Overfitting Intuitions

Lecture 2:
Over-fitting – the "enemy"

Why such *cheap* minimization algorithms ? (e.g. GD)

→ Do we *really* want to minimize $J(\theta, X_{train})$
The **true objective:** obtain **good performance** on new (unseen) data, which are *different*, but *similar*

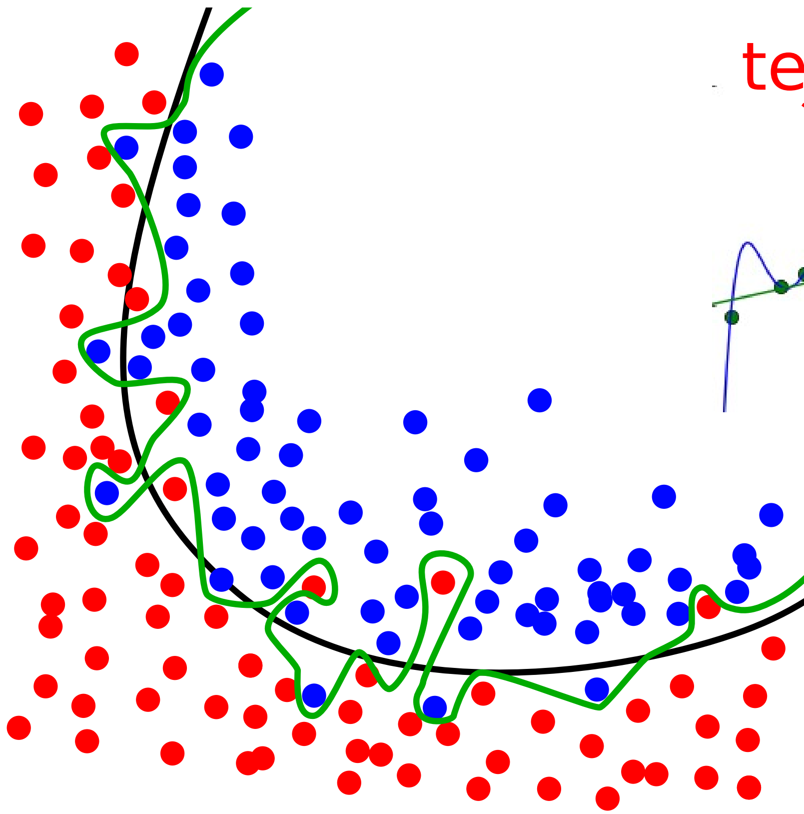
We want ***"generalization"***

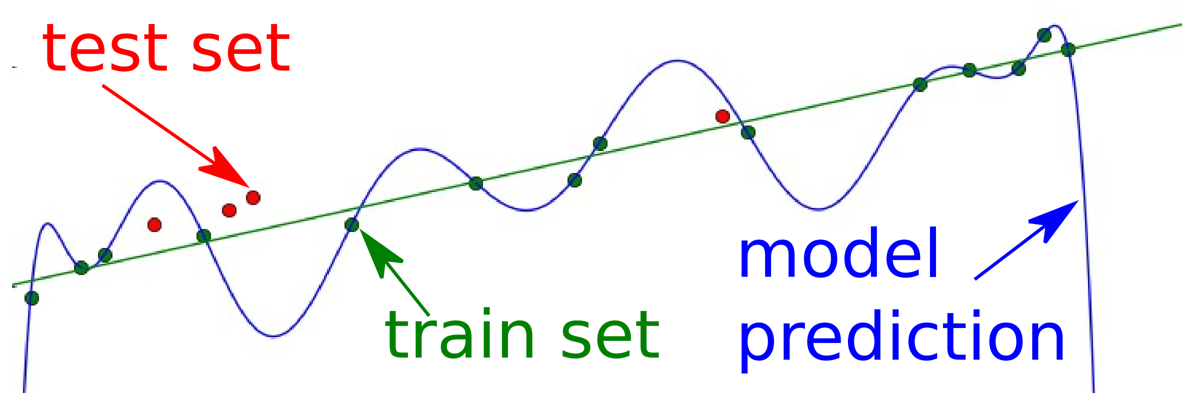
It's a very ***ill-posed*** problem !

- Example:
  Can we define the probability distribution of the *subspace of dog pictures*, within the space of *100x100 pixels RGB pictures*?
  → We can picture the space of *100x100 pixels RGB pictures*. It's a 3.10^4-dimensional hypercube, easy.
  → Subspace of *dog pictures* : **no** (unthinkable-of manifold, and it makes no human sense to define this mathematically)
  → We may just assume simple things, like, probably that manifold has a smaller *intrinsic dimension*. But it'll be difficult to measure, etc.
  → so instead, we use **data**

Over-fitting
intuitive definition

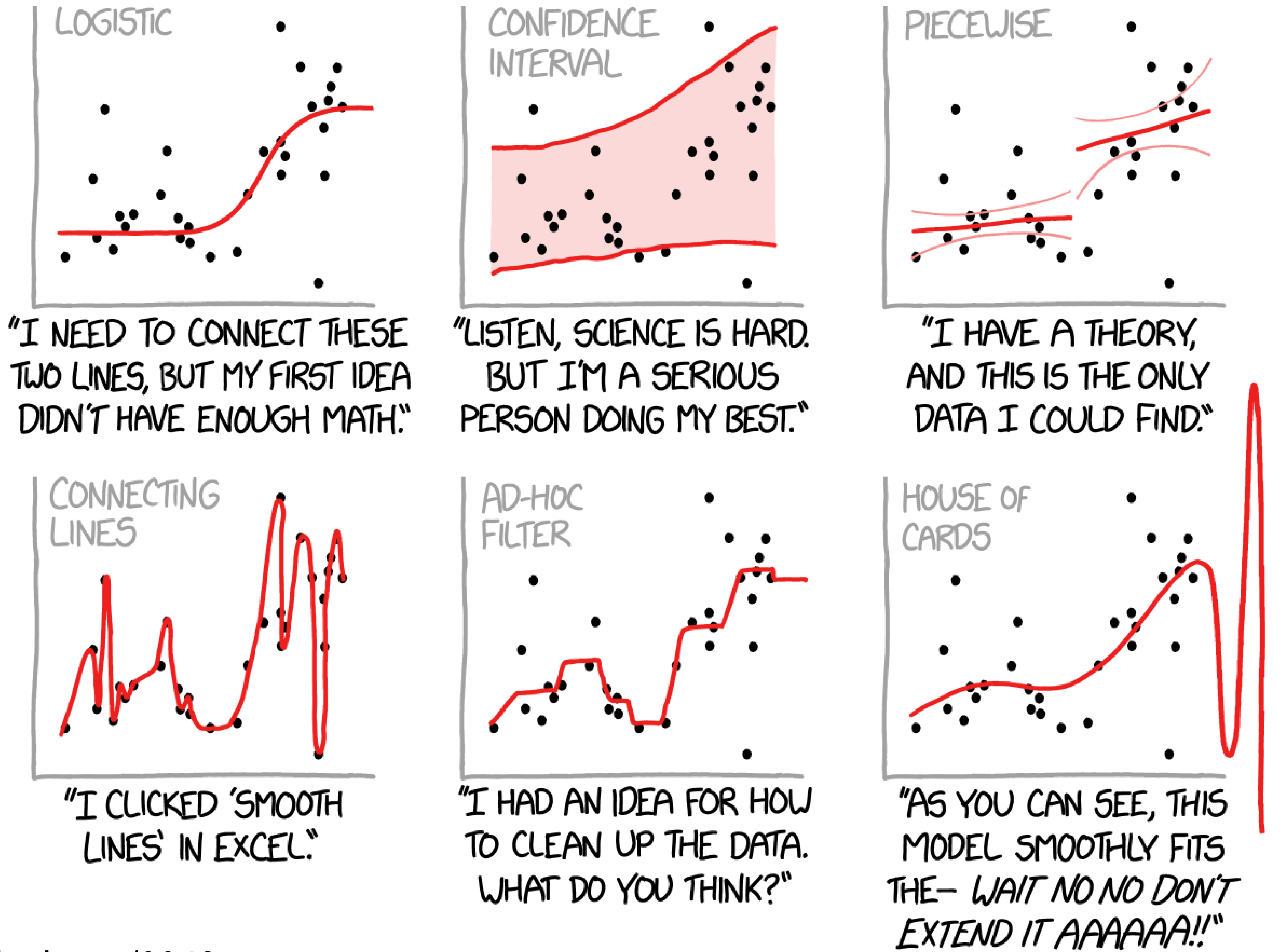
Reminder: N points are always ***exactly*** interpolated by an **Nth-order** polynomial.
→ Yes, but with **horrible over-fitting:**





We want to do well…
also on the ***test set*** !

curve-fitting: https://xkcd.com/2048/



xkcd.com/2048

Measuring the over-fitting: concept of ***Test set***

Over-fitting: visually, in 2D, easy to see

→ but how to characterize it **quantitatively** ?

With only N data points:

- We simulate the arrival of **new data** by setting aside some examples. N_{test}
- We **train** the model with the $N_{train} = N - N_{test}$ examples (opimization of the parameters *Θ*)
- We **test** the model (measure performance) on the "new" data N_{test}
(test: *model prediction* vs. *Ground Truth*)

Measuring the over-fitting: concept of ***Test set***

- Few errors ≃ good performance
- The difference between

  - ***train*** set error
  - ***test*** set error

  is <u>*a*</u> **measure of *over-fitting***

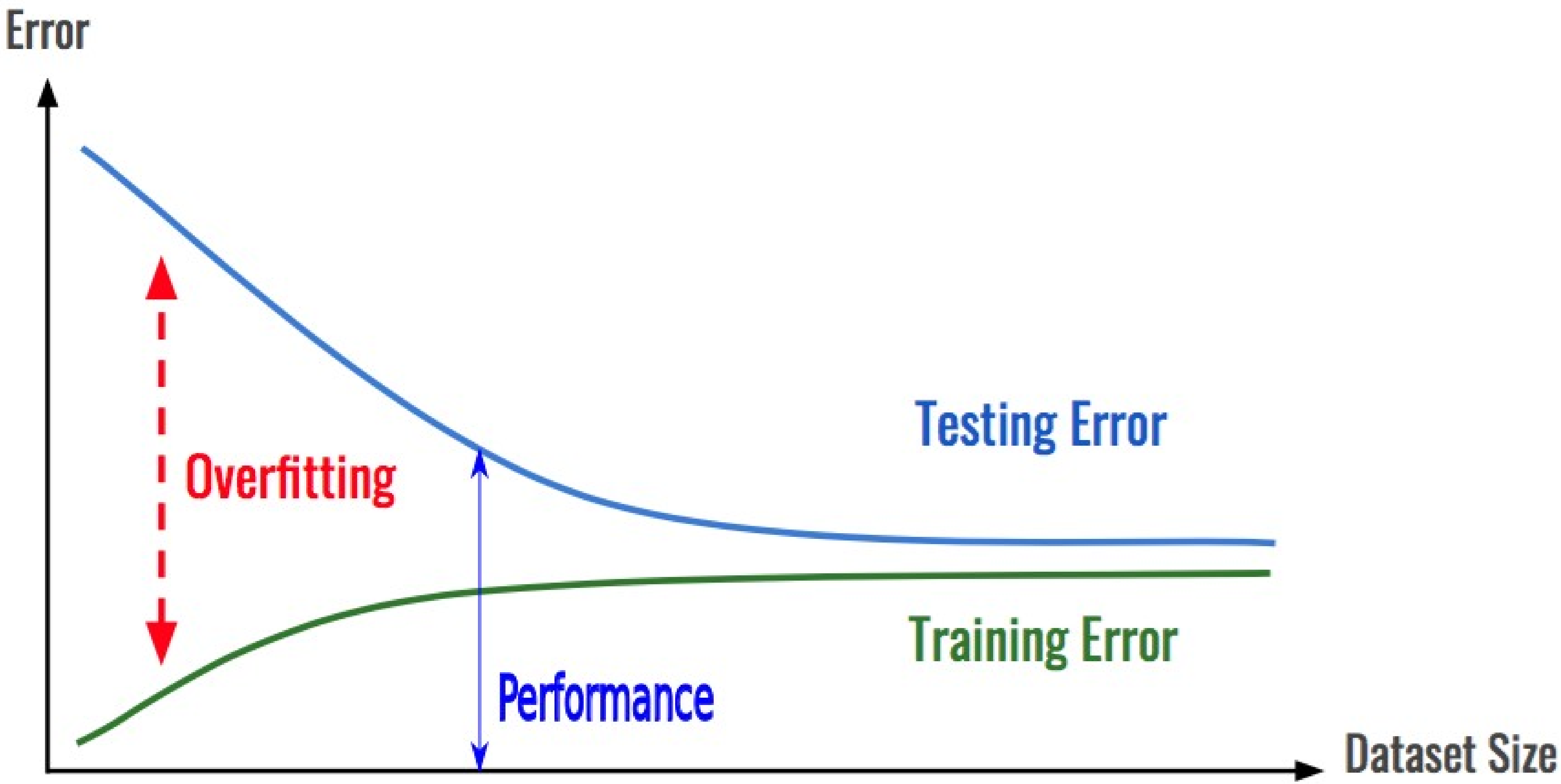
→ Low overfitting = good generalization

→ High overfitting = bad generalization

- **Amount of overfitting ≠ performance**

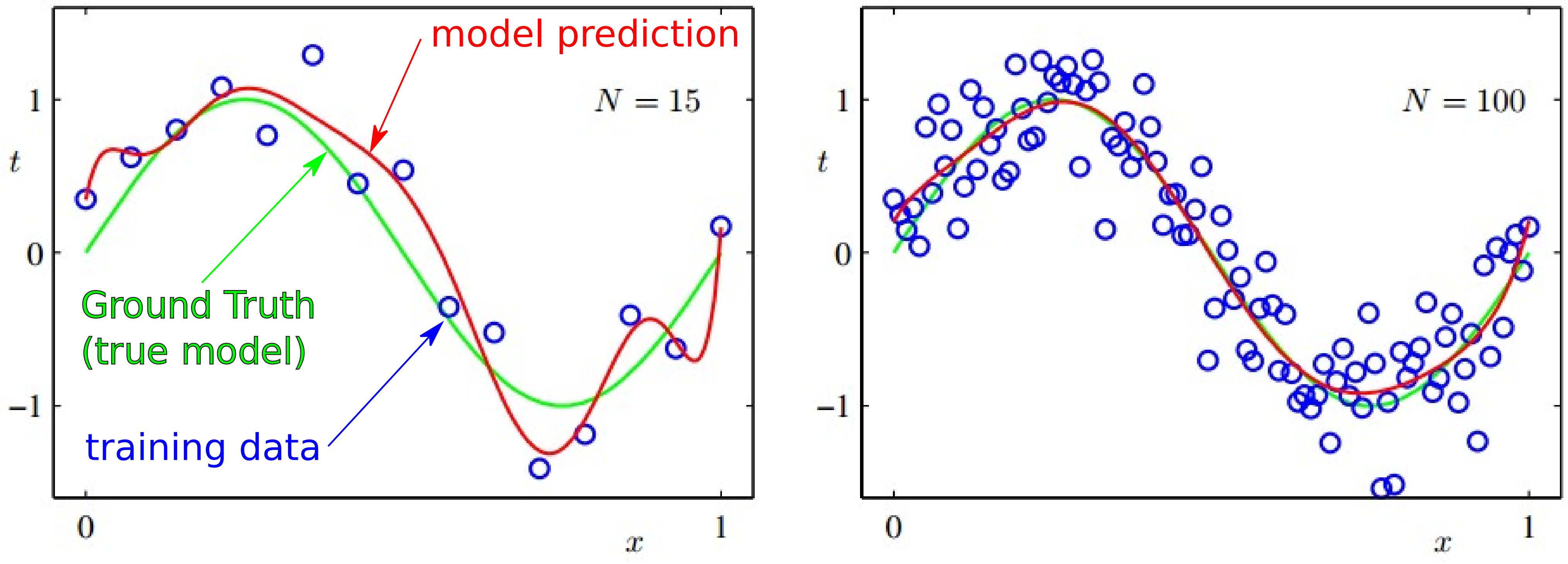
Overfit vs Train set size

- For instance, set: $N_{test} = 0.2N, N_{train} = 0.8N$

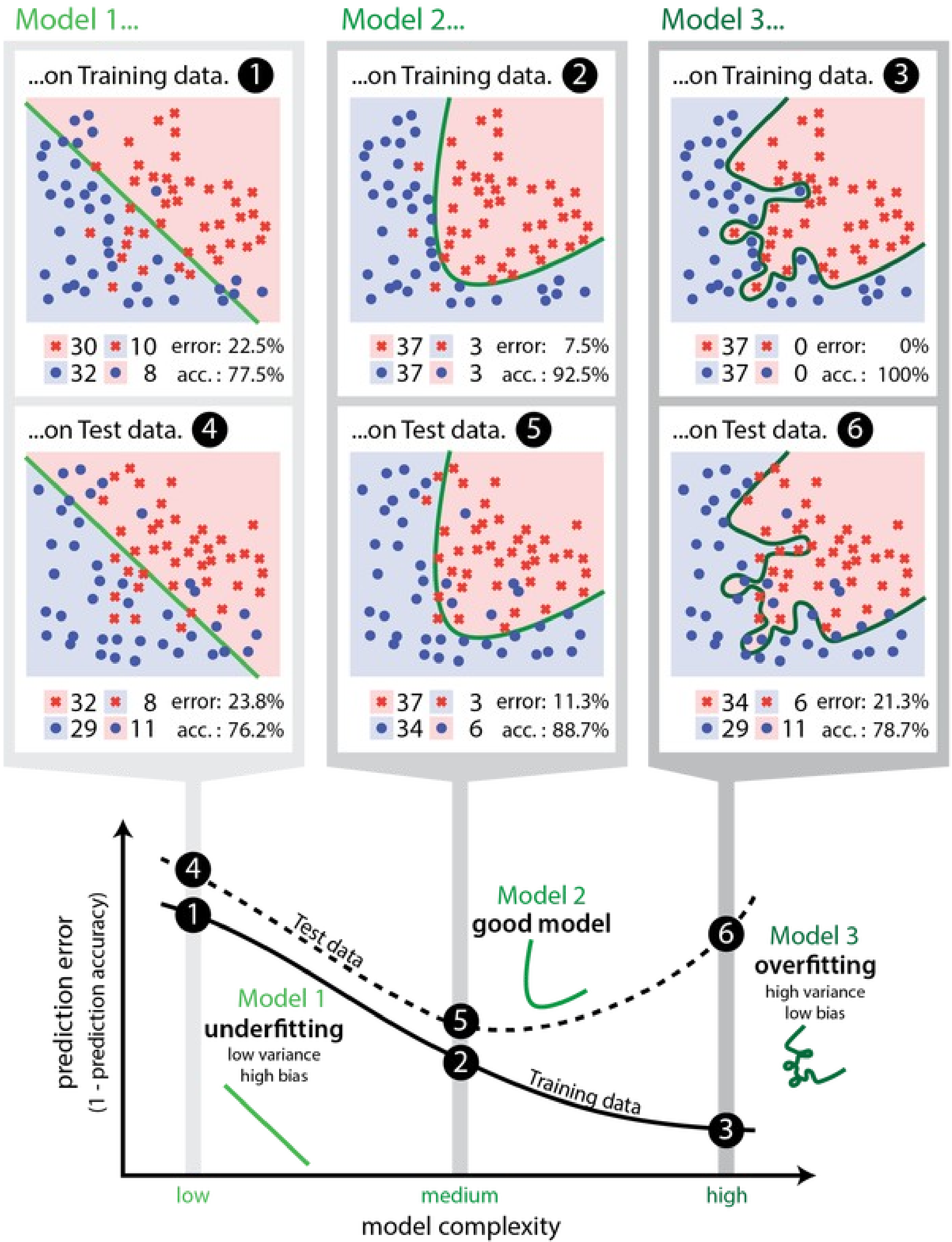


Overfit vs Train set size

- Restrain overfitting by adding more training data (here, using a 9th order polynomial)



Overfit vs model *complexity*



Model *complexity*
= model *capacity*

Concept of **hyper-parameter**

Not really a choice: Train set size $Ntrain$ (quite fundamental)
→ use as much as available, + study the *learning curve*

Overfitting depends on many choices: <u>the **hyper-parameters *μ***</u>

- ***Learning rate*** η
- **Initialization** of parameters $\mathbb{P}(\theta_0)$
- Learning strategy (size m of the mini-batch for instance)
- **Stopping criterion** (iterative methods: *MaxIter* or *tolerance*)
- **Pre-processing** choices (standardization or not, etc)
- Model ***capacity*** (or ***complexity***): not well defined. It's a bit of everything. Concretely: number of parameters (Cardinal of Θ), architecture, Kernel,…

→ Basically, everything that is not a parameter (a $\theta \in \Theta$) … is a hyper-parameter !

→ Let's optimize also these hyper-parameters *μ* !

Validation Set

If we also optimize the hyper-params,
then we can also over-fit them !?!

Train set

Train parameters $\theta \rightarrow \theta^*$

(Hyper-param. fixed) $\mu = \mu_0$

Validation set

→ Optimize hyper-param. $\mu \rightarrow \mu^*$

(Parameters fixed) $\theta = \theta^*$

Test set (single use!)

(Hyper-param. fixed) $\mu = \mu^*$

(Parameters fixed) $\theta = \theta^*$

→ measure performances

Over-fitting
General things

How to improve performance ?

- Seeing a lot of overfitting ?
→ **reduce the model complexity**
(try **simpler** models)

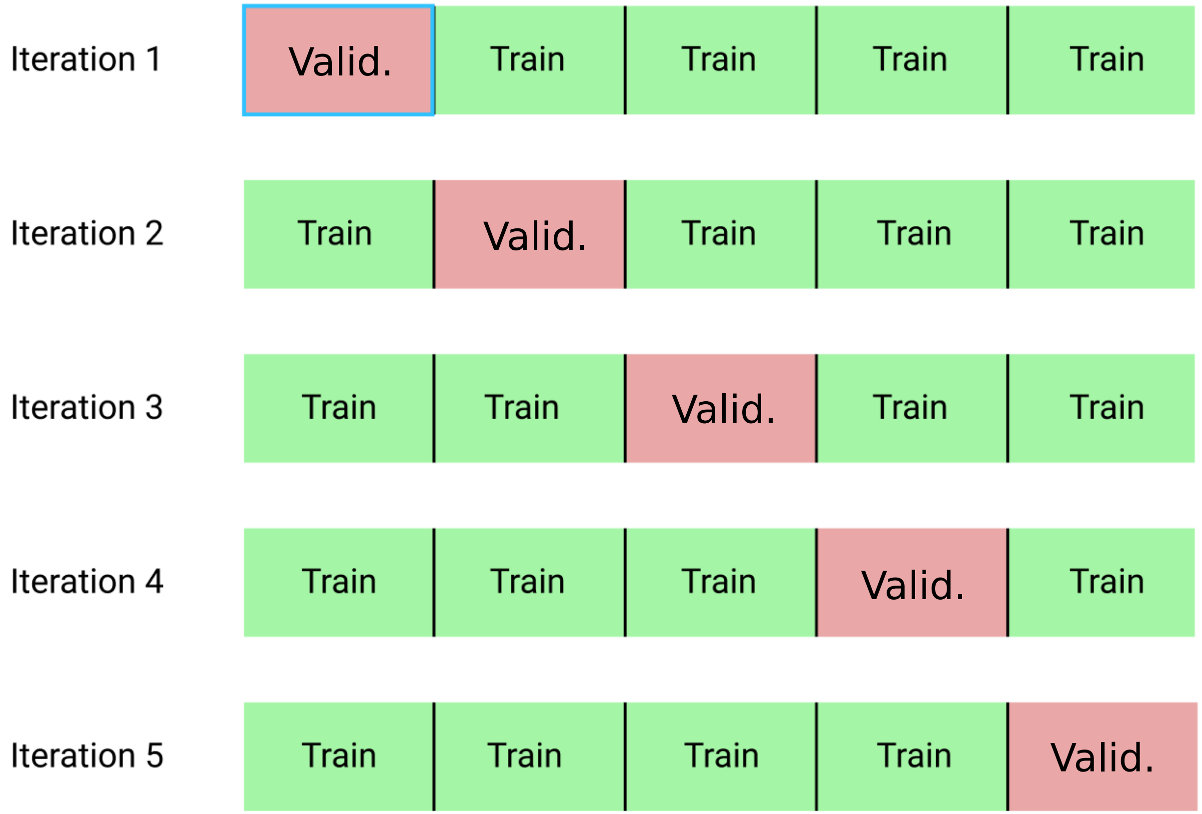
- Little to no overfit ?
→ try more ***expressive,*** more ***flexible*** models

Searching the ***global minimum*** $J(\theta, X_{train})$ .. or not ?
→ "best fit" possible but… on the *train set* !
→ in general, global min. = large over-fit.

- Ill-defined problem: what is *generalizability* ?
→ How to sample "the set of all 2D images showing a dog" ? → *Generative Models*. Quality ??
→ *Transfer Learning*

a **Cross-Validation**
K-fold CV

- Make $K\ folds$, e.g. $K{=}5$ train/validation splits



→ reduces the splitting-related noise

Another Cross-Validation
Leave-one-out CV (**LOO**)

Def: Like K-fold but with $\mathrm{K{=}Ntrain}$.

- **Useful** esp. for small data sets
  (reduce Ntrain by only 1 example)
- **Reasonable** only for small data sets
  (otherwise, too many computations)

Key concepts

- Generalization, ***over-fitting***, *under-fitting*, performance
- **The split : *Train, validation, test***
- **Amount of overfitting ≠ performance**
- Train set size
- **Hyper-parameters**
- Complexity ~ capacity ~ expressiveness
- Cross-Validation
- Curse of dimensionality

To go further: keywords

Basic: *Hypothesis space*, finite vs infinite.

A rather classic, finite-dim, finite set approach:

- Vapnik–Chervonenkis dimension (VC dimension)
- Probably approximately correct learning (PAC learning)

- Another kind of approach:
  There are exact results for *random data sets* (some are physcists' or mathematicians works).
  More keywords: tensor PCA, planted solution, random constraint satisfaction problems (CSP), dynamic threshold (algorithmic threshold), Information Theoretic threshold (IT),