Bias-variance tradeoff & Regularisation

Luca Citi
lciti@essex.ac.uk

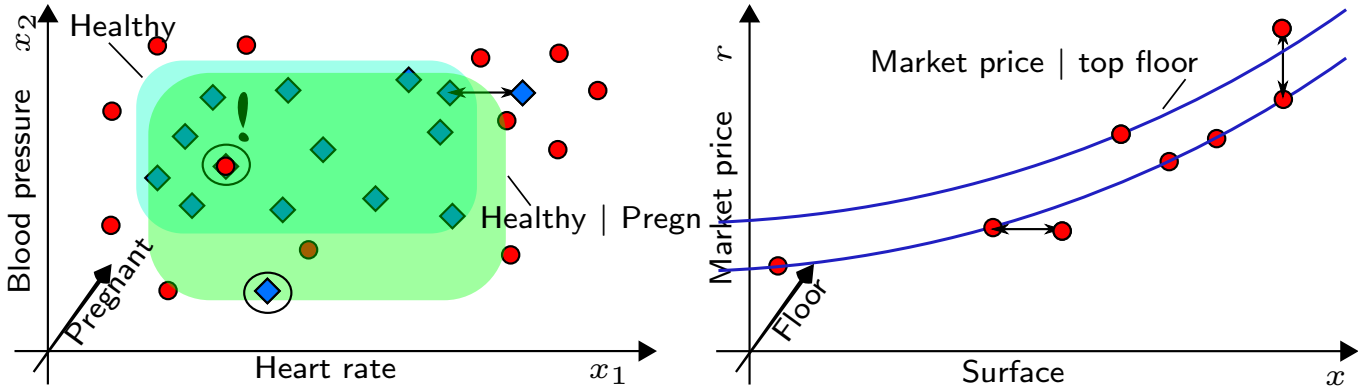
School of Computer Science and Electronic Engineering
University of Essex (UK)

CE802

Outline

Noise in the data

- In machine learning, data typically contain errors
- Possible causes:
  - measurement errors, human mistakes and other sources of imprecision in recording the input attributes;
  - measurement errors, errors of expert judgement in classifying training examples and other sources of imprecision affecting the target value (teacher noise);
  - effect of additional attributes that affect the label of an instance but are unavailable to the machine learner (neglected/hidden/unobservable); these can be modelled as source of random error
- We refer to all of these as noise

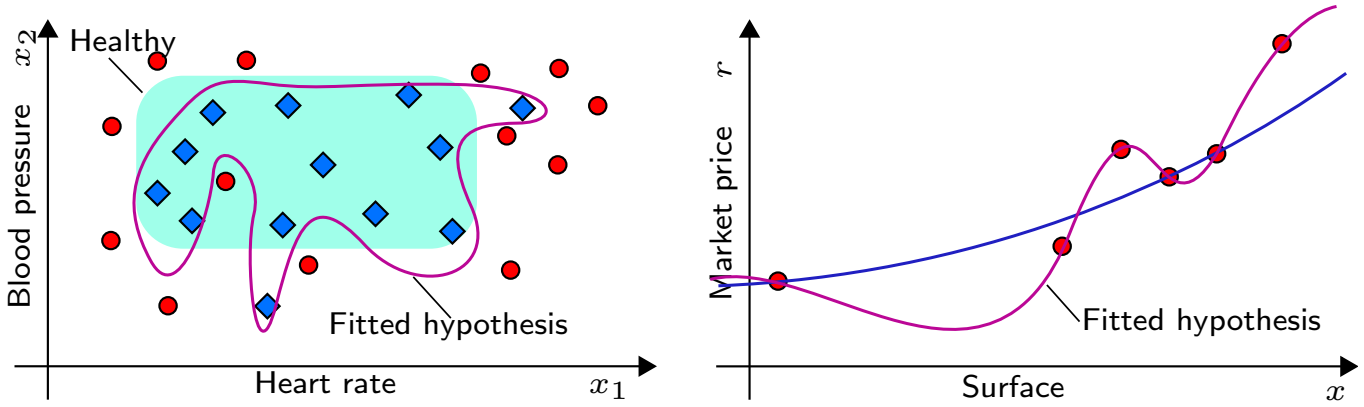


Consequences of noise

The main consequence of noise in the training data is overfitting that occurs when the fitted hypothesis reflects the random error instead of the underlying relationship between the features and the target.

In turn, overfitting causes:

- Solutions that are more complex than necessary and hard to interpret; e.g., noise-induced growth of DTs
- Solutions that don't generalize, i.e., poor accuracy on new unseen data



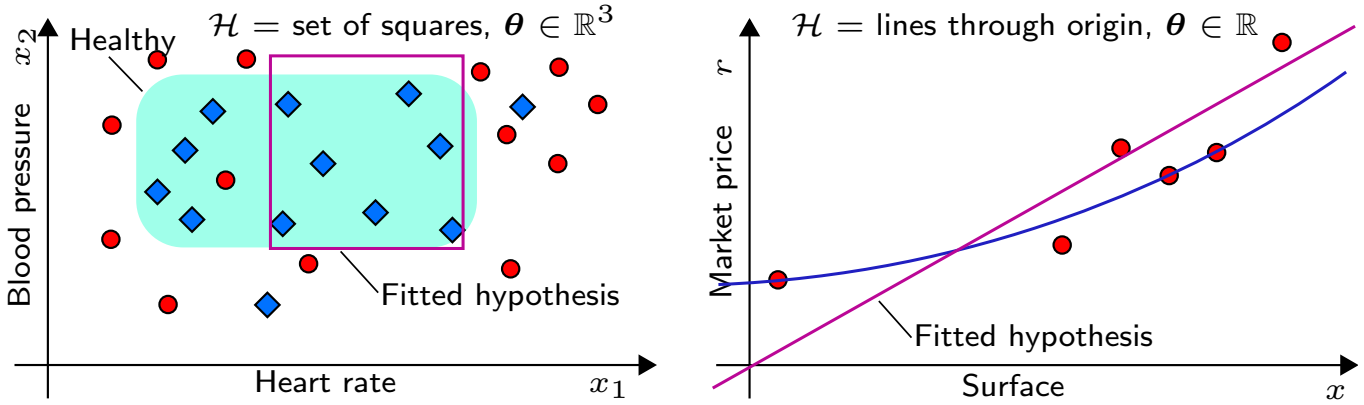
Alleviate effect of noise

To alleviate these harmful effects of noise, we have to prevent overfitting.

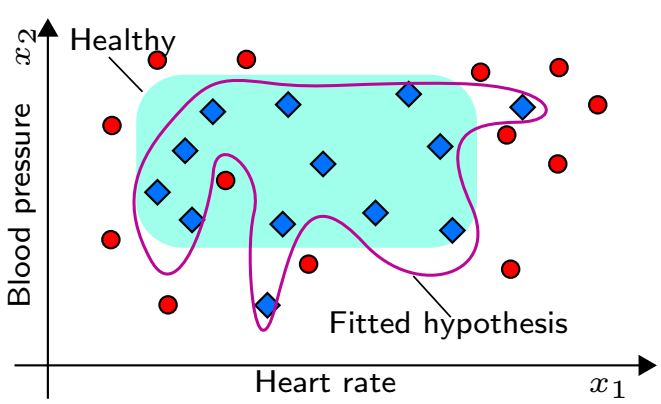
The common approach is to simplify induced hypotheses, e.g.:

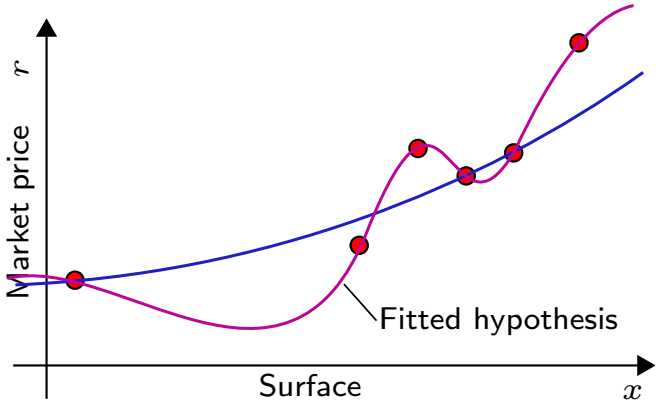
- Using Naïve Bayes instead of full Bayes classifiers
- Pruning DTs
- Using k-NN with $k > 1$
- In general, use models requiring a smaller number of parameters that need be estimated from the data

On the other hand, if the hypothesis is too simple, we incur in the opposite type of error: underfitting

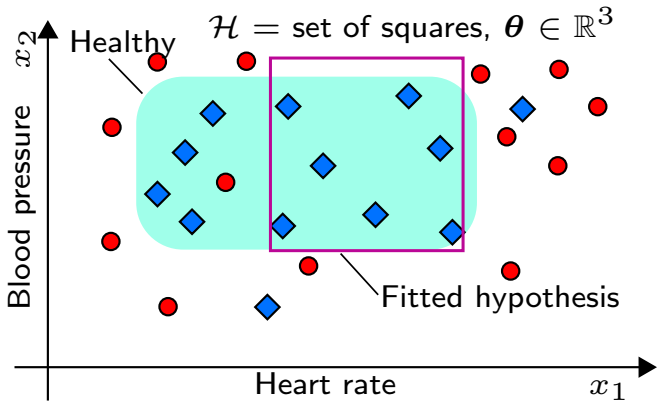


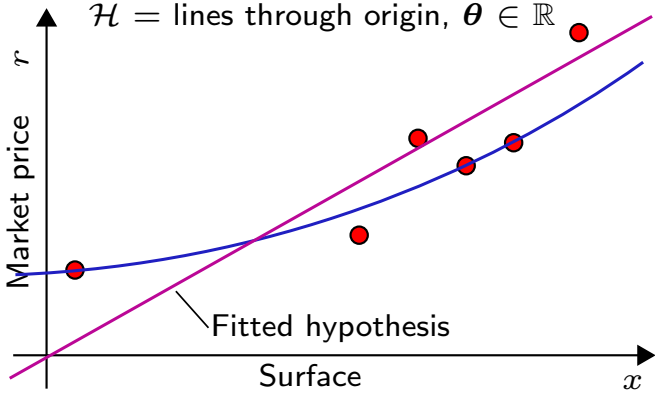
Overfitting VS Underfitting





Q1: How can we make sure we choose the hypothesis of "the right size", not too simple and not too complex?





Errors in predictions

After analysing possible sources of error in the training data, we are interested in the errors we make when predicting unseen examples.

Possible sources of error:

- Same errors as in the training data (after all, we made the assumption that training data and test data are drawn from the same population)
- Errors due to a model that underfits the training data
- Errors due to a model that overfits the training data

Q2: Can we quantify the errors we make when predicting unseen examples?

In the following, we do this for the case of linear regression but the same considerations hold for any ML algorithm

Nice interactive tutorial on the bias-variance trade-off for classification at: `http://scott.fortmann-roe.com/docs/BiasVariance.html`

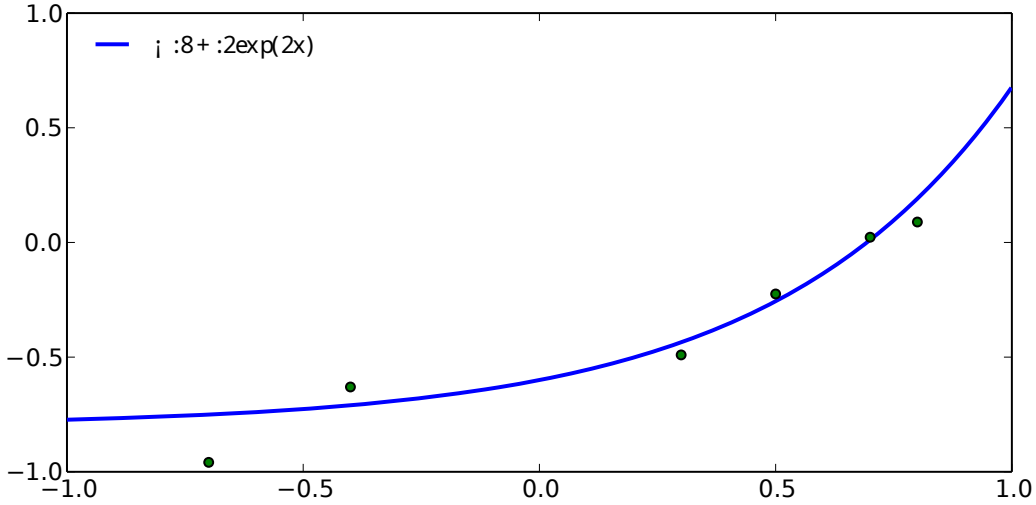
Example: linear regression

Assumption: all sources of noise on the training data (errors on features, errors on targets, errors due to unobserved features) can be adequately modelled as an additive zero-mean uncorrelated observation noise

x	$\dots$	x_{M}	r
0.4	$\dots$	0.8	0.5
0.1	$\dots$	0.5	0.3
$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\boldsymbol{x}^{(t)}$			$r^{(t)} \leftarrow f(\boldsymbol{x}^{(t)}) + n^{(t)}$
$\vdots$	$\ddots$	$\vdots$	$\vdots$

1

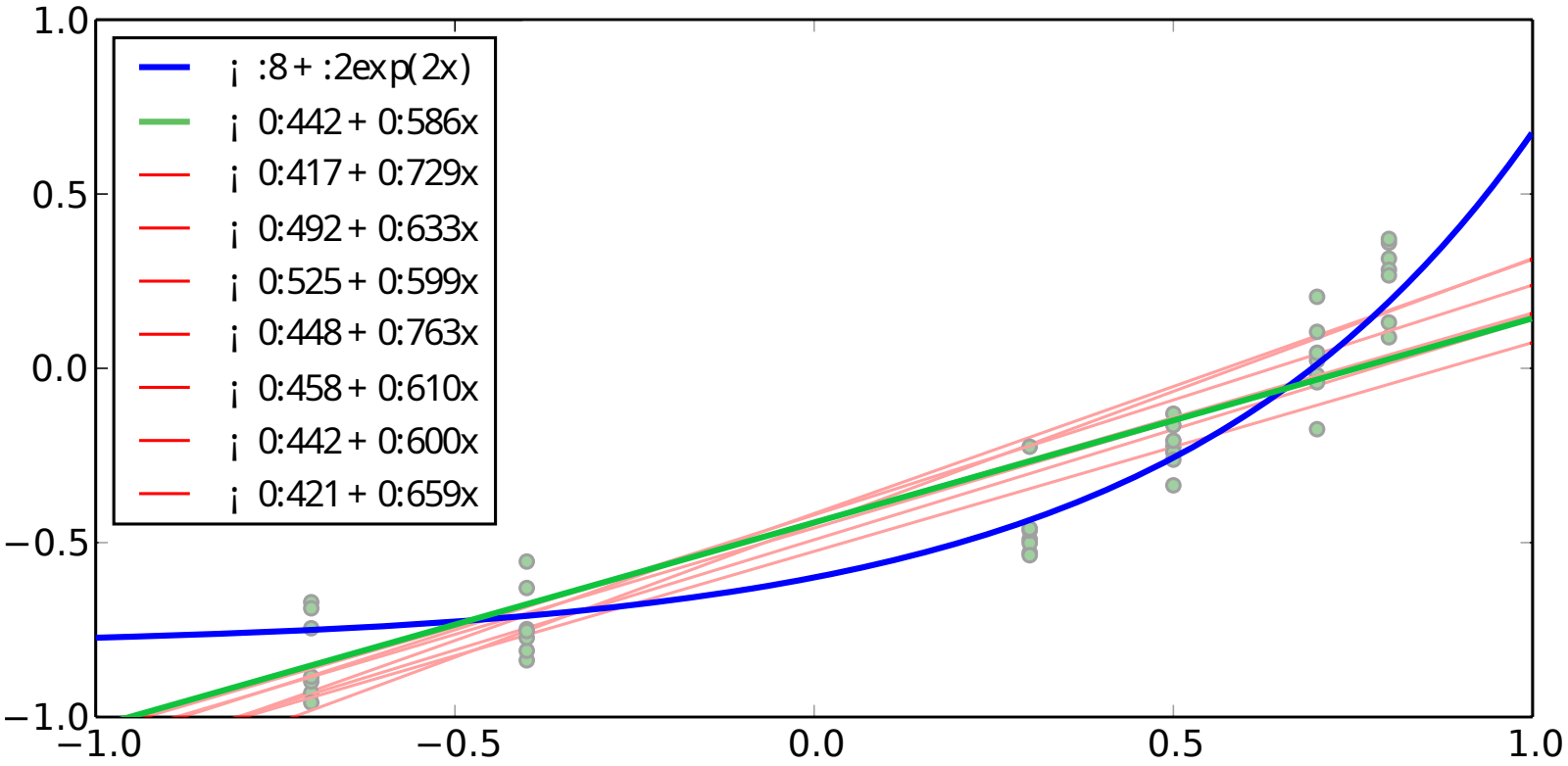
We further simplify by considering a single feature.
Let $f(x) = 0.2\exp(2x) - 0.8$



[1] As in the following we will use the superscript to indicate powers, we use $\boldsymbol{x}^{(t)}$ and $r^{(t)}$ to indicate the features and target of the t-th training instance

Fitting a straight line

Set of linear functions: $\mathcal{H} = \{h(\boldsymbol{x}|\boldsymbol{\theta})\}_{\boldsymbol{\theta}\in\mathbb{R}^2}$ with $h(\boldsymbol{x}|\boldsymbol{\theta}) = \theta_0 + \theta_1 x$

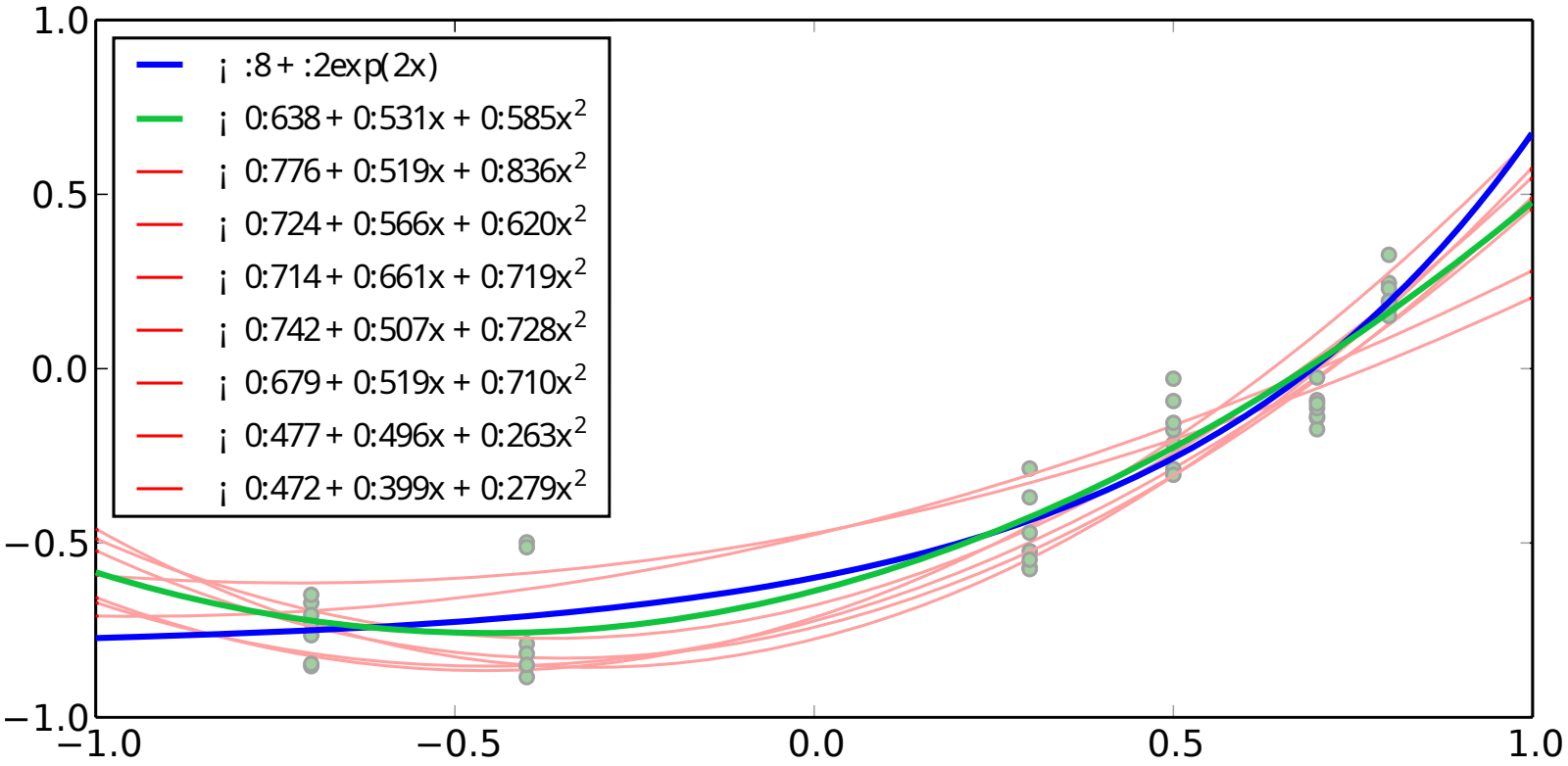


Training using a different training set gives slightly different solutions: overfitting

If we collect (infinitely) many training sets and take the average of all solutions, we still make an error (e.g., $\bar{h}(0) = -0.442 \neq -0.6 = f(0)$): underfitting

Fitting a quadratic polynomial

Set of quadratic polynomials: $\mathcal{H} = \{h(\boldsymbol{x}|\boldsymbol{\theta})\}_{\boldsymbol{\theta}\in\mathbb{R}^3}$ with $h(\boldsymbol{x}|\boldsymbol{\theta}) = \theta_0 + \theta_1 x + \theta_2 x^2$

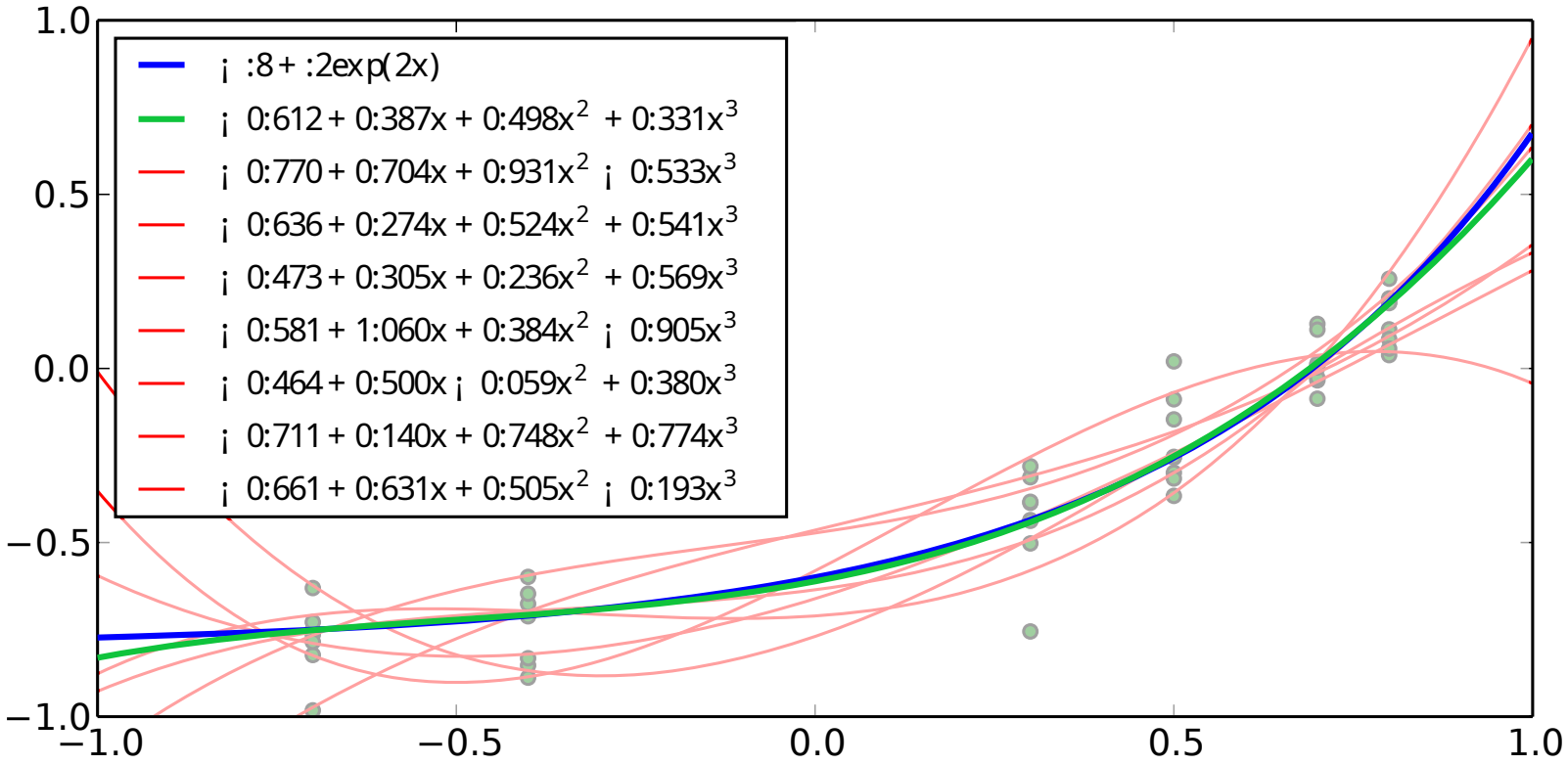


It seems like increasing the degree of the polynomial (i.e. the complexity of the model) makes the problem of overfitting worse

OTOH, there is less underfitting now: e.g., $\bar{h}(0)$ is closer to $f(0)$ than before

Fitting a cubic polynomial

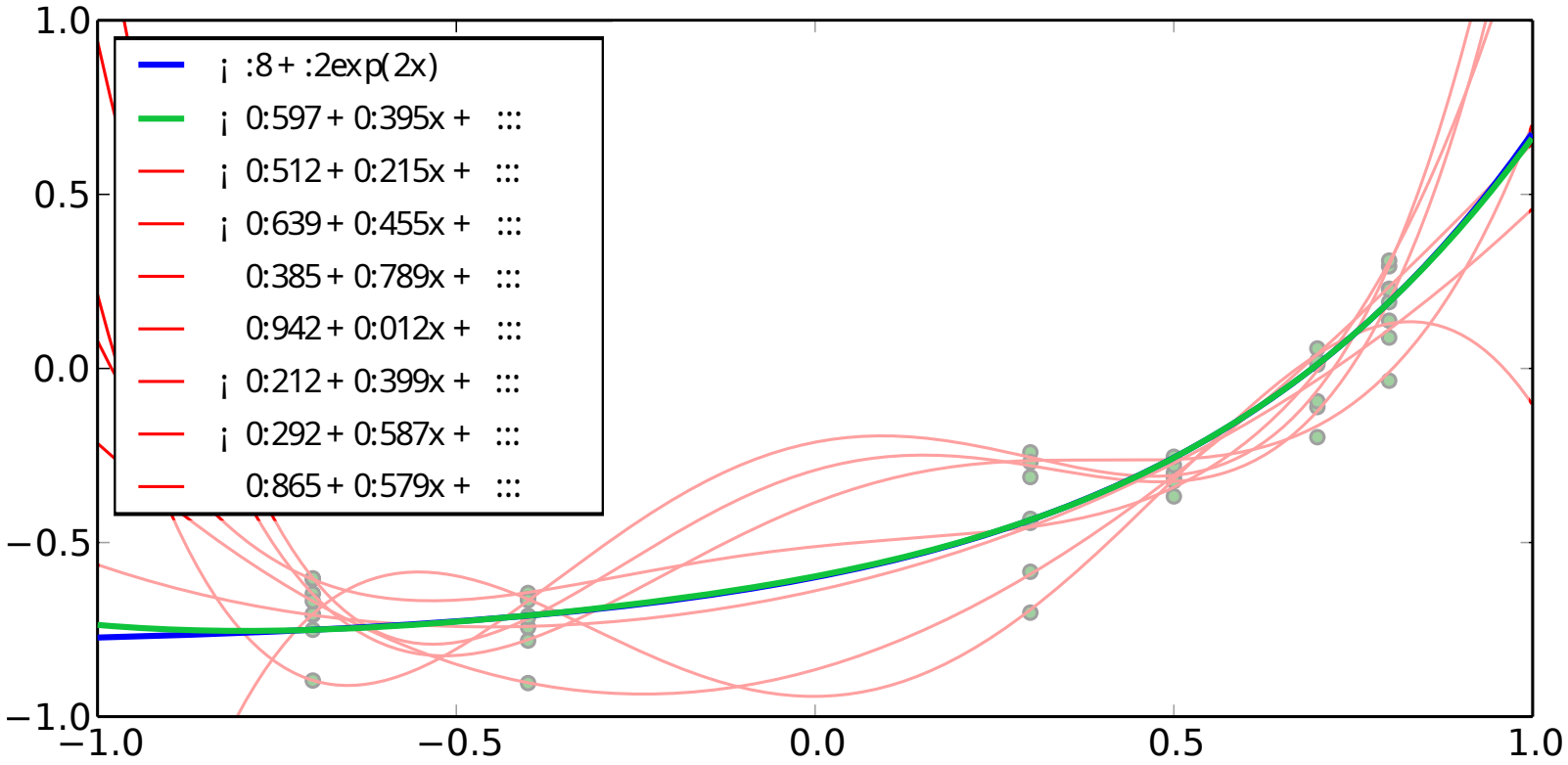
Set of cubic polynomials: $\mathcal{H} = \{h(\boldsymbol{x}|\boldsymbol{\theta})\}_{\boldsymbol{\theta}\in\mathbb{R}^4}$ with $h(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{i=0}^{3} \theta_i x^i$



... also in this case ...

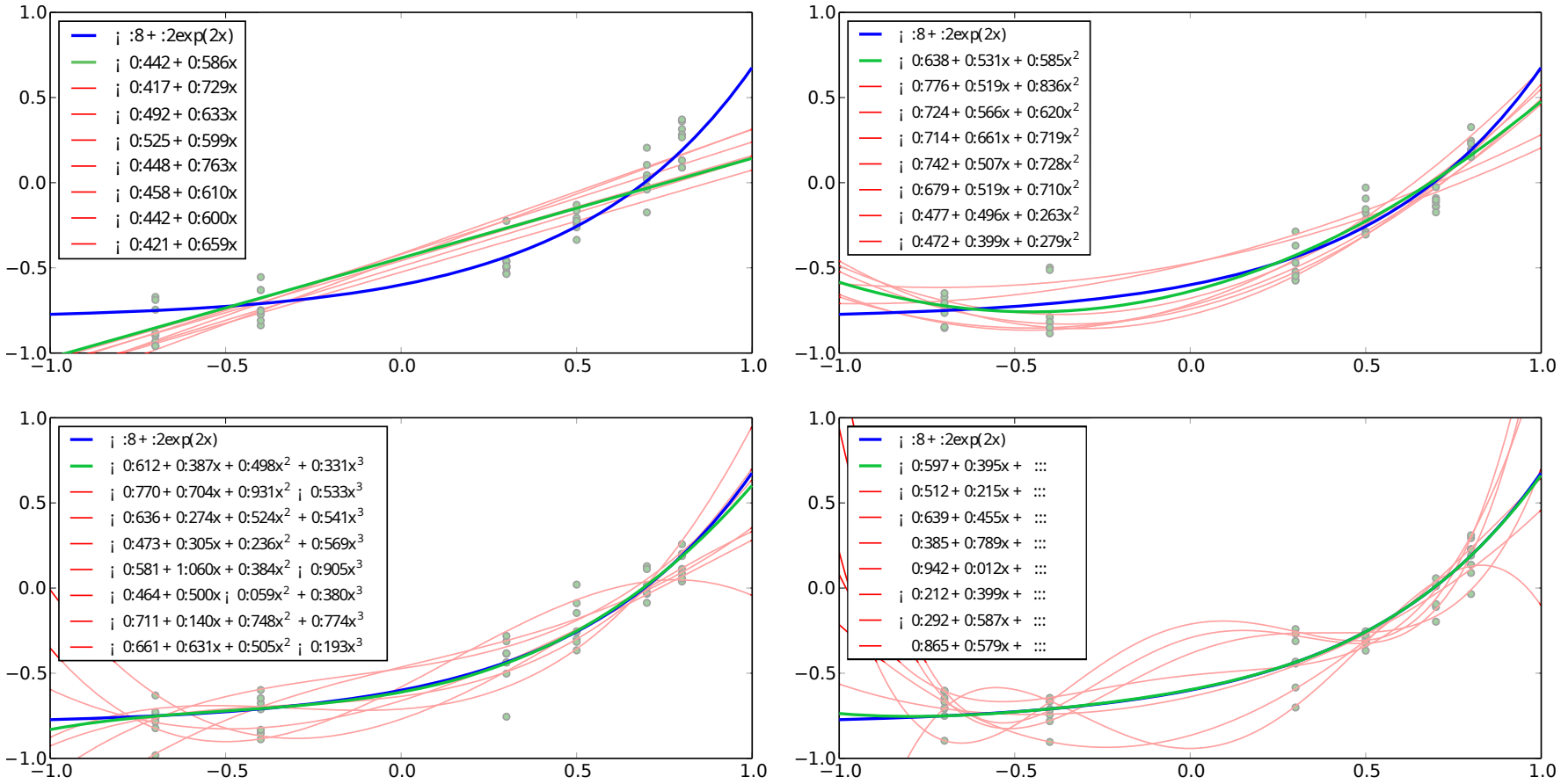
Fitting a 4th order polynomial

Set of 4th order polynomials: $\mathcal{H} = \{h(\boldsymbol{x}|\boldsymbol{\theta})\}_{\boldsymbol{\theta}\in\mathbb{R}^5}$ with $h(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{i=0}^{4} \theta_i x^i$



... and in this case.

General rule



In general, increasing complexity reduces bias (i.e. the error related to underfitting) and increases variance (i.e. the error related to overfitting).

Expected squared prediction error

Let's suppose, for simplicity, that we are interested in the prediction error for a fixed value of x: $x^{\text{test}} = \tilde{x}$ (e.g., error in the prediction of the market price of a 100 m^2 apartment).[2]

We repeat the following experiment:

- Randomly sample a training set $\mathcal{X}$
- Fit a model $h(x)$
- Randomly draw a test point such that $x^{\text{test}} = \tilde{x}$ and record its target r^{test}
- Compute the squared prediction error $(r^{\text{test}} - h(x^{\text{test}}))^2$

What is the squared prediction error that we should expect on average?

We can define it using the "expected value" operator.

[2] Assuming a fixed value of x^{test} makes the working easier because we can work with the probability distribution of r^{test} (given x^{test}) rather than the joint distribution of $(x^{\text{test}}, r^{\text{test}})$ but the result we will obtain is general.

"Expected value" operator

Intuitively, the expected value (or expectation) is the average value that one would "expect" when carrying out an infinite number of independent repetitions of an experiment.

Given a random variable V, the expected value is denoted $\mathbb{E}[V]$

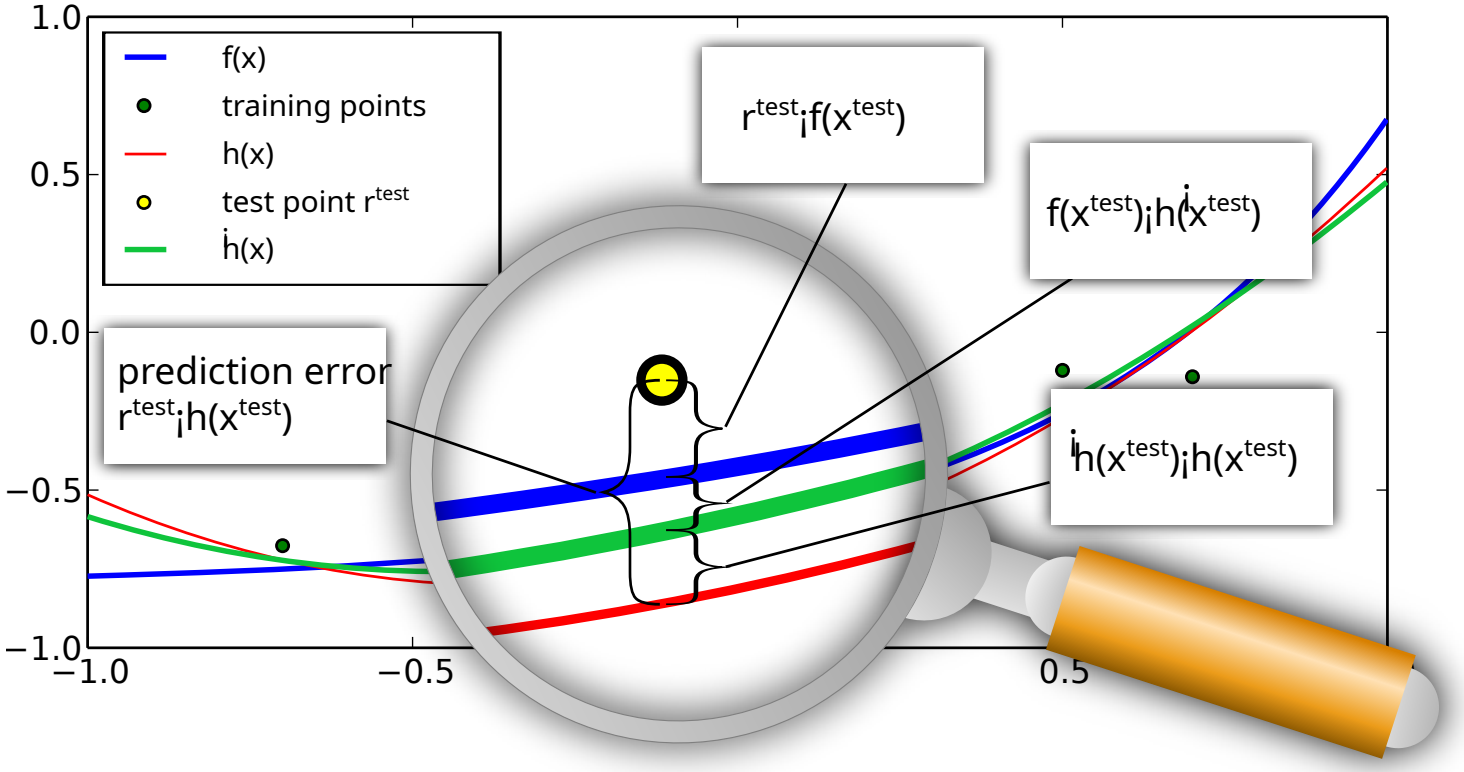
In general, $\mathbb{E}[f(V)] \neq f(\mathbb{E}[V])$

The following properties hold (V and W are random variables; c is deterministic):

- $\mathbb{E}[c] = c$
- $\mathbb{E}[V + W] = \mathbb{E}[V] + \mathbb{E}[W]$
- $\mathbb{E}[cV] = c\,\mathbb{E}[V]$
- V and W are independent r.v. $\Rightarrow \mathbb{E}[VW] = \mathbb{E}[V]\,\mathbb{E}[W]$

Decomposing the prediction error

From a training set..., we fit a model $h(x)$. If we are given a new test data point $(x^{\text{test}}, r^{\text{test}})$, what prediction error $r^{\text{test}} - h(x^{\text{test}})$ should we expect?



Considering the (unknown) functions $f(x)$ and $\bar{h}(x)$..., the prediction error is

$$r^{\text{test}} - h(x^{\text{test}}) = [r^{\text{test}} - f(x^{\text{test}})] + [f(x^{\text{test}}) - \bar{h}(x^{\text{test}})] + [\bar{h}(x^{\text{test}}) - h(x^{\text{test}})]$$

Decomposing the squared prediction error (maths as ref.)

After renaming $r^{\text{test}} = r$ (for simplicity of notation) and using $x^{\text{test}} = \tilde{x}$, the expected squared prediction error can be written as:[3]

$$\mathbb{E}\left[\{r - h(\tilde{x})\}^2\right] = \mathbb{E}\left[\left\{[r - f(\tilde{x})] + [f(\tilde{x}) - \bar{h}(\tilde{x})] + [\bar{h}(\tilde{x}) - h(\tilde{x})]\right\}^2\right] =$$

$$\mathbb{E}\left[\{r - f(\tilde{x})\}^2\right] + \quad \leftarrow \text{noise contribution}$$

$$\mathbb{E}\left[\{f(\tilde{x}) - \bar{h}(\tilde{x})\}^2\right] + \quad \leftarrow \text{bias squared}$$

$$\mathbb{E}\left[\{\bar{h}(\tilde{x}) - h(\tilde{x})\}^2\right] + \quad \leftarrow \text{variance of the model}$$

$$2\,\mathbb{E}\left[\{r - f(\tilde{x})\}\{f(\tilde{x}) - \bar{h}(\tilde{x})\}\right] + \quad \leftarrow = \mathbb{E}\left[\ldots\right]\{\ldots\} = \{f(\tilde{x}) - f(\tilde{x})\}\{\ldots\} = 0$$

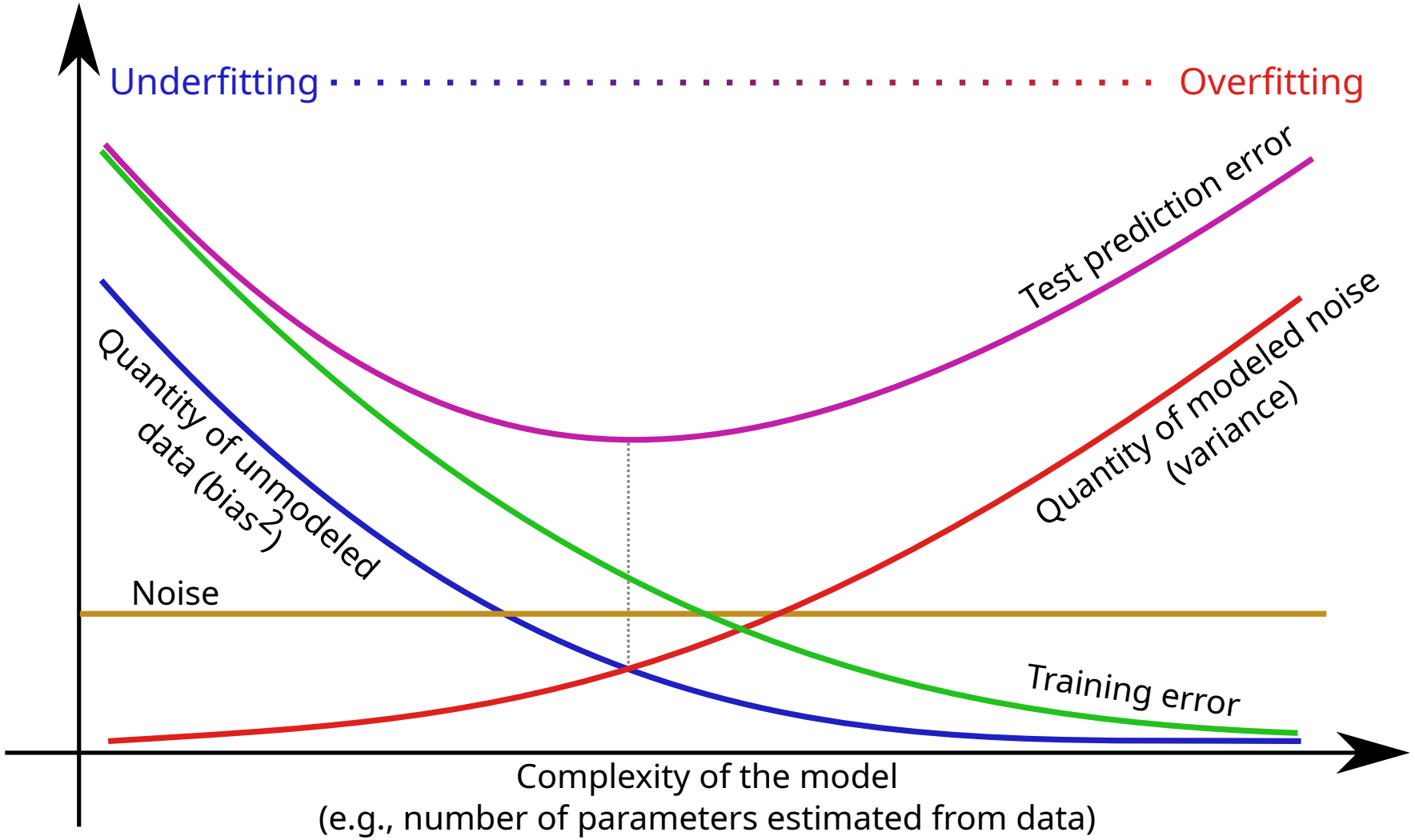
$$2\,\mathbb{E}\left[\{f(\tilde{x}) - \bar{h}(\tilde{x})\}\{\bar{h}(\tilde{x}) - h(\tilde{x})\}\right] + \quad \leftarrow = \{\ldots\}\,\mathbb{E}\left[\ldots\right] = \{\ldots\}\{\bar{h}(\tilde{x}) - \bar{h}(\tilde{x})\} = 0$$

$$2\,\mathbb{E}\left[\{\bar{h}(\tilde{x}) - h(\tilde{x})\}\{r - f(\tilde{x})\}\right] = \quad \leftarrow = \mathbb{E}\left[\ldots\right]\mathbb{E}\left[\ldots\right] = 0 \cdot 0 \text{ because indep 0-avg}$$

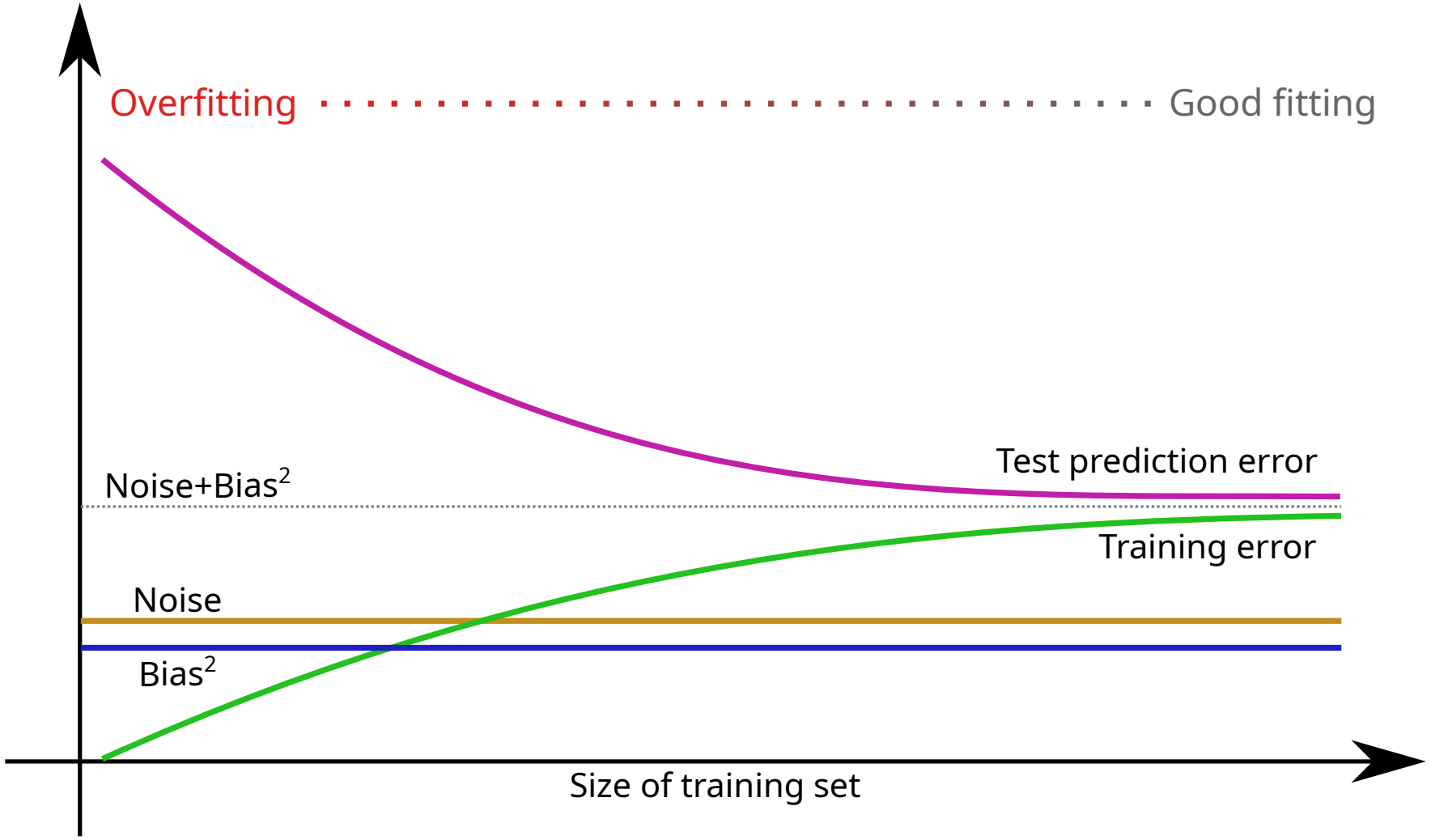
$$\underbrace{\mathbb{E}\left[\{r - f(\tilde{x})\}^2\right]}_{\text{noise}} + \underbrace{\{f(\tilde{x}) - \bar{h}(\tilde{x})\}^2}_{\text{bias}^2} + \underbrace{\mathbb{E}\left[\{\bar{h}(\tilde{x}) - h(\tilde{x})\}^2\right]}_{\text{variance}}$$

[3] Some of the terms involved are deterministic (in our setting), random because they depend on $\mathcal{X}$, or random because they depend on r

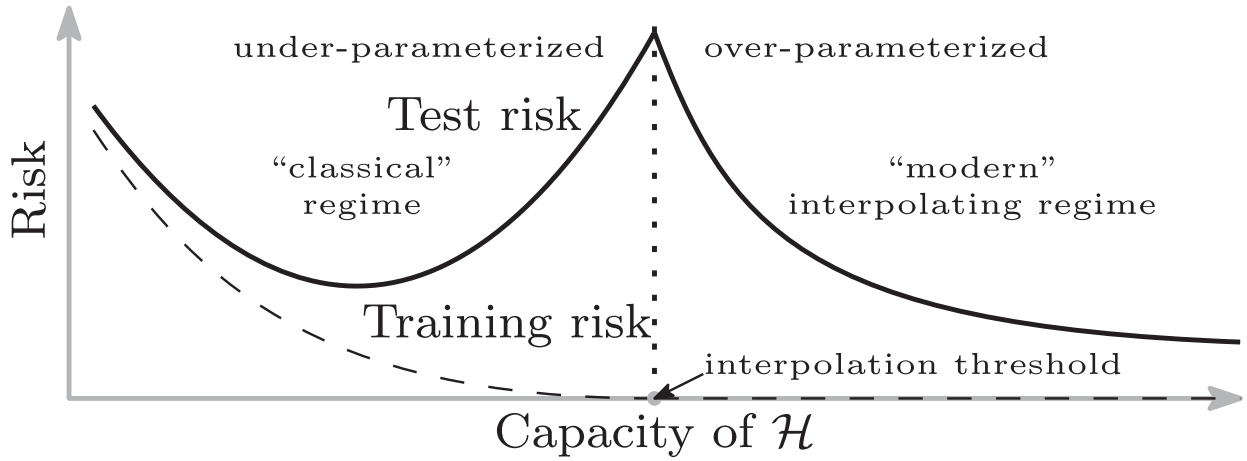
Bias and variance as a function of model complexity



Bias and variance as a function of training set size



Double descent in modern machine learning



Interesting and surprising recent results showing that:

- The "classical regime" holds on the left of the interpolation threshold (complexity just enough to interpolate data $\rightarrow$ zero training error)
- Past that point, we are in the "modern regime" where the test error decreases again, possibly due to implicit inductive bias of optimisers (e.g. SGD finds "shallow" as opposed to "sharp" minima)

Belkin, M., Hsu, D., Ma, S., & Mandal, S. (2019). "Reconciling modern machine-learning practice and the classical bias–variance trade-off." https://dx.doi.org/10.1073/pnas.1903070116

Outline

- Bias-variance tradeoff
  - Noise in the data
  - Errors in predictions
  - Example: linear regression
  - Expected squared prediction error
  - Decomposing the prediction error
  - Bias and variance as a function of model complexity
  - Bias and variance as a function of training set size
  - Double descent in modern machine learning
- Feature selection
- Regularisation

Feature selection

In most cases, data contain many redundant or irrelevant features. We generally prefer simpler models, using only relevant features, in order to:

- improve model interpretability;
- improve generalisation (by reducing overfitting/variance).

The process of selecting a subset of relevant features is called feature selection and requires:

- an evaluation measure to score the different feature subsets, e.g.:
  - R^2 in linear regression
  - accuracy in classification problems
  - or, more in general, a loss function $L(\cdot)$
- a search technique for proposing new feature subsets
  - exhaustive (unfeasible in most cases)
  - greedy forward selection
  - greedy backward elimination
  - genetic algorithms, ...

/

Greedy feature selection

Greedy feature selection schemes are applicable to most ML algorithms and they are normally performed using cross-validation:

- Greedy forward feature selection

```
Start with no attributes in the model
REPEAT
   FOR each attribute a not in current model
      train the model with current attributes plus a
      assess its performance on a separate dataset (or using CV)
   add the attribute that leads to biggest performance increase
UNTIL no further improvement
```

- Greedy backward feature elimination

```
Start with all attributes in the model
REPEAT
   FOR each attribute a in current model
      train the model without a
      assess its performance on a separate dataset (or using CV)
   remove attribute whose removal leads greatest performance increase
UNTIL no further improvement
```

Why is CV-based feat. sel. different from Stepwise Regression? See slide 18!

Outline

- Bias-variance tradeoff
  - Noise in the data
  - Errors in predictions
  - Example: linear regression
  - Expected squared prediction error
  - Decomposing the prediction error
  - Bias and variance as a function of model complexity
  - Bias and variance as a function of training set size
  - Double descent in modern machine learning
- Feature selection
- Regularisation

Regularisation: general idea

Feature selection is part of a more general model selection problem that aims at reducing the complexity of the fitted hypothesis.

In some cases, we can move model selection into the induction algorithm by using regularisation. This approach is usually more efficient (only requires fitting one model) and less greedy (under certain assumptions, it's a convex problem).

The general idea is simple: place a penalty for complexity (magnitude of the parameters, non-smoothness of the decision boundary or regression line, ...); then, solve for the new objective function.

Example: in genetic programming penalize individuals corresponding to long programs to prevent bloat.

References

Required course material reading:

Alpaydin 2010/2014 4.7,4.8(Cross validation)