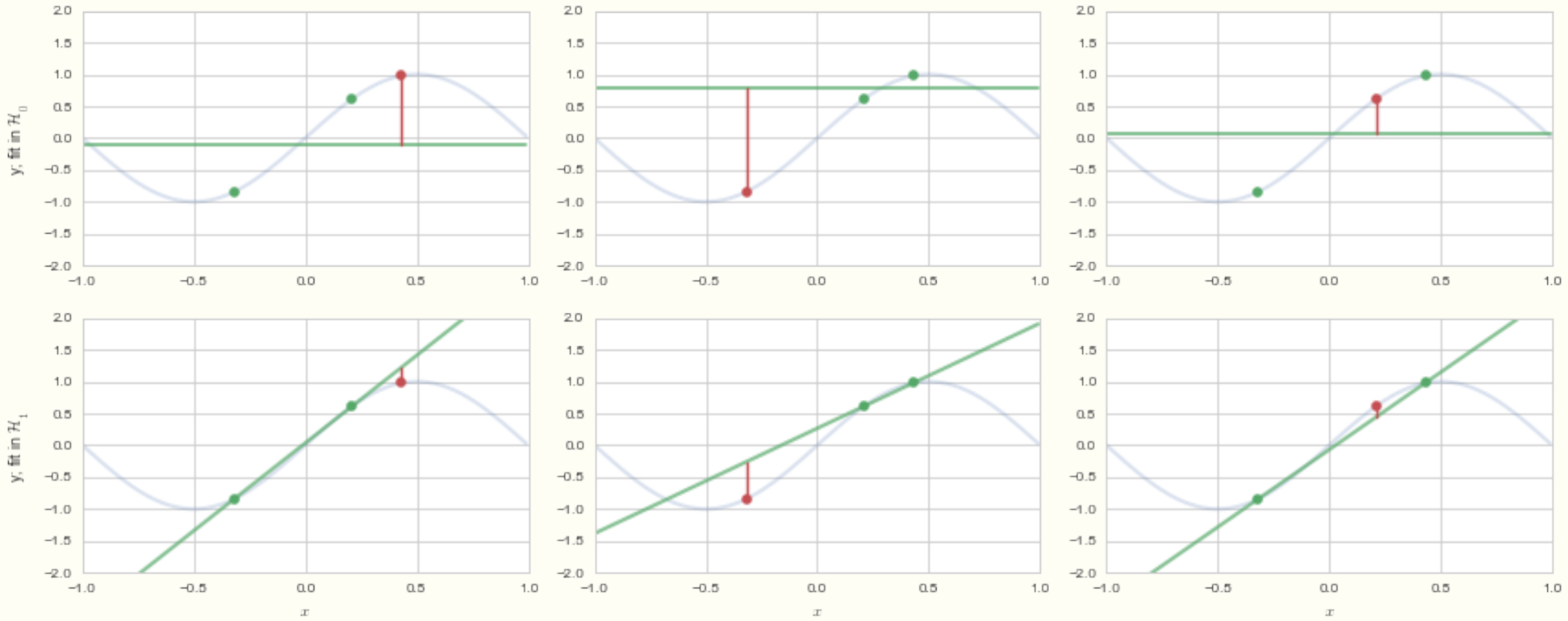
- we **wrongly** already attempted to fit d on our previous test set.
- choose instead the d^*, g^{-*} combination with the lowest validation set risk.
- $R_{val}(g^{-*}, d^*)$ has an optimistic bias since d effectively fit on validation set

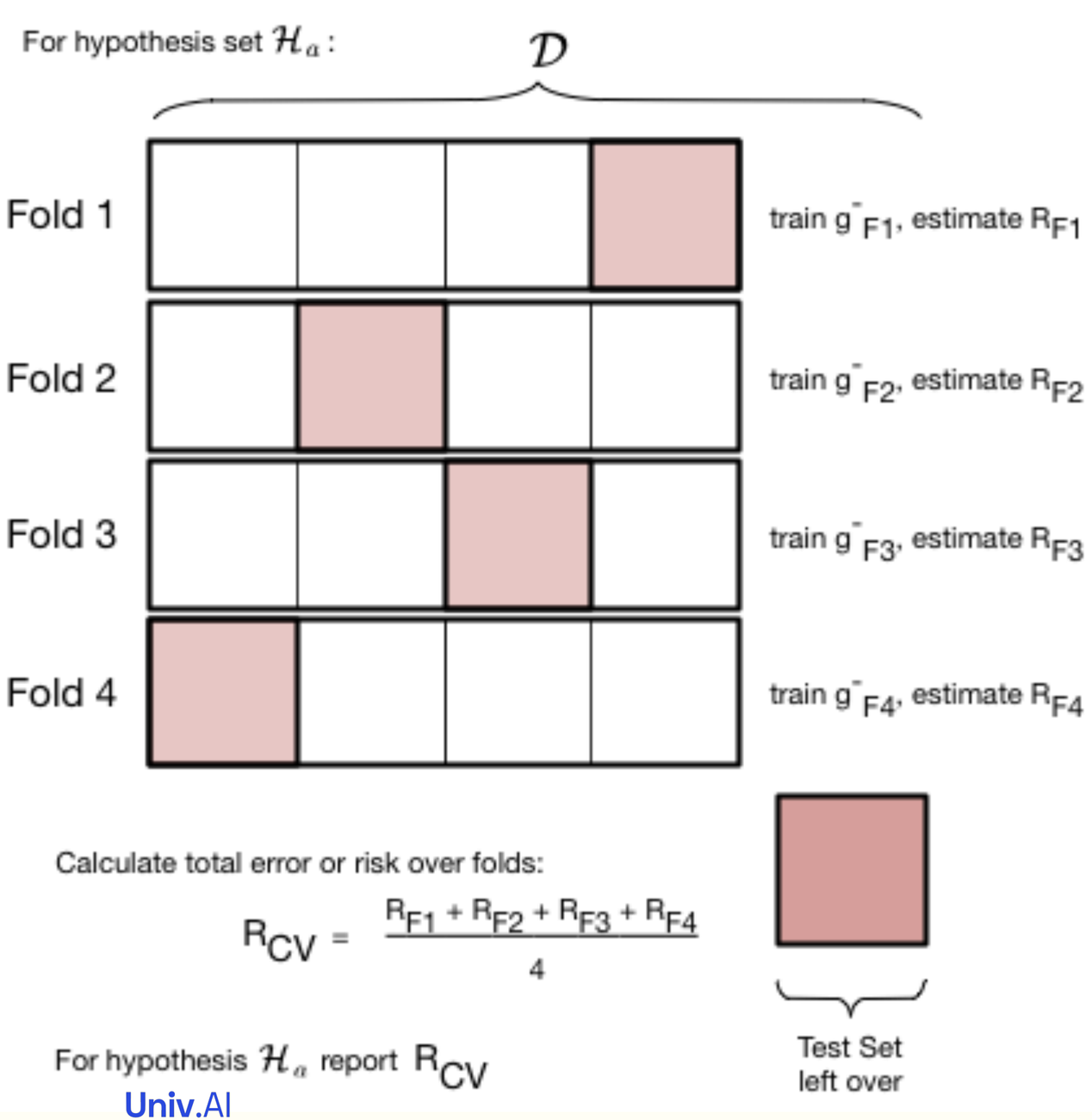
Then Retrain on entire set!

- 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 spead out?



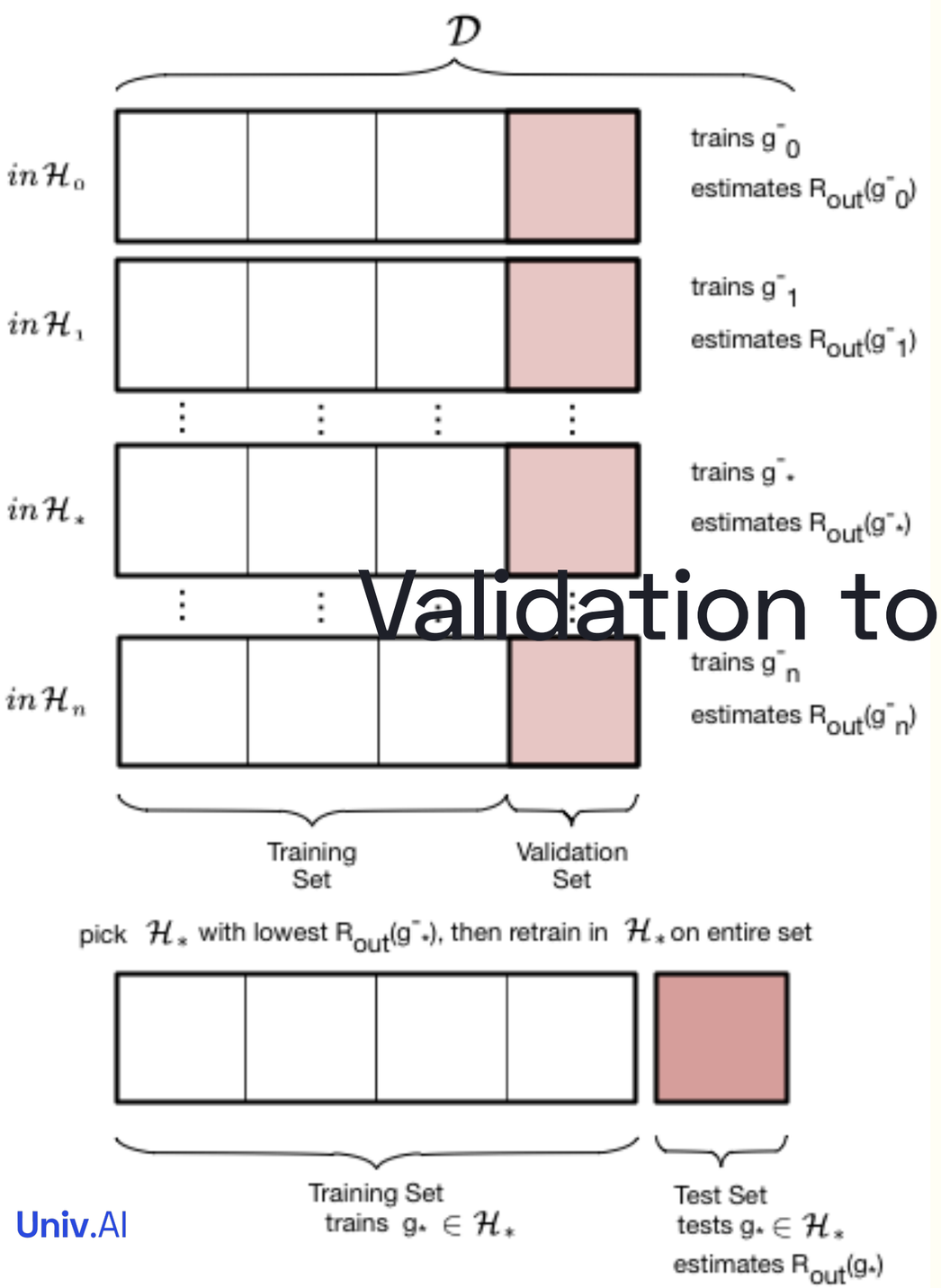


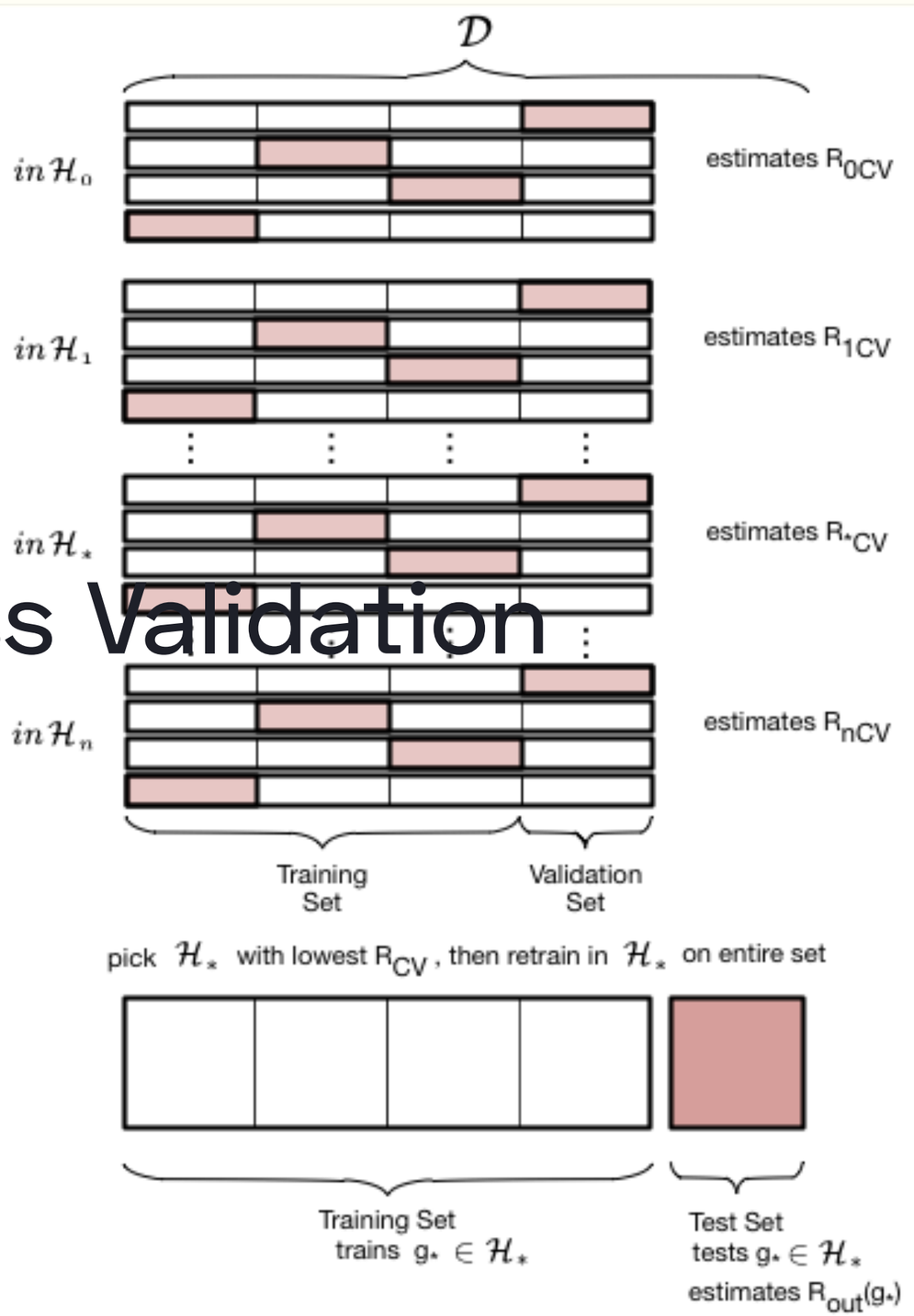
We then do

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.

Validation to Cross Validation

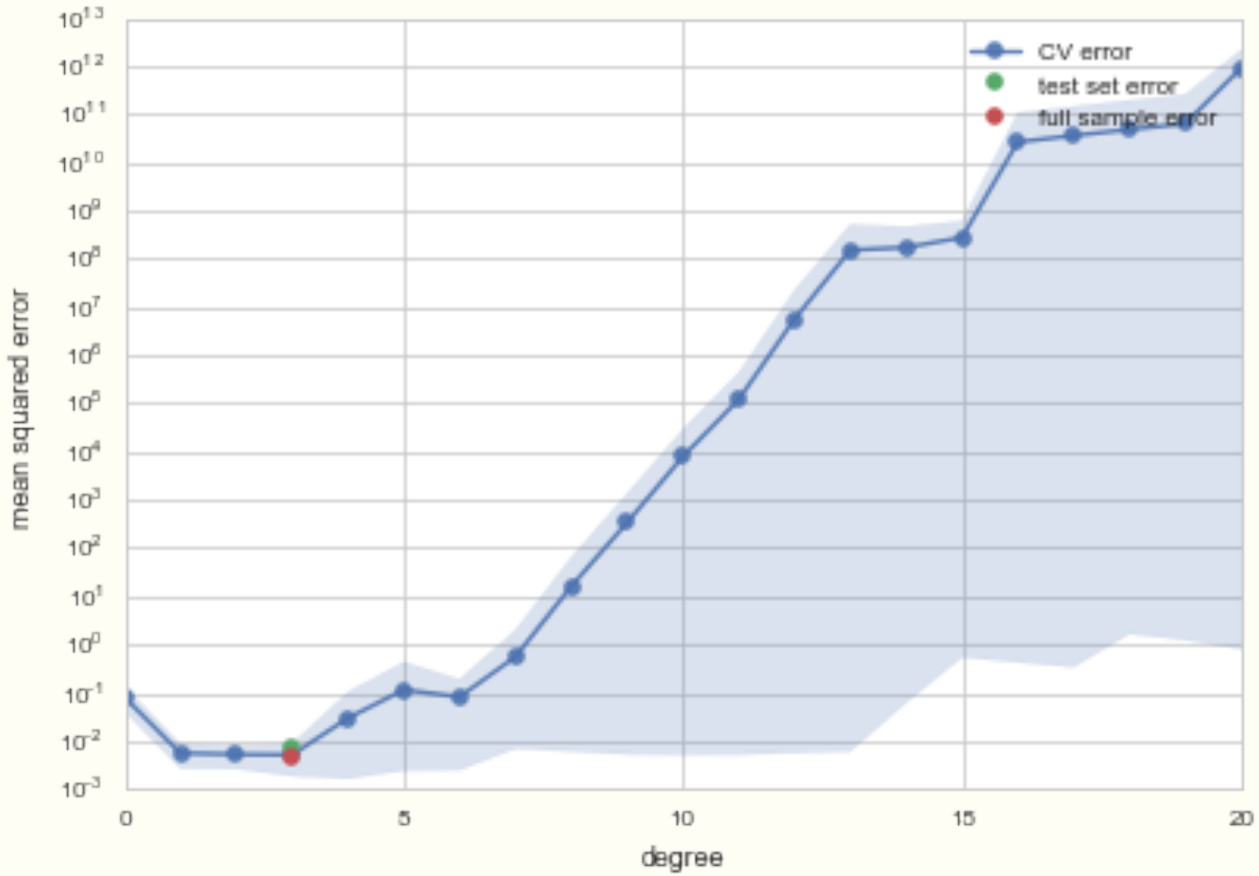




Univ.AI

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 a **small-world** approach to deal with finding the right model, where we'll choose a Hypothesis set that includes very complex models, and then find a way to subset this set. This method is called

2. Regularization

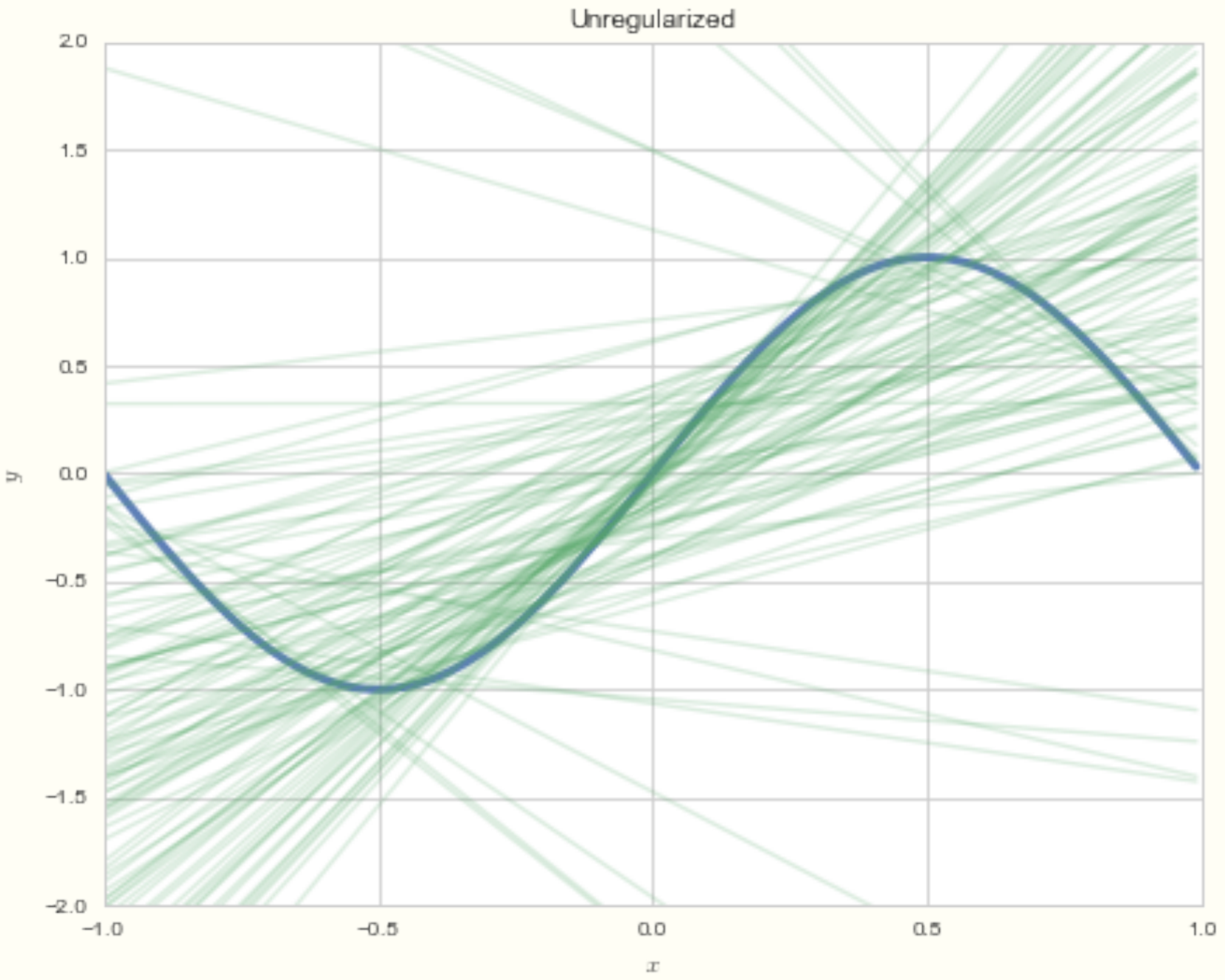
REGULARIZATION: A SMALL WORLD APPROACH

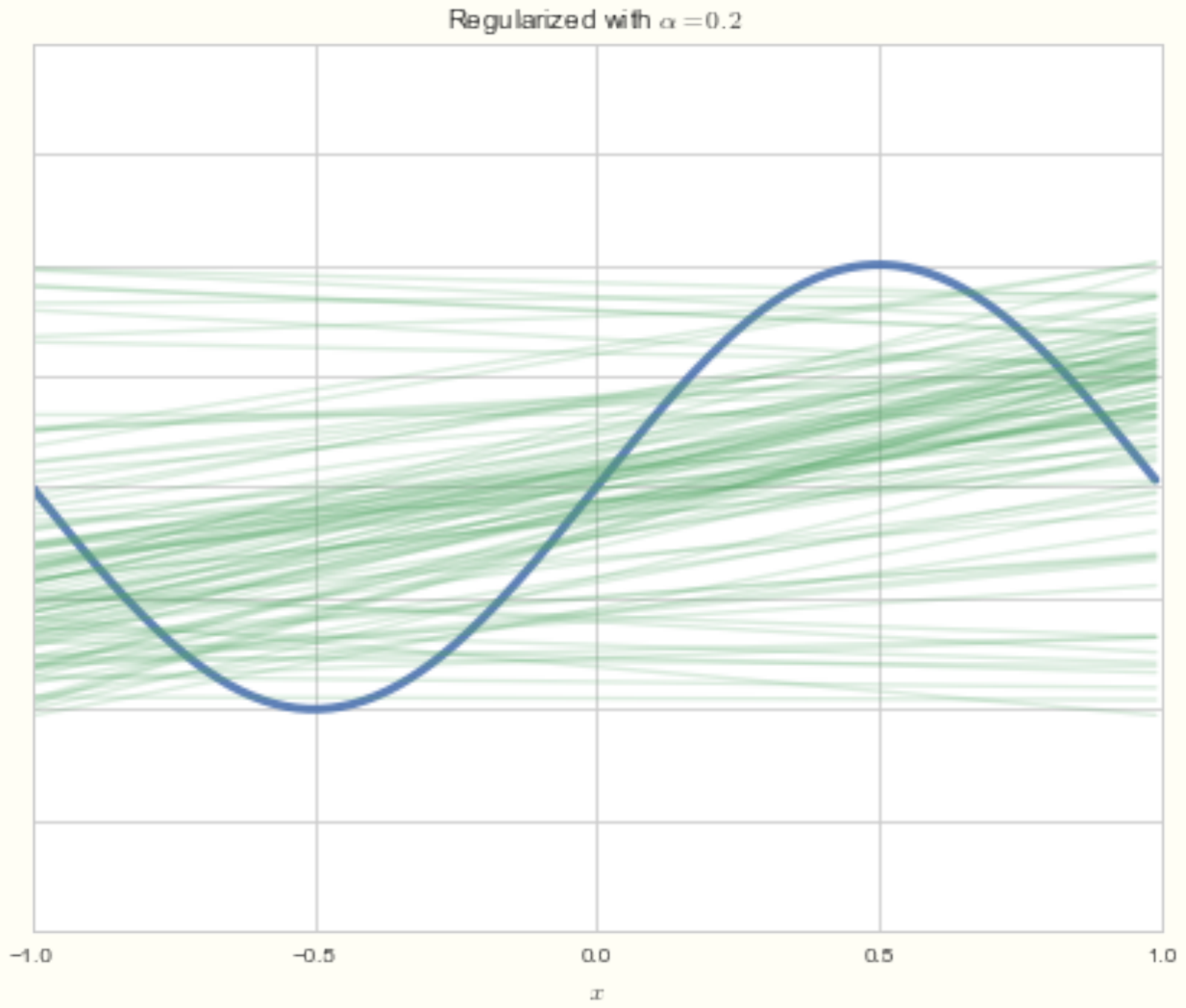
Keep higher a-priori complexity and impose a

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
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
y
−1.0
−0.5
0.0
0.5
1.0
x


Regularized with $\alpha = 0.2$
−1.0
−0.5
0.0
0.5
1.0
x

The math of regularization: small world

Consider the set of 10th order polynomials:

$$\mathcal{H}_{10} = \{h(x) = w_0 + w_1 \Phi_1(x) + w_2 \Phi_2(x) + w_3 \Phi_3(x) + \cdots + w_{10} \Phi_{10}(x)\}$$

Now suppose we just set some of these to 0, then we get $\mathcal{H}_2$ as a subset:

$$\mathcal{H}_2 = \begin{cases} h(x) = w_0 + w_1 \Phi_1(x) + w_2 \Phi_2(x) + w_3 \Phi_3(x) + \cdots + w_{10} \Phi_{10}(x) \\ \textbf{such that: } w_3 = w_4 = \cdots = w_{10} = 0 \end{cases}$$

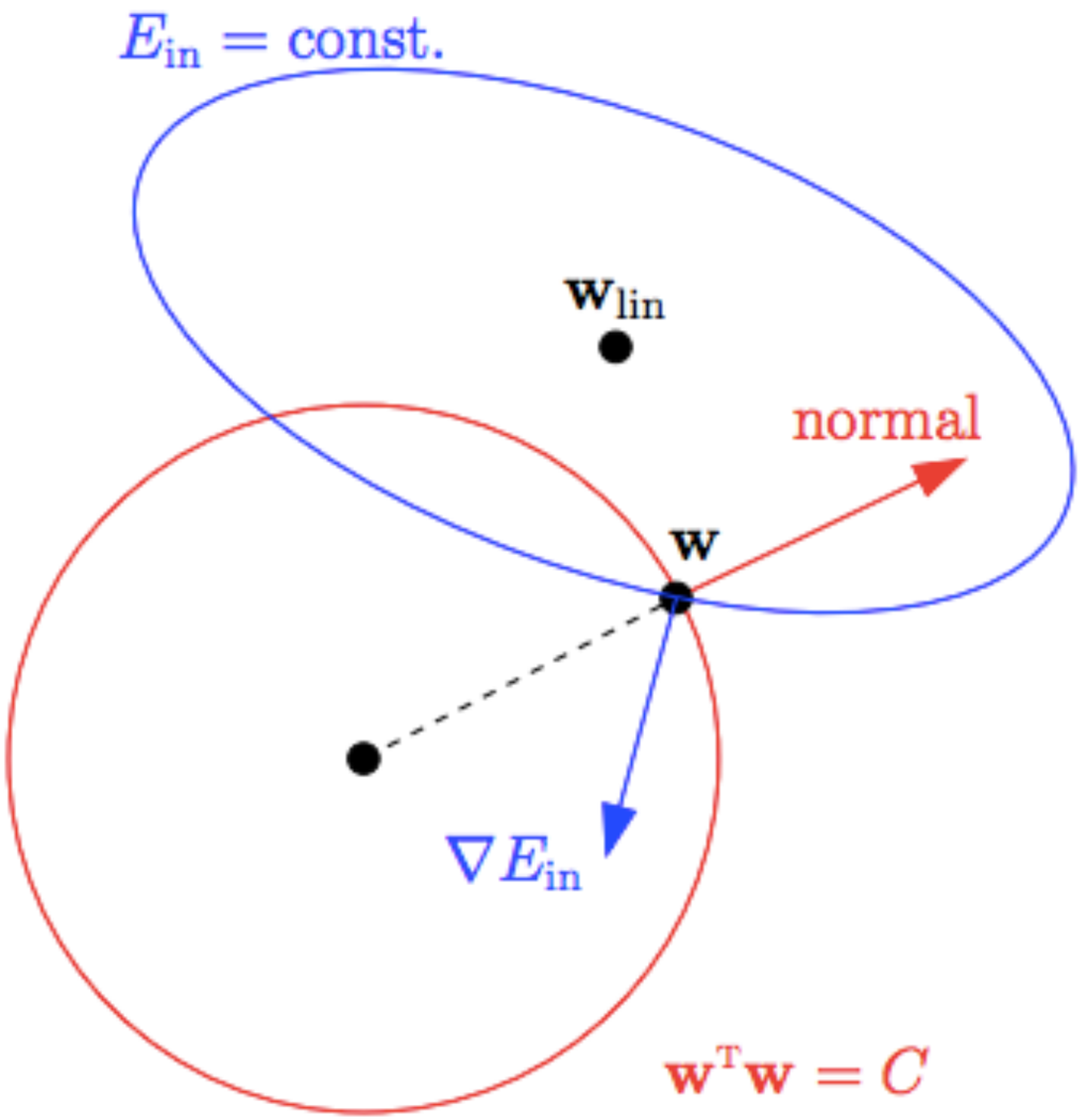
This is called a hard-order constraint.

L_2 Regularization or a soft budget constraint

$$\sum_{q=0}^{\mathrm{Q}} w_q^2 \leq C \leftarrow \text{BUDGET}$$

$$\mathcal{H}_C = \begin{cases} h(x) = w_0 + w_1\Phi_1(x) + w_2\Phi_2(x) + w_3\Phi_3(x) + \cdots + w_{10}\Phi_{10}(x) \\ \textbf{such that: } \sum_{q=0}^{10} w_q^2 \leq C \end{cases}$$

a soft budget constraint



The geometry of regularization

1. Optimal $\mathbf{w}$ tries to get as 'close' to $\mathbf{w}_{lin}$. Thus, optimal $\mathbf{w}$ will use full budget and be on the surface $\mathbf{w}^T\mathbf{w} = C$.
2. Surface $\mathbf{w}^T\mathbf{w} = C$, at optimal $\mathbf{w}$, should be perpindicular to ∇E_{in}.
3. Normal to surface $\mathbf{w}^T\mathbf{w} = C$ is the vector $\mathbf{w}$.
4. Surface is $\perp \nabla E_{\text{in}}$ and thus must be "tangent"

$$\nabla E_{\text{in}} \left(\mathbf{w}_{\text{reg}}\right) = -2\lambda_C \mathbf{w}_{\text{reg}}$$

Back to the Math: the lagrange multiplier formalism

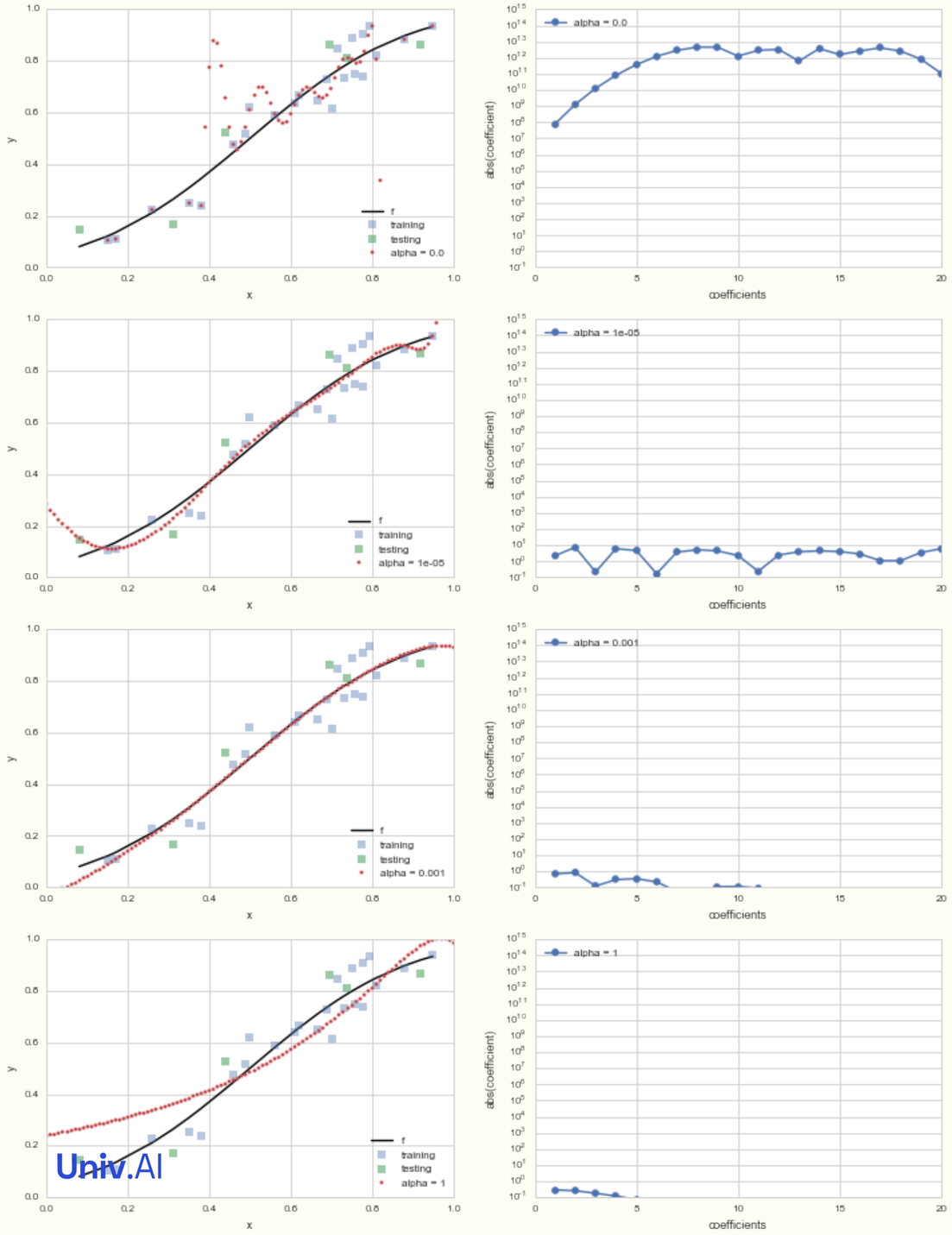
$$E_{\text{in}}(\mathbf{w}) \quad \text{is minimized, subject to: } \mathbf{w}^{\mathrm{T}}\mathbf{w} \leq C$$

$$\Leftrightarrow \quad \nabla E_{\text{in}}\left(\mathbf{w}_{\text{reg}}\right) + 2\lambda_C \mathbf{w}_{\text{reg}} = \mathbf{0}$$

$$\Leftrightarrow \nabla \left(E_{\text{in}}(\mathbf{w}) + \lambda_C \mathbf{w}^{\mathrm{T}}\mathbf{w}\right)\Big|_{\mathbf{w}=\mathbf{w}_{\text{rgg}}} = \mathbf{0}$$

$$\Leftrightarrow \quad E_{\text{in}}(\mathbf{w}) + \lambda_C \mathbf{w}^{\mathrm{T}}\mathbf{w} \quad \text{is minimized, unconditionally}$$

There is a correspondence: $C \uparrow \quad \lambda_C \downarrow$



ok so now how do we do

REGULARIZATION

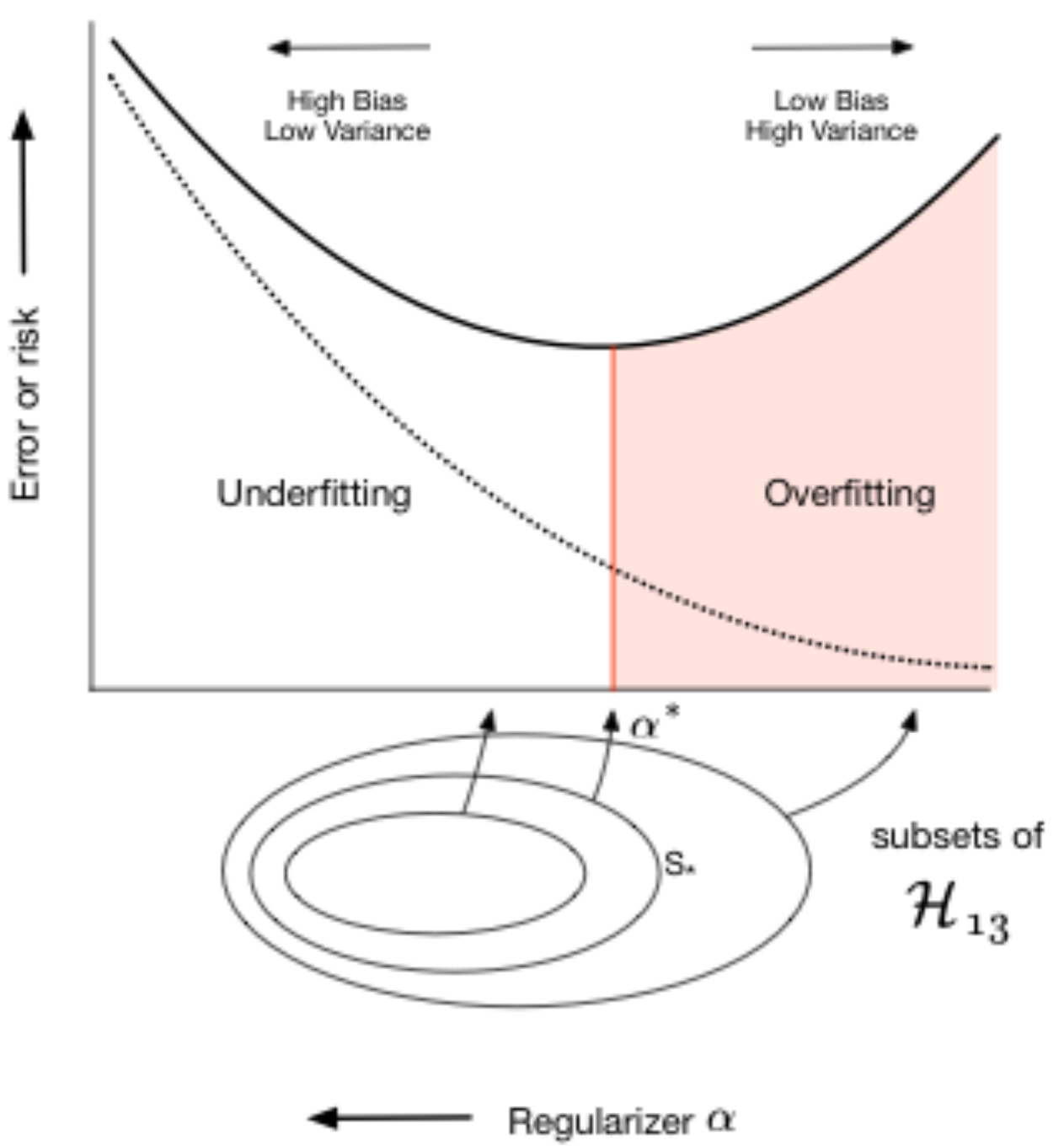
$$\mathcal{R}(h_j) = \sum_{y_i \in \mathcal{D}} (y_i - h_j(x_i))^2 + \lambda \sum_{i=0}^{j} \theta_i^2.$$

As we increase λ, coefficients go towards 0.

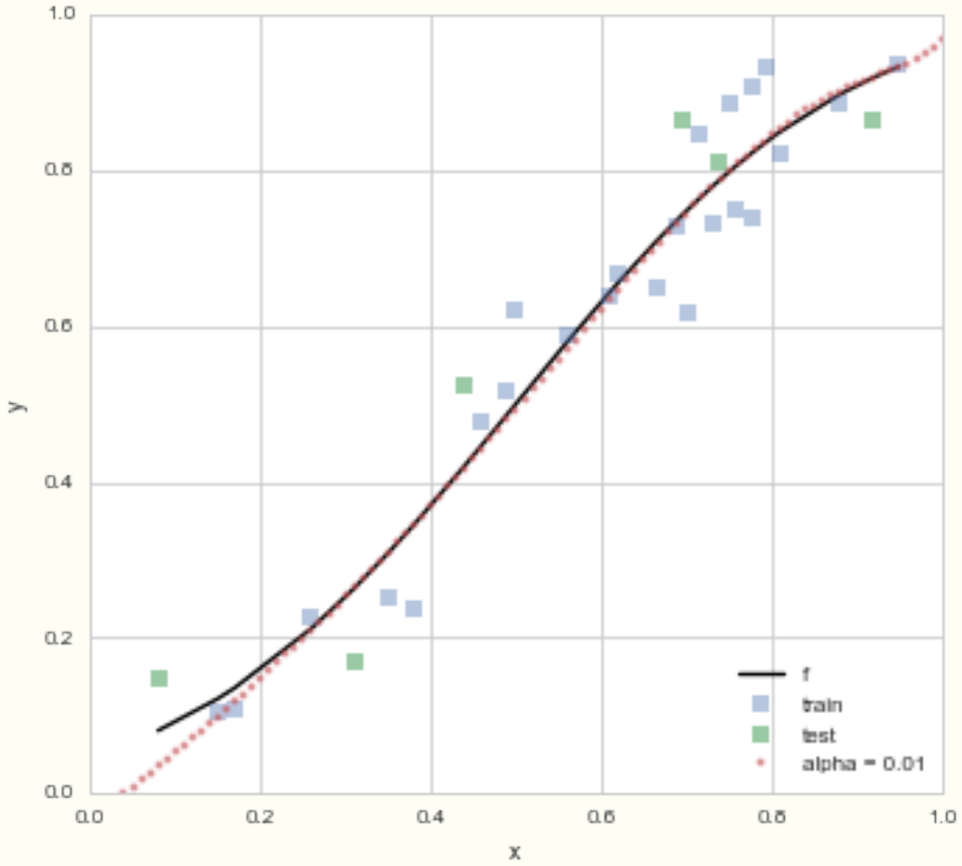
Lasso uses $\lambda \sum_{i=0}^{j} |\theta_i|$, sets coefficients to exactly 0.

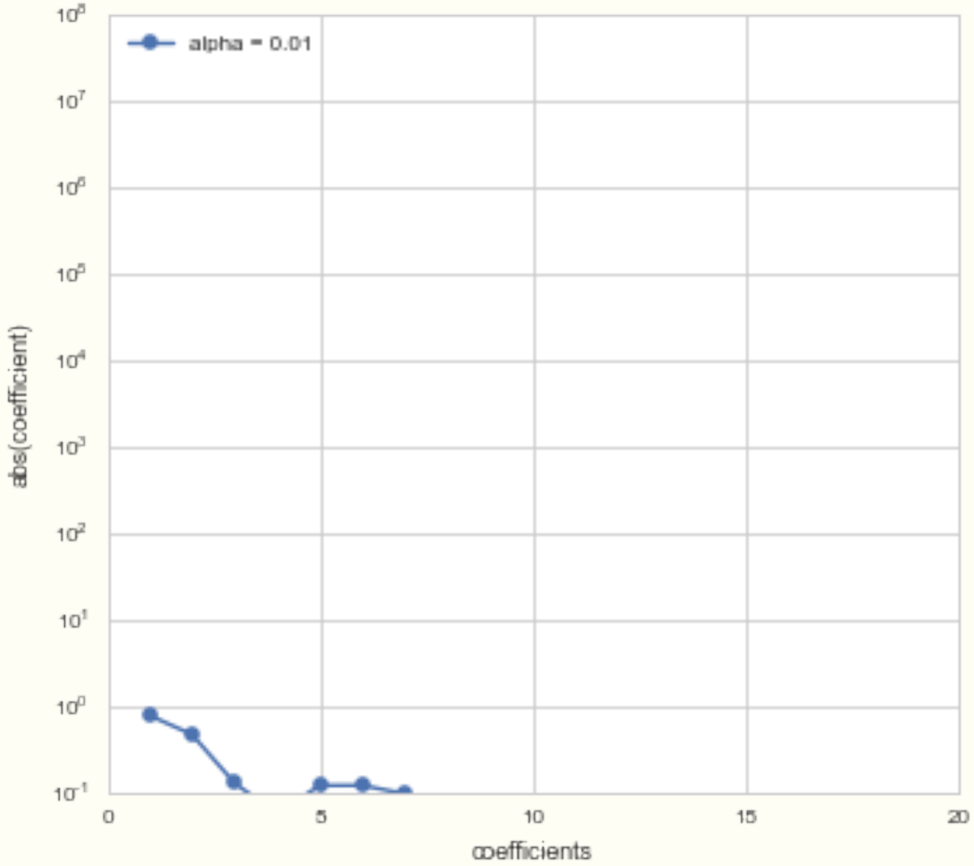
Structural Risk Minimization

- 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

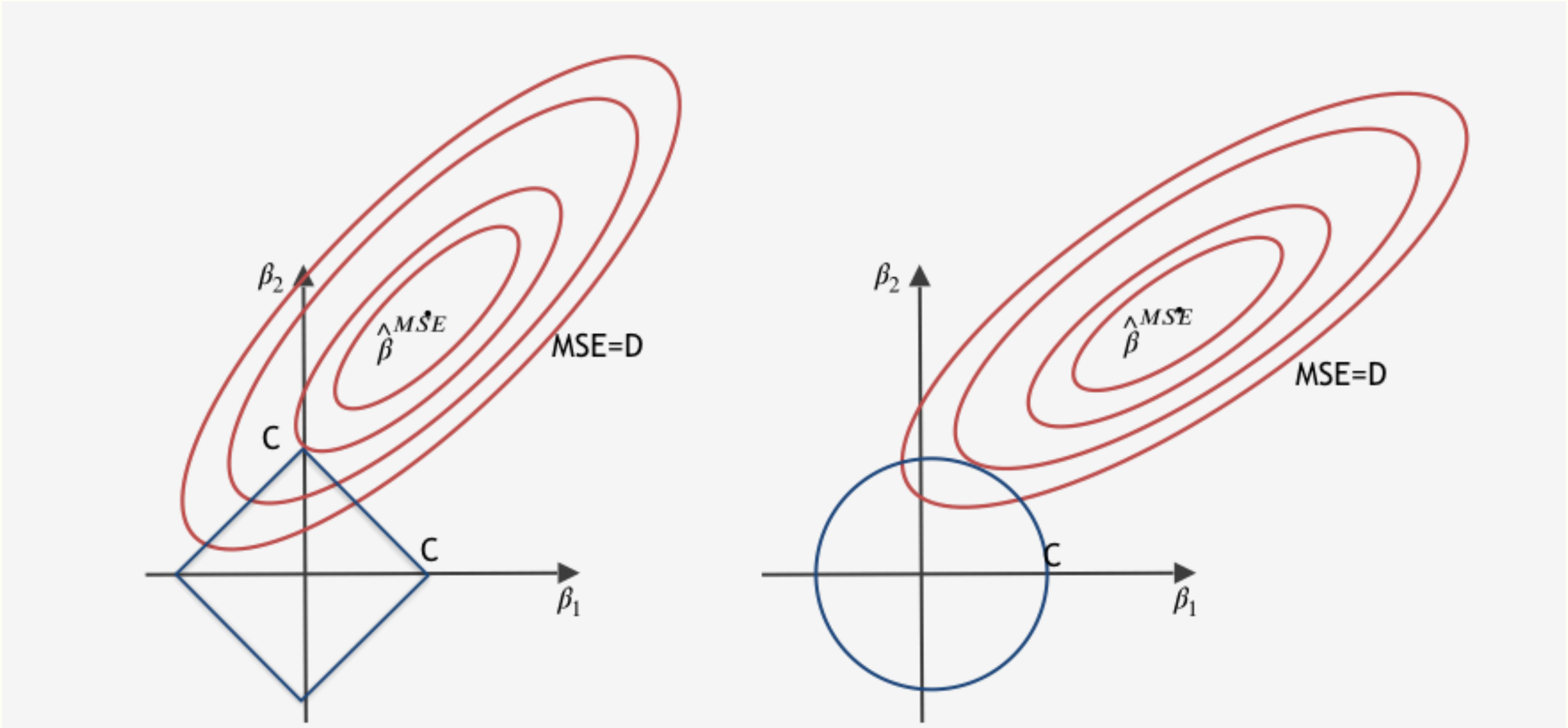


Regularization with Cross-Validation





Lasso vs Ridge Geometry



3. Lots of features

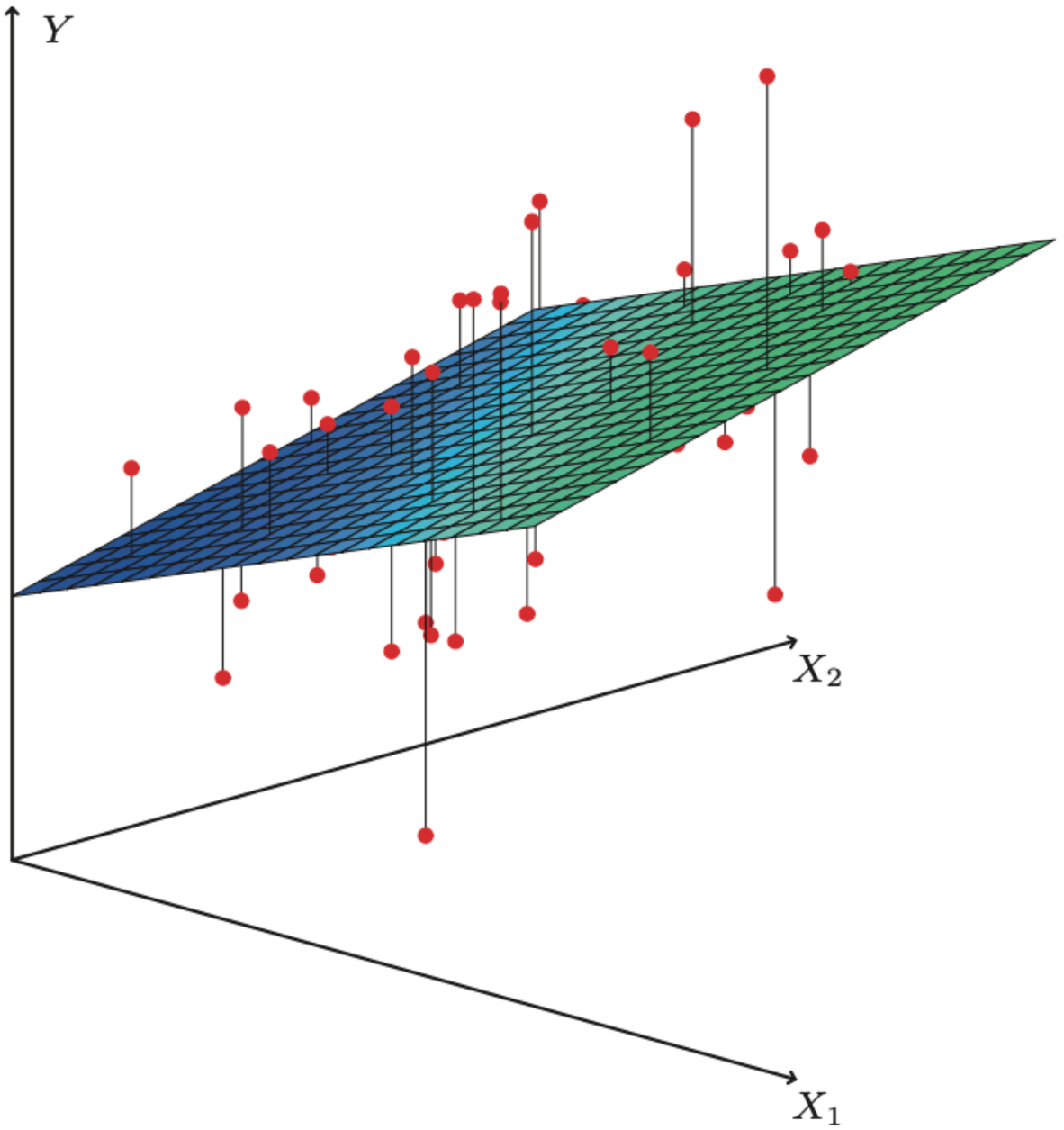
AKA: features are not just polynomial powers

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.



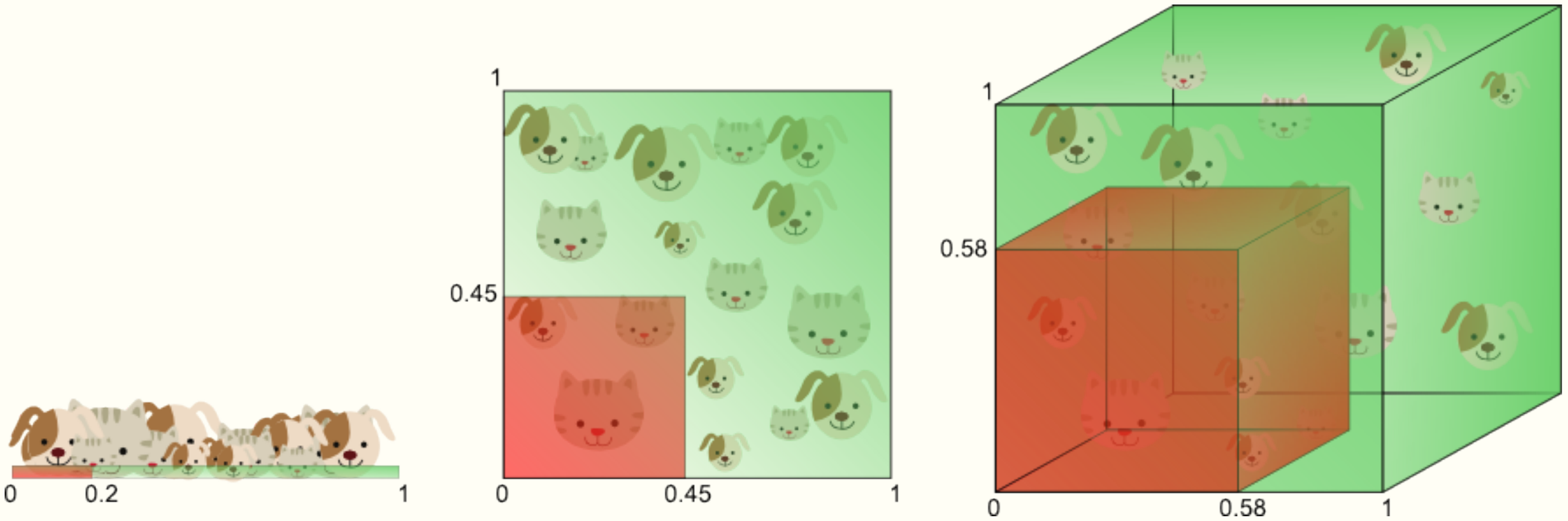
Curse of Dimensionality

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 :

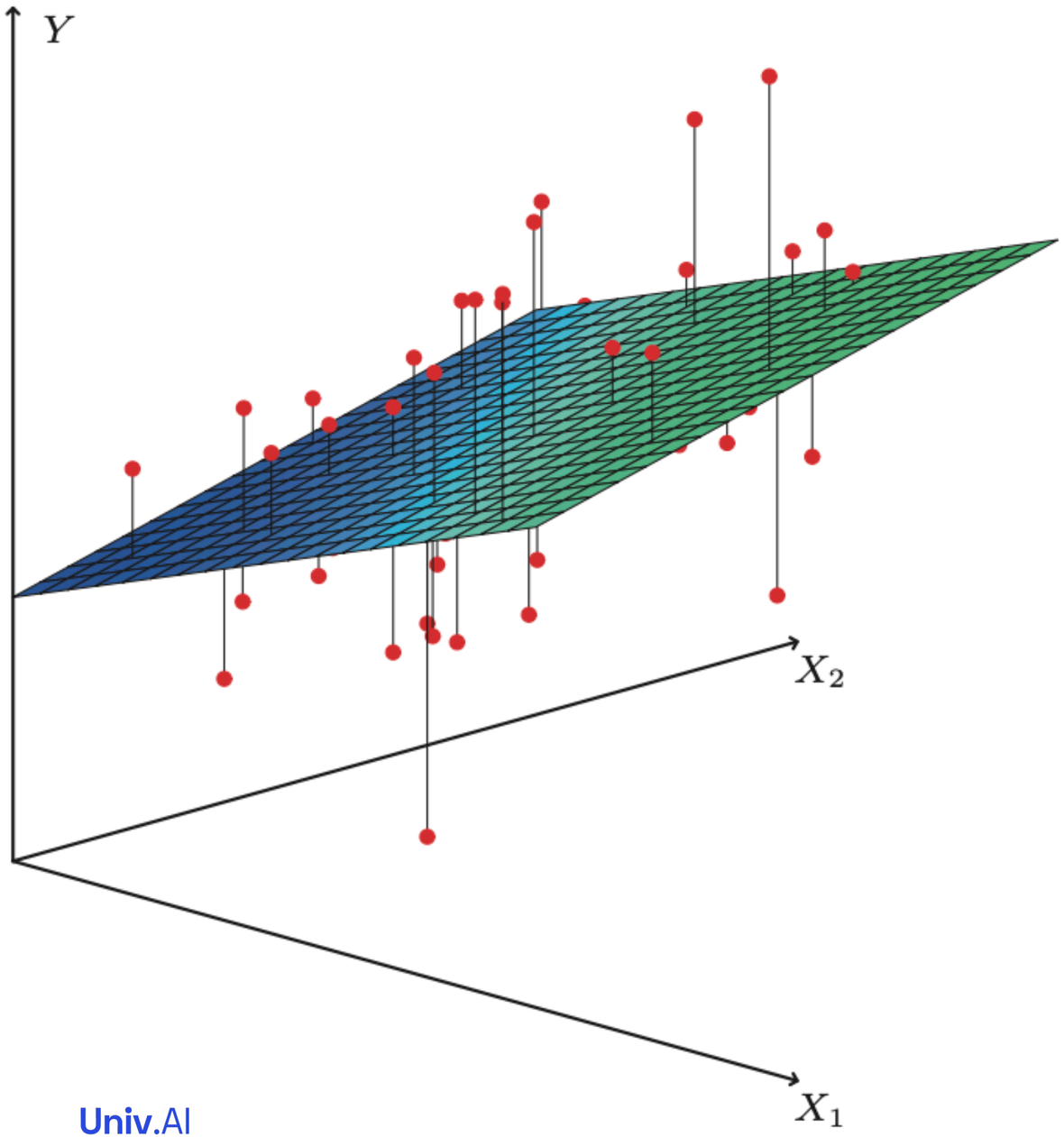
- data is sparser in higher dimensions
- volume moves to the outside

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.