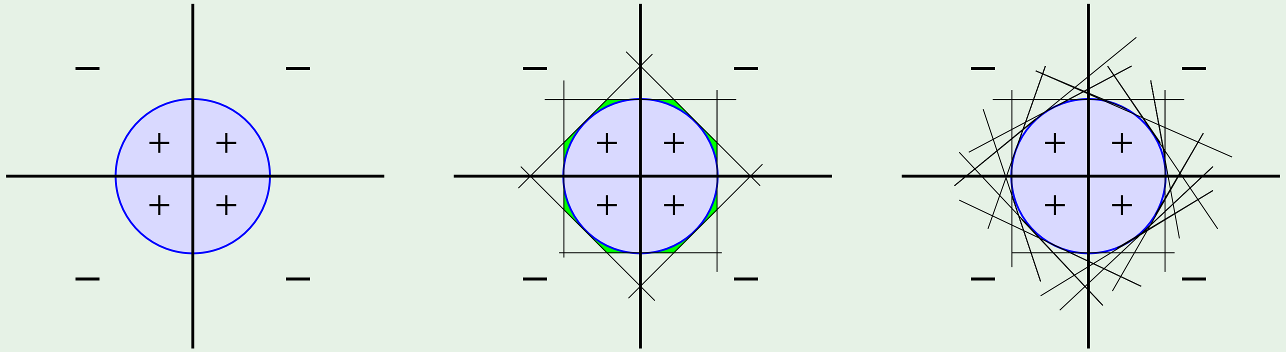
Review of Lecture 10

- **Multilayer perceptrons**



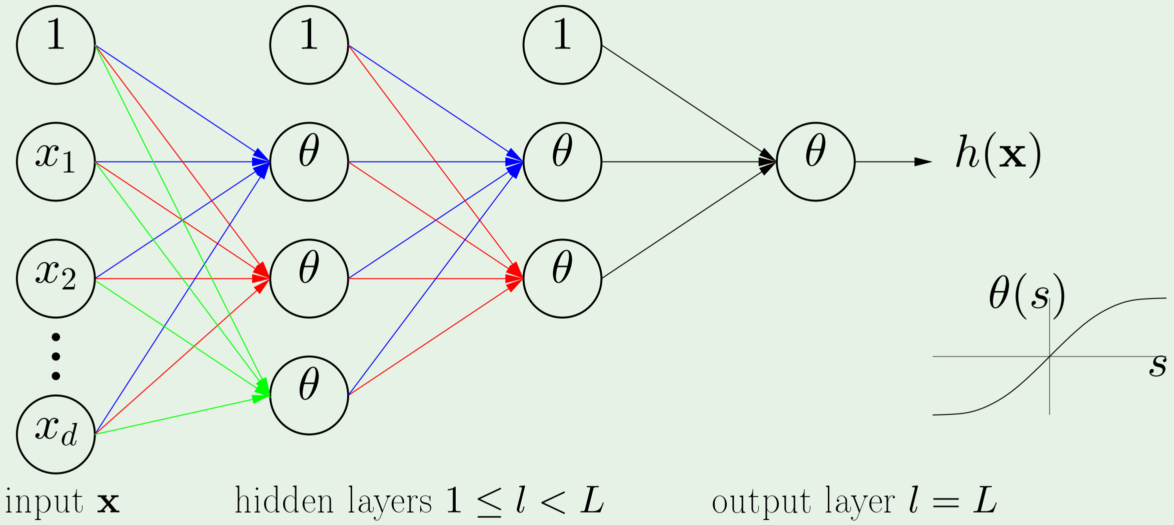
Logical combinations of perceptrons

- **Neural networks**

$$x_j^{(l)} = \theta\left(\sum_{i=0}^{d^{(l-1)}} w_{ij}^{(l)}\; x_i^{(l-1)}\right)$$

where $\theta(s) = \tanh(s)$

Most important aspect of tanh is that it is smooth/twice differentiable, so the dependency of the error in the output on w will be a well-behaved function and we can apply gradient descent



- **Backpropagation**

$$\Delta w_{ij}^{(l)} = -\eta\; x_i^{(l-1)} \delta_j^{(l)}$$

where

$$\delta_i^{(l-1)} = (1 - (x_i^{(l-1)})^2) \sum_{j=1}^{d^{(l)}} w_{ij}^{(l)}\; \delta_j^{(l)}$$

Learning From Data

Yaser S. Abu-Mostafa
California Institute of Technology

Lecture 11: **Overfitting**

Sponsored by Caltech's Provost Office, E&AS Division, and IST • Tuesday, May 8, 2012

Outline

- What is overfitting?
- The role of noise
- Deterministic noise
- Dealing with overfitting

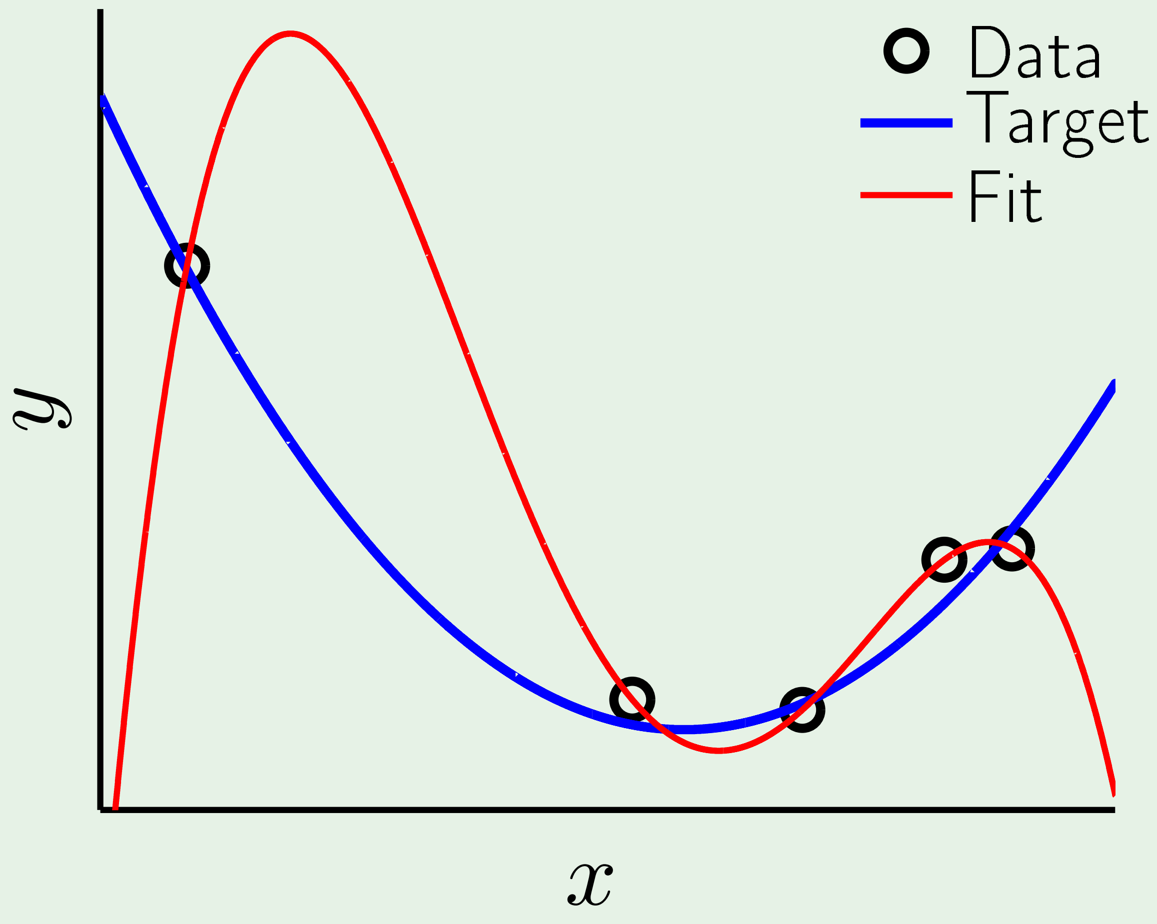
Illustration of overfitting

Simple target function

5 data points- **noisy**

4th-order polynomial fit

$$E_{\text{in}} = 0, \quad E_{\text{out}} \text{ is huge}$$



Overfitting is a comparative term, so one situation must be worse than another - you went further than you should have. There is a distinction between overfitting and bad generalization: since using a 3rd order polynomial we would not in general get zero Ein but we would get a better Eout, so the bad generalization was a result of using 4th order rather than 3rd order, hence overfitting.

Note that this example shows overfitting between two models. However overfitting in neural networks is within the same model.

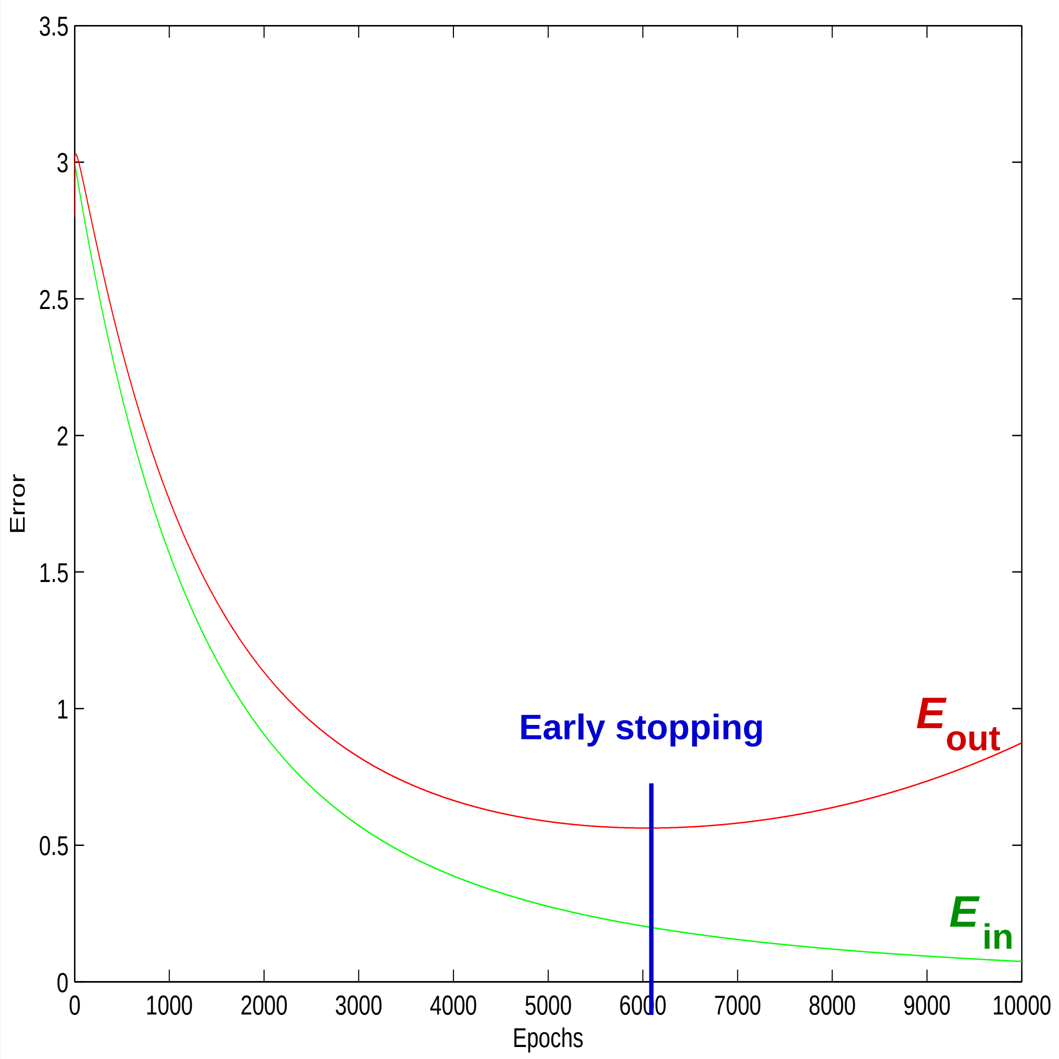
Overfitting versus bad generalization

Neural network fitting noisy data

occurs when but

Overfitting: $E_{\text{in}} \downarrow \quad E_{\text{out}} \uparrow$

Ein and Eout are similar values at the start since we only have one hypothesis which does not depend on the dataset, the randomly initialized weights, so not a surprise they are the same value. As the epochs increase and we explore more of the weight space, we get the associated benefit/harm from this increase gradually - the effective VC dimension (if we can define that) grows as we explore the space until we have (potentially) explored the whole space, ending as the total number of free parameters of the model.



Note: stopping the algorithm based off of the Eout from the test set means that the test set has now been used in training - it is now contaminated. The test set should only be used for the very final test of generalization - when the best model has been selected. See validation later for a way around this.

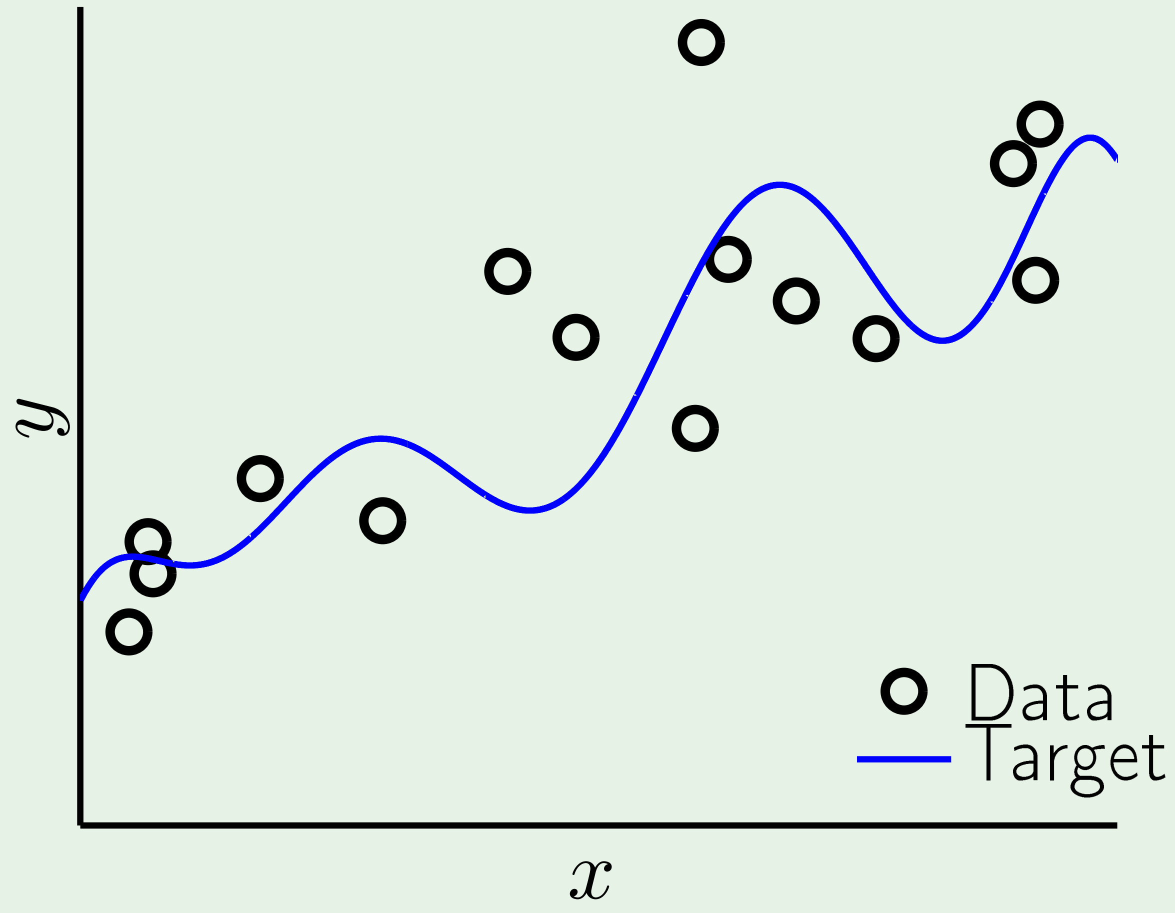
The culprit

Overfitting: "fitting the data more than is warranted"

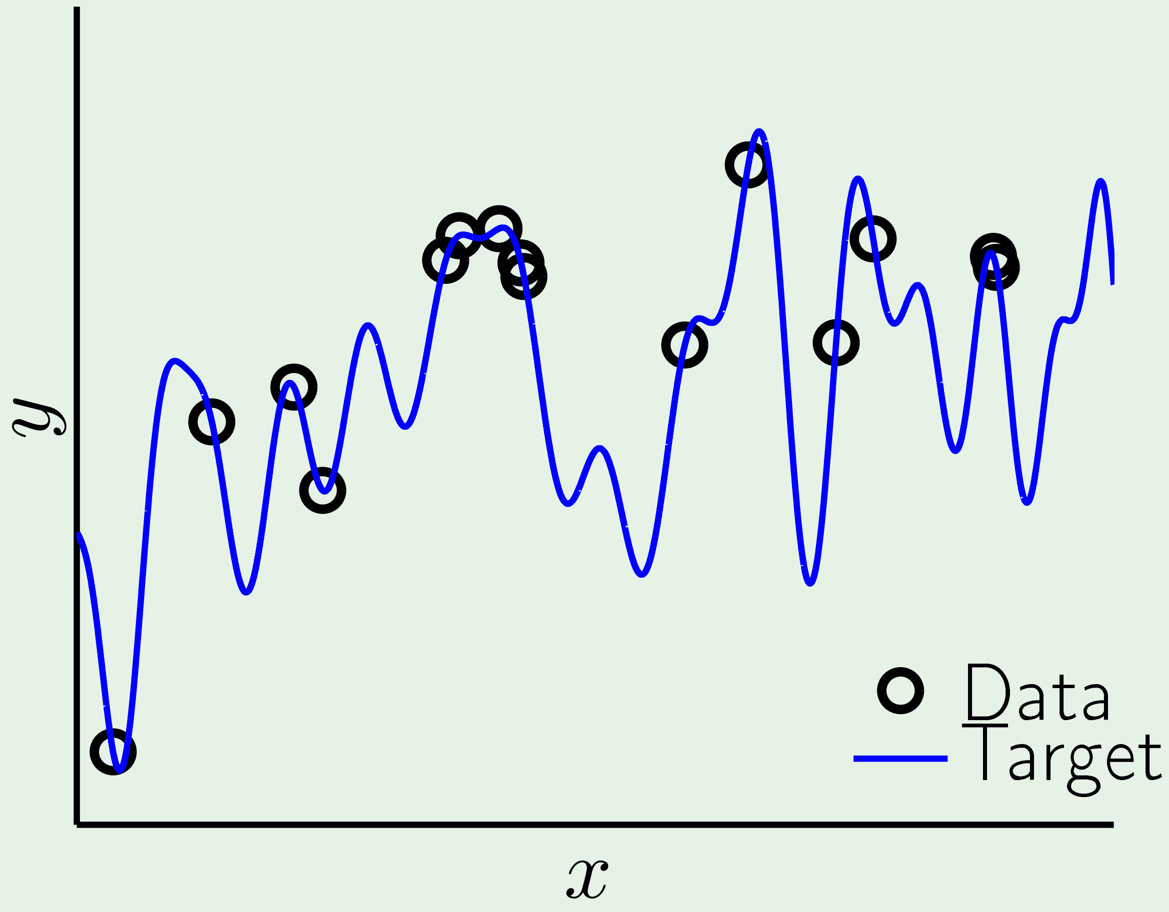
Culprit: fitting the noise - harmful

Case study

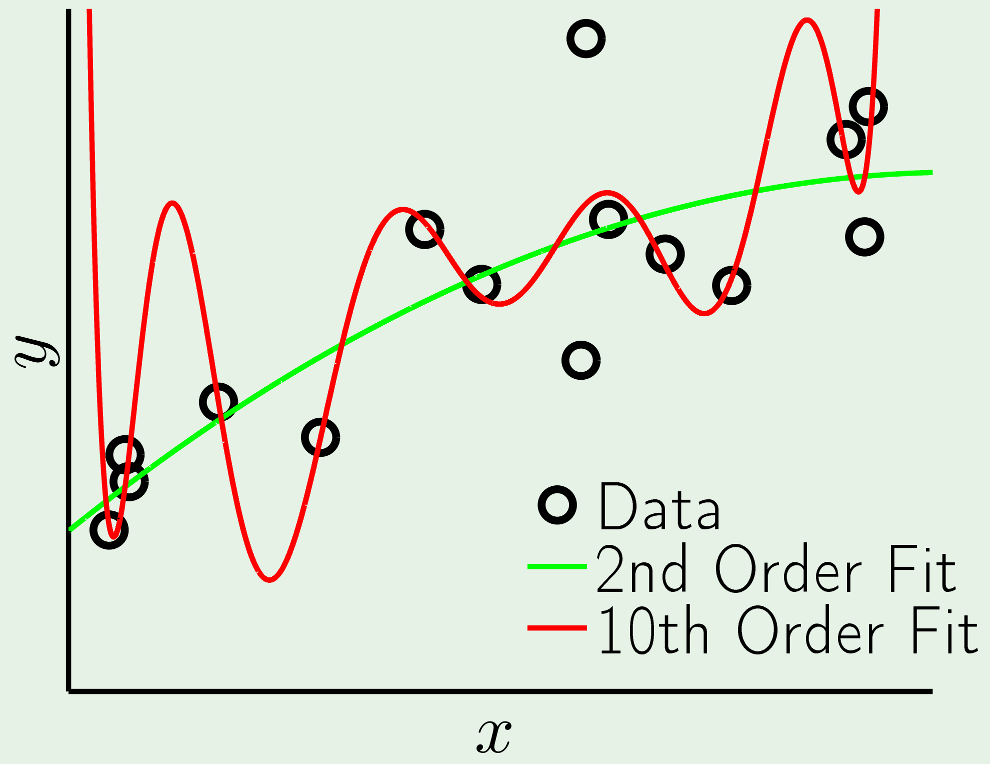
10th-order target + noise

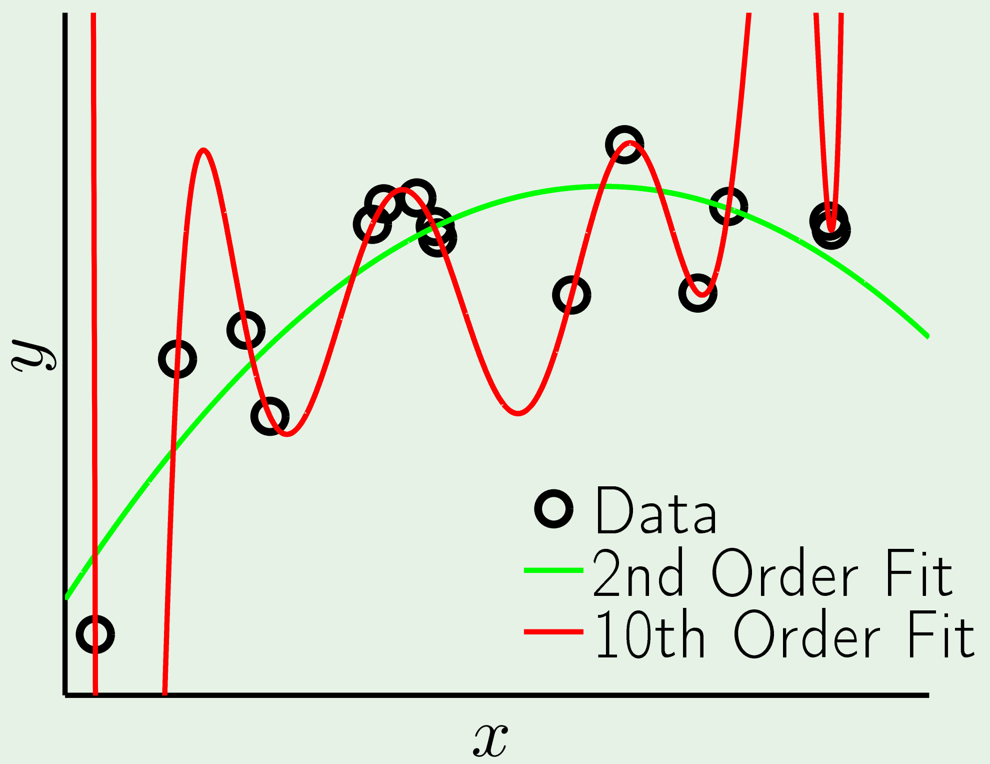


50th-order target



Two fits for each target





Noisy low-order target

	2nd Order	10th Order
E_{in}	0.050	0.034
E_{out}	0.127	**9.00**

Noiseless high-order target

	2nd Order	10th Order
E_{in}	0.029	10^{-5}
E_{out}	0.120	**7680**

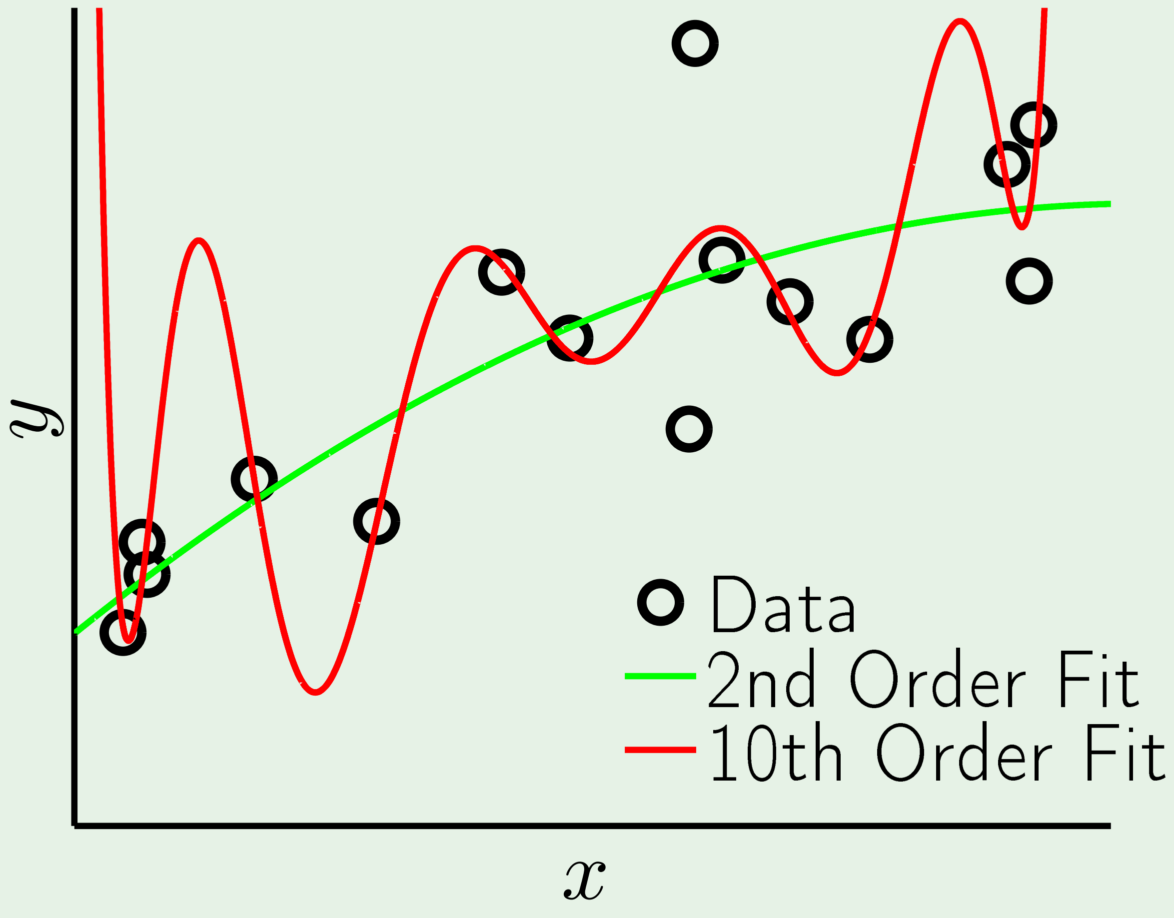
An irony of two learners

Two learners O and R

They <u>know</u> the target is 10th order

O chooses $\mathcal{H}_{10}$ R chooses $\mathcal{H}_2$

Better to match complexity of the model to the amount of data we have - generalization issues depend on the size and the quality of the dataset.

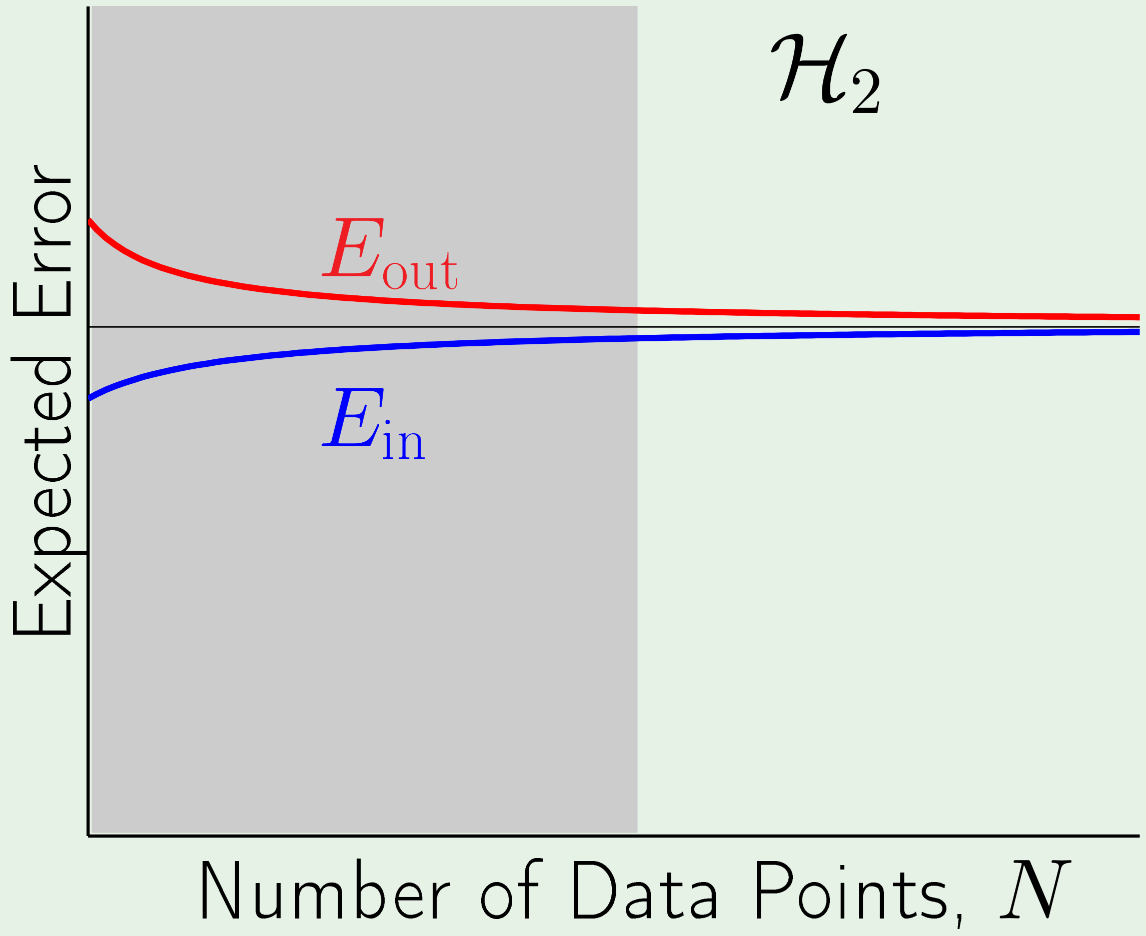


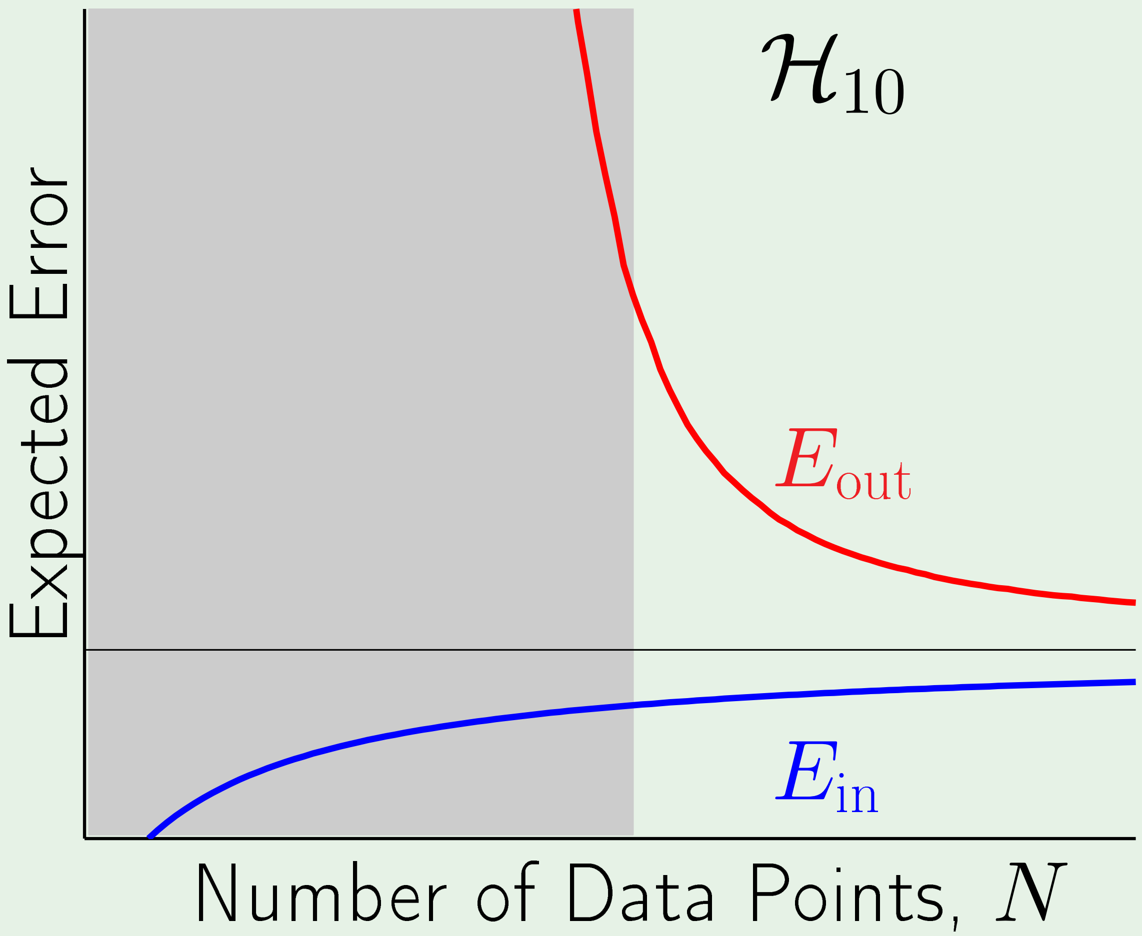
Learning a 10th-order target

We have see this case

Remember learning curves?

The baseline/inevitable error comes from both the limitations of the model and the noise in the data. The grey area is where overfitting is happening: while Ein for H10 is smaller (even if it is smaller for all N), Eout for H10 is larger than for H2.





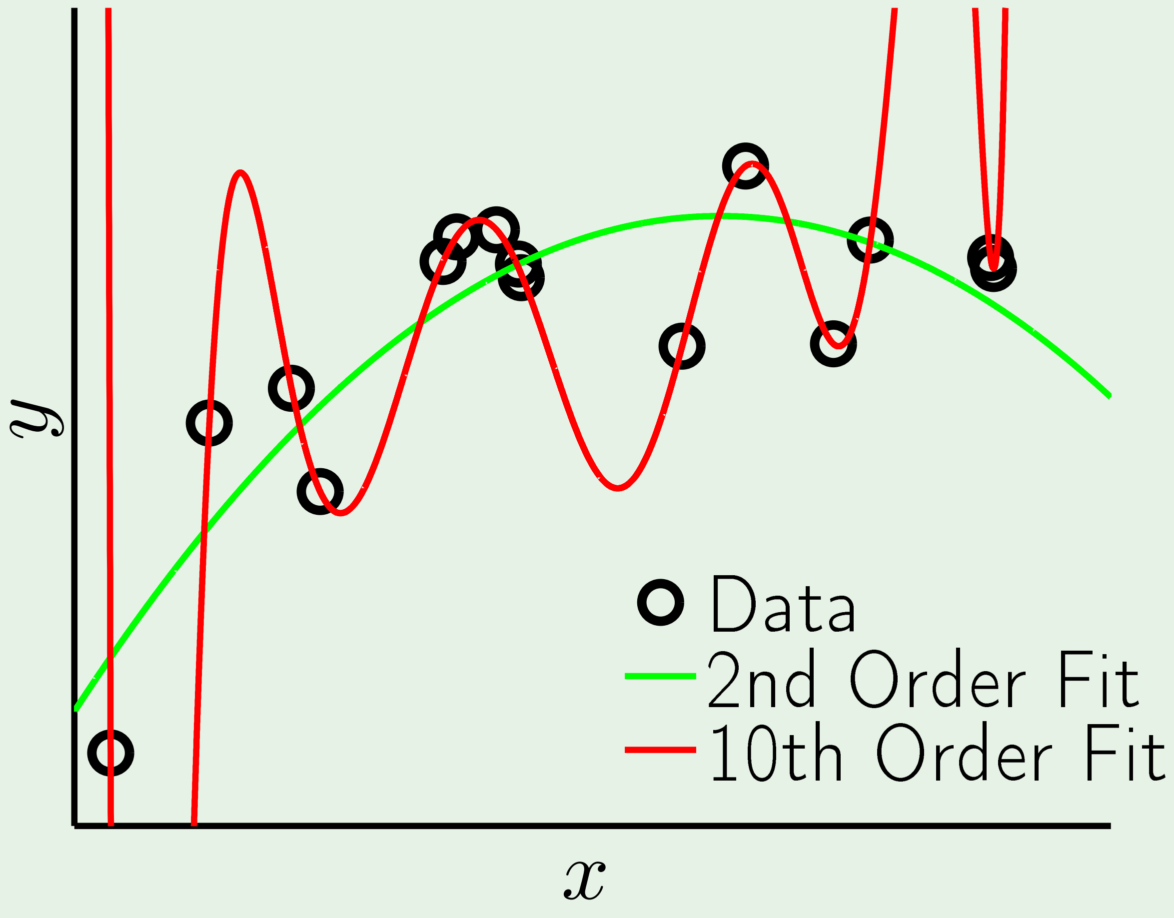
Even without noise

The two learners $\mathcal{H}_{10}$ and $\mathcal{H}_2$

They <u>know</u> there is no noise

Is there really no noise?

It seems that overfitting occurs both when we have noise in the data and when the data appears noiseless, so it seems target complexity may be related - see next slide.

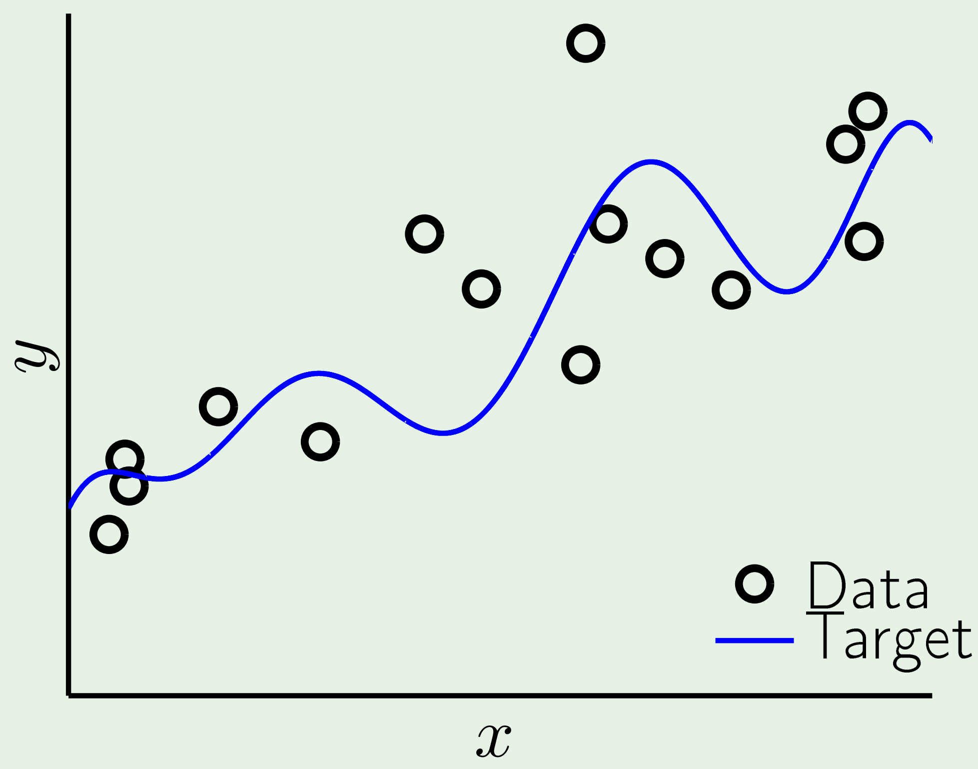


Learning a 50th-order target

A detailed experiment

We do not generate the alpha coefficients randomly as doing that gives us non-interesting polynomials that are generally dominated by a single power of x. Instead of random alpha, we use normalized Legendre polynomials (i.e. the choice of the coefficients is such that two Legendre polynomials of different order are orthogonal) - like harmonics in a sinusoidal expansion, the inner product of each order Legnedre polynomial are orthogonal. If we have a combination of Legendre with random coefficients, we can generate interesting shapes. The alphas now just become to sum of the coefficients for each power of x. We are now just generating the alphas in an elaborate way so that we get interesting targets. Note we normalize to make sigma^2 meaningful as the energy of the stochastic noise.

Impact of **noise level** and **target complexity**



$$y = f(x) + \underbrace{\epsilon(x)}_{\sigma^2} = \underbrace{\sum_{q=0}^{Q_f} \alpha_q \, x^q}_{\text{normalized}} + \epsilon(x)$$

noise level: σ^2 - with expectation value (of the error function) as zero. Sigma^2 can be seen as the energy of stochastic noise

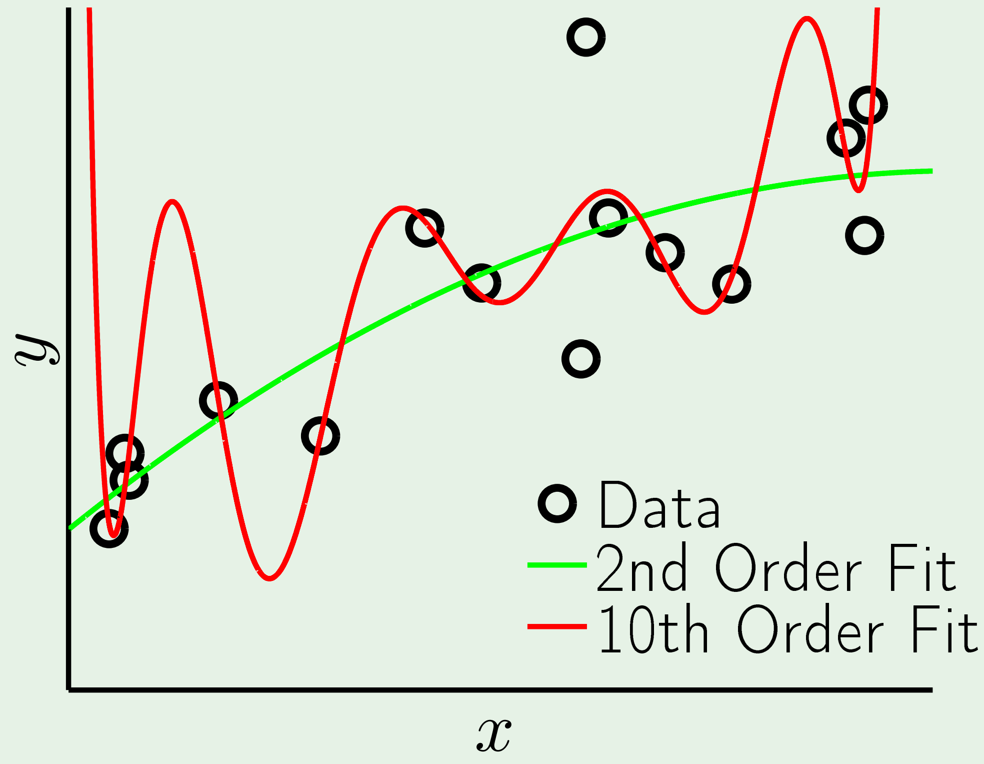
target complexity: Q_f - describes the complexity of f

data set size: N

The overfit measure

We fit the data set $(x_1, y_1), \cdots, (x_N, y_N)$ using our two models:

$\mathcal{H}_2$: 2nd-order polynomials $\mathcal{H}_{10}$: 10th-order polynomials



Compare out-of-sample errors of

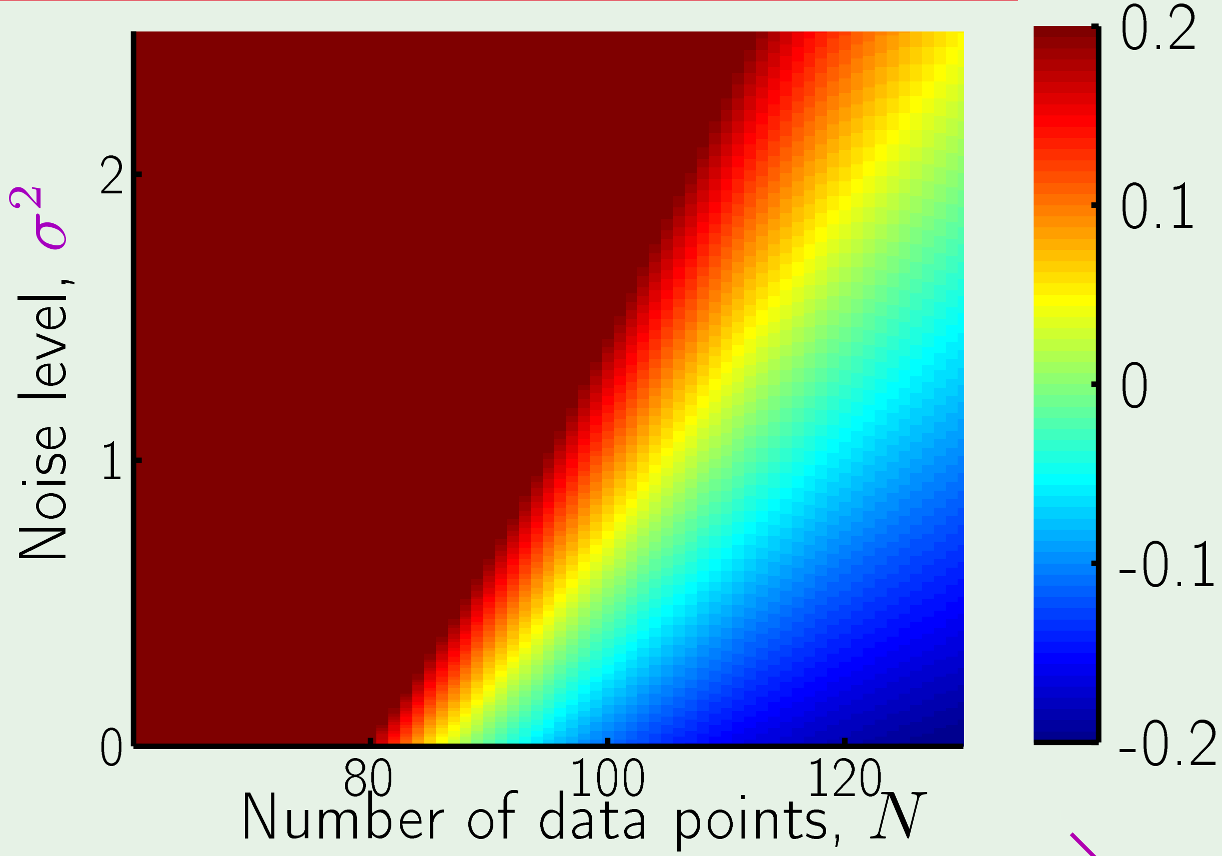
$$g_2 \in \mathcal{H}_2 \quad \text{and} \quad g_{10} \in \mathcal{H}_{10}$$

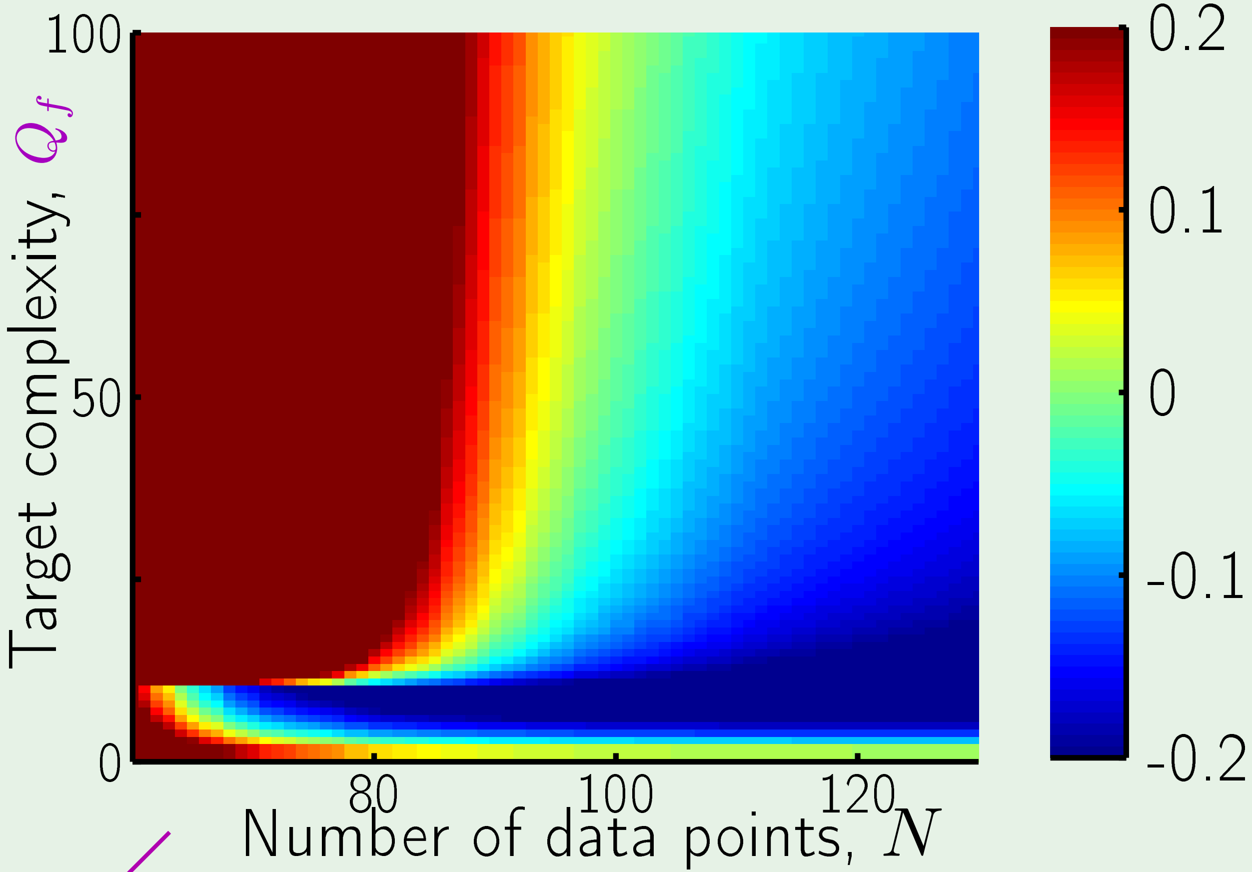
overfit measure: $E_{\text{out}}(g_{10}) - E_{\text{out}}(g_2)$

Large positive value indicates a lot of overfitting, small positive means less overfitting, negative indicates no overfitting

Here we do the experiment over ten million iterations (for all kinds of parameters): generate f, generate a dataset, fit using both hypothesis sets, evaluate the overfit measure. Here the colourbar indicates the intensity of overfit (the overfit measure) - depending on the number of data points N and the level of the noise. Red indicates more overfitting, and blue indicates no overfitting.

The results

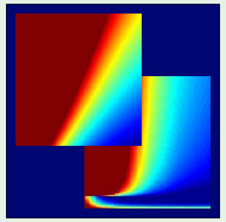




Impact of σ^2

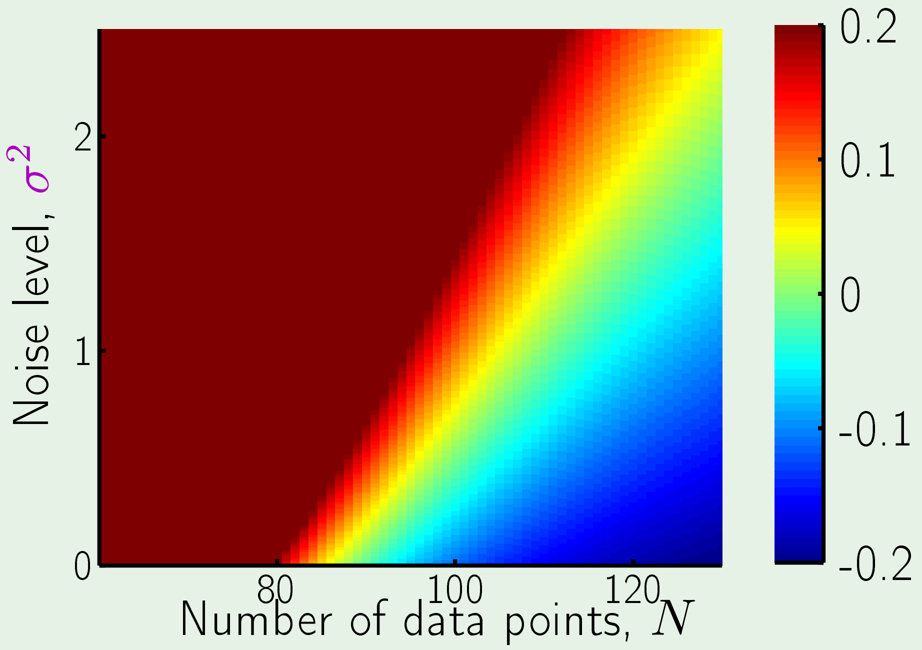
In this graph, Qf is fixed at 20

Impact of Q_f

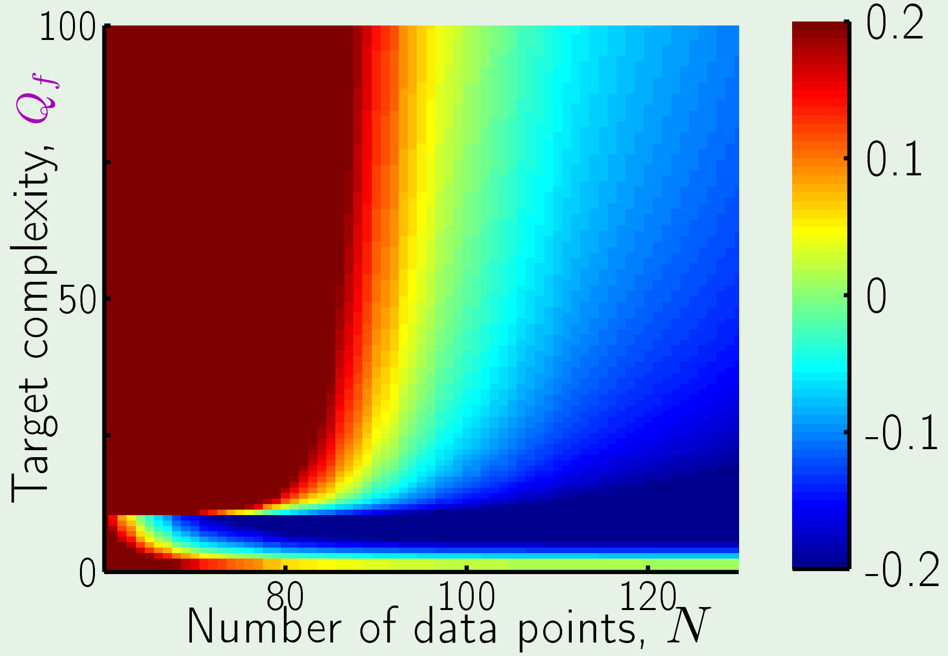


In this graph, the noise level is fixed at sigma^2 = 0.1. Note that, where overfitting is concerned, the target complexity Qf behaves in a similar way to the noise level sigma^2.

Impact of "noise"



Stochastic noise



Deterministic noise

number of data points	↑	Overfitting	↓
stochastic noise	↑	Overfitting	↑
deterministic noise	↑	Overfitting	↑

Outline

- What is overfitting?
- The role of noise
- Deterministic noise
- Dealing with overfitting

Definition of deterministic noise

(In this case we have already chosen H given prescriptions of N and VC dimension and other considerations)

The part of f that $\mathcal{H}$ cannot capture: $f(\mathbf{x}) - h^*(\mathbf{x})$

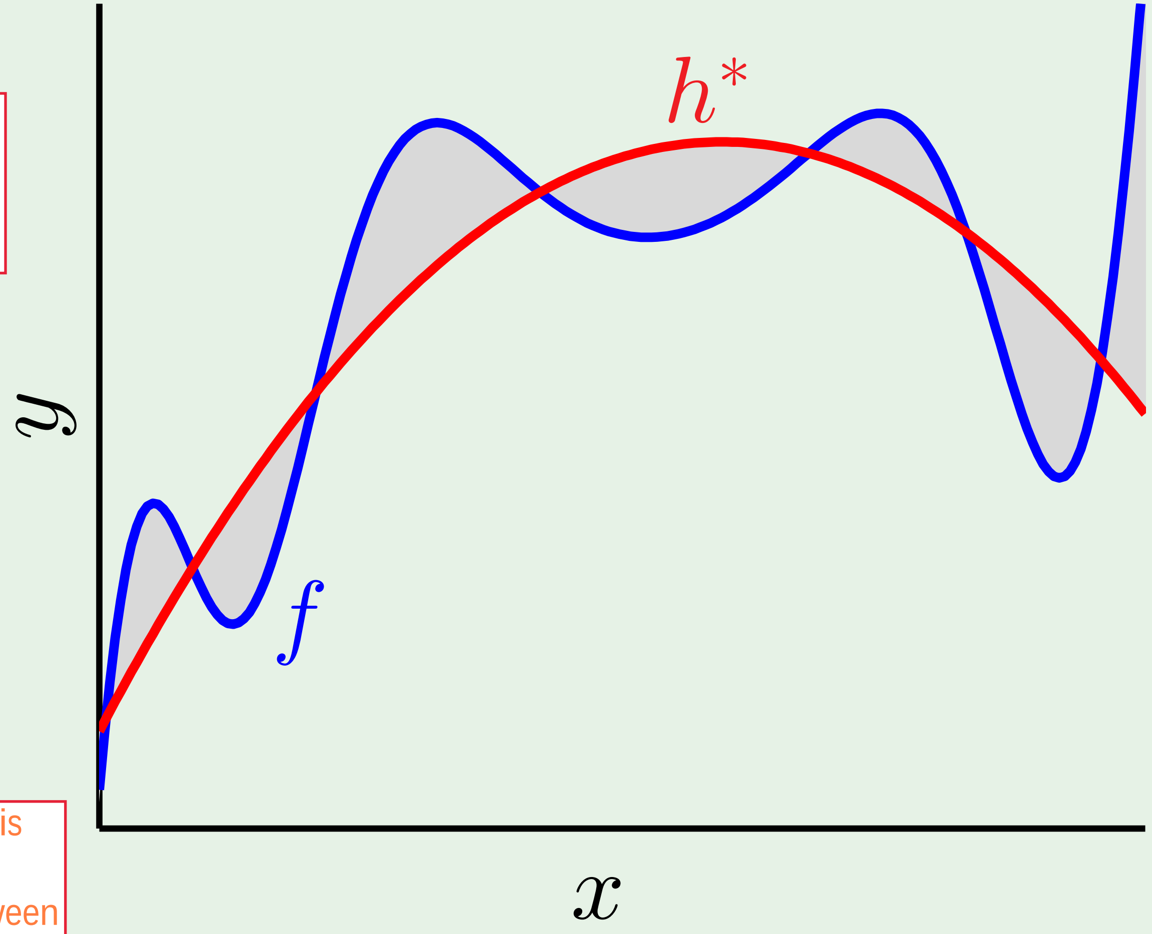
So deterministic noise "looks like noise" to some hypothesis sets but does not look like noise to others and experimentally this affects overfitting

Why "noise"?

Stochastic and deterministic noise are both unreachable, as far as our resources are concerned - our failure to be able to capture f with H makes it noise. Even in a stochastically "noiseless target", here we conceptually identify something which can affect overfitting and so we have to put in place measures to avoid overfitting in these instances.

Main differences with stochastic noise:

1. depends on $\mathcal{H}$
2. fixed for a given $\mathbf{x}$



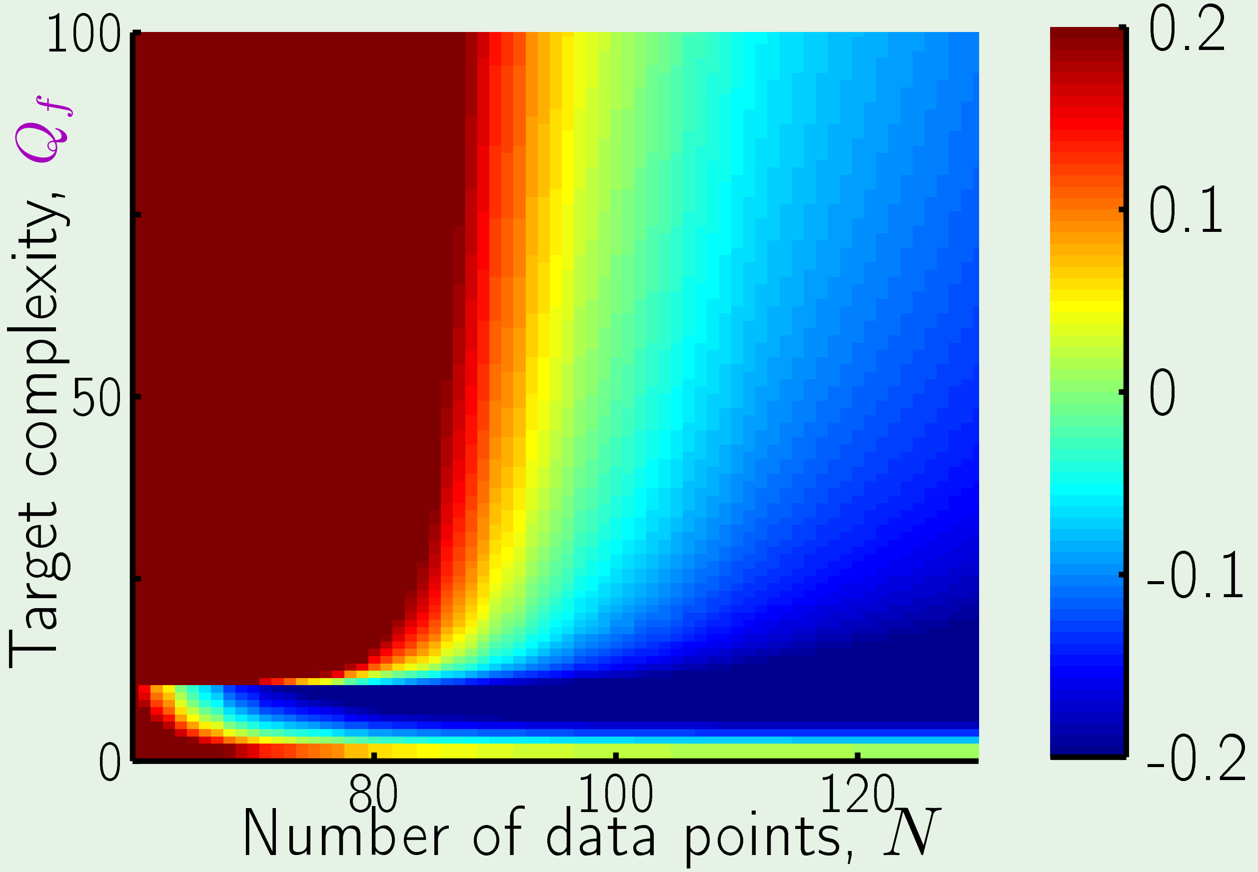
Note we cannot characterize an "energy" of this noise like we can with sigma^2 for stochastic noise, partly because we normalize f to be between 0 and 1, so the amount of energy of whatever the deterministic noise is will be limited (regardless of the complexity of the target).

Impact on overfitting

Deterministic noise and Q_f

Finite N: $\mathcal{H}$ tries to fit the noise

The finite dataset means that the hypothesis set may be able to capture a small amount of stochastic or deterministic noise. With more data points, the model is less affected by stochastic noise as it cannot fit it. With a smaller dataset, our model gains the unfortunate ability of being able to fit both the stochastic and deterministic noise, and so the model will fit it - even if it is deterministic where there is no point in fitting it because we know that our model it is out of our ability and we have no way to generalize out of sample for it



how much overfit

Noise and bias-variance

Recall the decomposition:

$$\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x})\right)^2\right] = \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x})\right)^2\right]}_{\textbf{var}(\mathbf{x})} + \underbrace{\left[\left(\bar{g}(\mathbf{x}) - f(\mathbf{x})\right)^2\right]}_{\textbf{bias}(\mathbf{x})}$$

What if f is a noisy target?

$$y = f(\mathbf{x}) + \epsilon(\mathbf{x}) \qquad \mathbb{E}\left[\epsilon(\mathbf{x})\right] = 0$$

A noise term

$$\mathbb{E}_{\mathcal{D},\epsilon}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - \quad y \quad\right)^2\right] = \mathbb{E}_{\mathcal{D},\epsilon}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x}) - \epsilon(\mathbf{x})\right)^2\right]$$

$$= \mathbb{E}_{\mathcal{D},\epsilon}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x}) + \bar{g}(\mathbf{x}) - f(\mathbf{x}) - \epsilon(\mathbf{x})\right)^2\right]$$

$$= \mathbb{E}_{\mathcal{D},\epsilon}\Big[\left(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x})\right)^2 + \left(\bar{g}(\mathbf{x}) - f(\mathbf{x})\right)^2 + \left(\epsilon(\mathbf{x})\right)^2$$

$$+ \quad \text{cross terms} \Big]$$

Cross terms evaluate to zero since the additional ones which now include epsilon become zero (since E(epsilon) = 0 and it is independent of D, since although the dataset and noise is generated, epsilon is generated on each x (which are all independent), so we get zero)

We then take the expectation value w.r.t. x (as in the previous decomposition) to get the variance and bias independent of x

Actually, two noise terms

Last term characterizes the distance from f, the target proper, to the actual output (which has a noise aspect to it)

$$\ldots \quad \underbrace{\mathbb{E}_{\mathcal{D},\mathbf{x}}\left[\left(g^{(\mathcal{D})}(\mathbf{x})-\bar{g}(\mathbf{x})\right)^2\right]}_{\text{var}}+\underbrace{\mathbb{E}_{\mathbf{x}}\left[\left(\bar{g}(\mathbf{x})-f(\mathbf{x})\right)^2\right]}_{\text{bias}}+\underbrace{\mathbb{E}_{\epsilon,\mathbf{x}}\left[\left(\epsilon(\mathbf{x})\right)^2\right]}_{\sigma^2}$$

$\uparrow$ deterministic noise

$\uparrow$ stochastic noise

The bias characterizes the ability of our best possible hypothesis to approximate f, so this can be seen as the energy of the deterministic noise.

This presentation of the variance and two types of noise allows us to treat the noises in the same way - increasing the number of examples gives us a better variance (the red region representing the variance shrinks), the stochastic and deterministic noise are both inevitable (there is nothing we can do about stochastic noise, and the deterministic noise is fixed given a hypothesis set). Note that for a given dataset of N points, both noises have a finite value on the data points and the algorithm will try to fit the noise, so both noises will affect the variance by making the fit more susceptible to choosing different hypotheses depending on the dataset (since each noise will affect each dataset differently). We blindly follow the noise because we can't distinguish noise from signal, and so we end up with more variety/possible choices of g, so larger red region, so worse variance and overfit - the problem of overfitting happens in the variance term. Note that the noise terms values are static and do not change with N, so they do not themselves contribute to the overfitting aspect. The point is that, the exact values themselves do not reflect the overfitting severity. You cannot say that a smaller value of bias leads to less overfitting. A model may get a smaller bias by overfitting it to the samples. At the same time, the bias value does not reflect the deterministic noise level. The latter is reflected by the complexity of the target (when it is above the model complexity): the higher complexity, the more noise, hence more overfitting.

Outline

- What is overfitting?
- The role of noise
- Deterministic noise
- Dealing with overfitting

Two cures

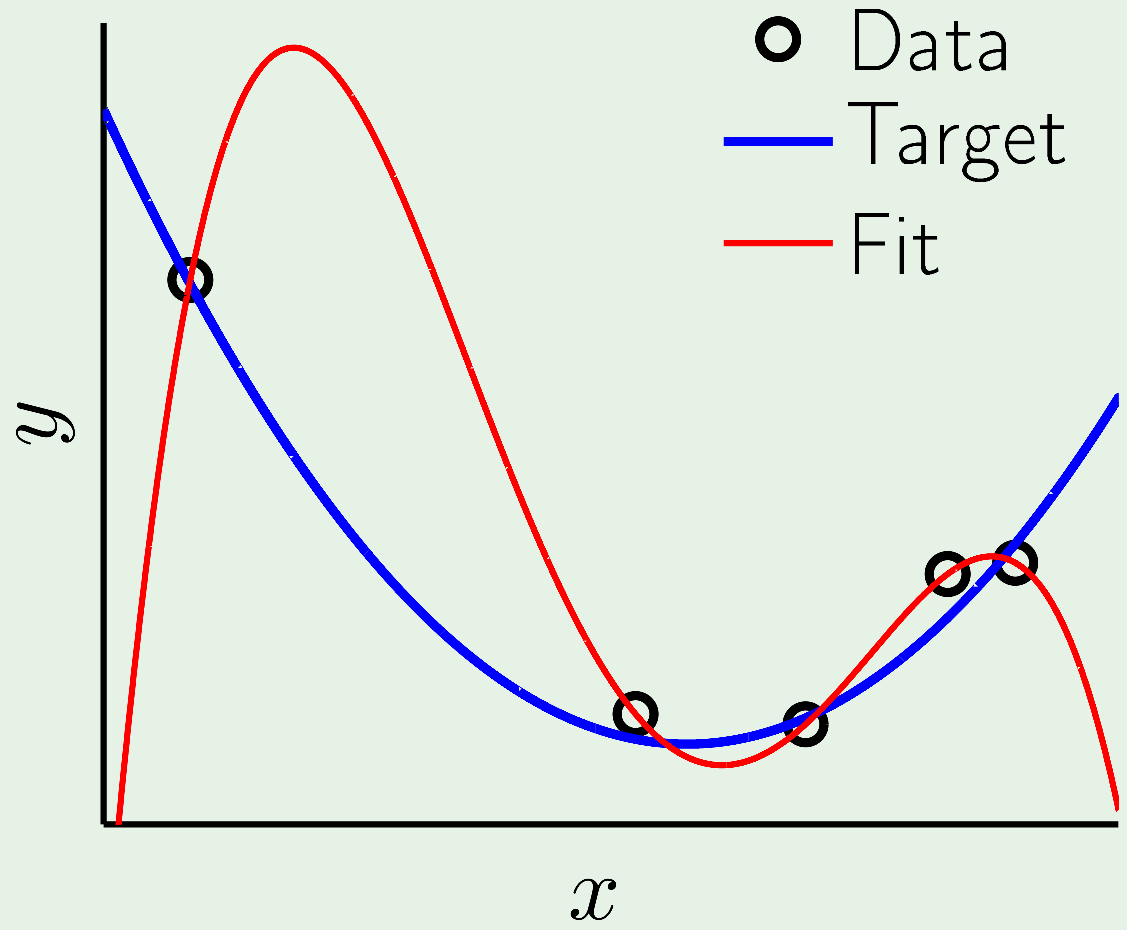
Regularization: Putting the brakes

stop further fitting/minimization of Ein to prevent Eout becoming worse

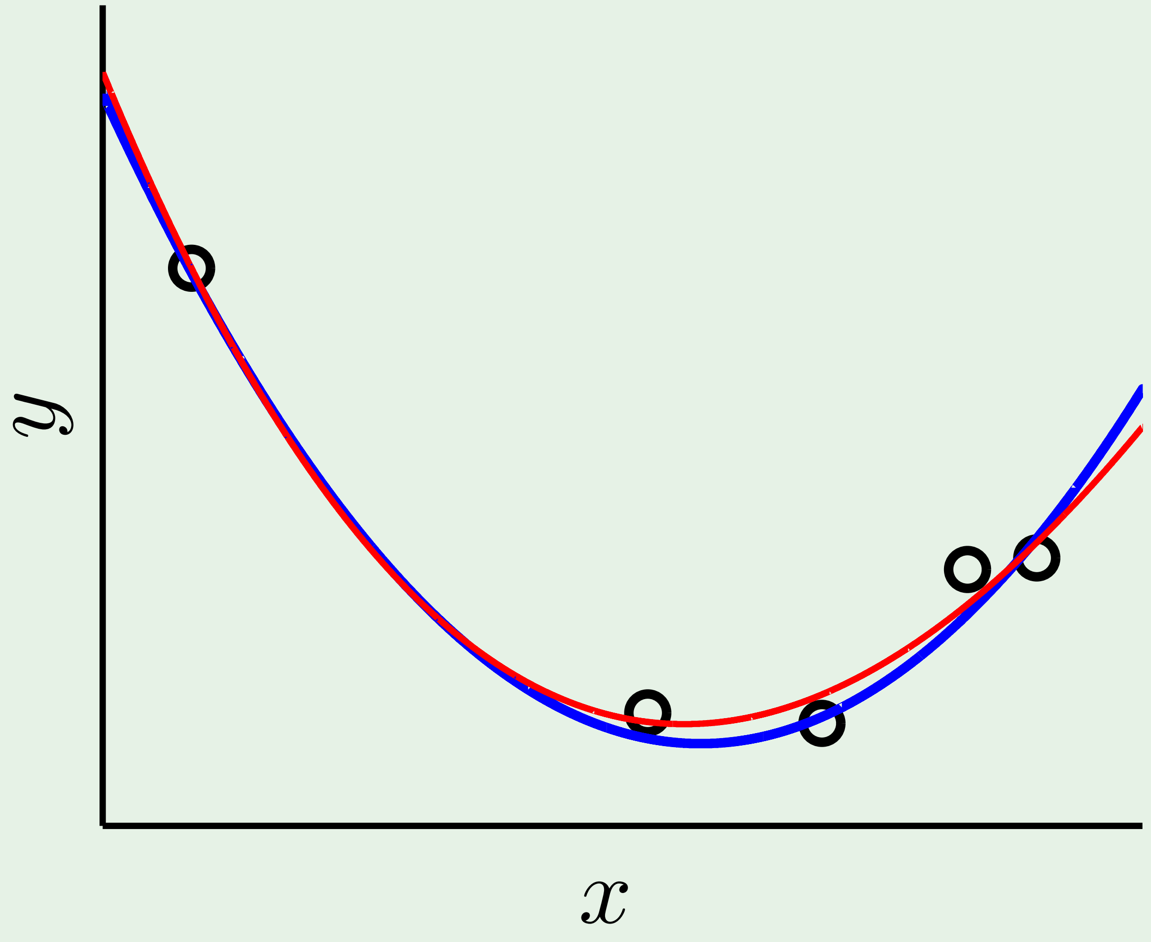
Validation: Checking the bottom line

Since Ein is not a very good indicator of generalization, we want another way of checking the generalization ability/what is happening out of sample - so we can avoid overfitting by checking what is happening in the quantity we care about (Eout)

Putting the brakes



free fit



restrained fit

still using a 4th order polynomial, but we are prevented from fitting the points perfectly