Machine learning, a bird's eye view

Kipton Barros
Los Alamos National Lab.

2nd ML in Solid Earth Geoscience
March 18, 2019

Main Collaborators:

Claudia Hulbert

Bertrand Rouet-Leduc

Nick Lubbers

LA-UR-19-22370

Forms of machine learning

Traditional Stats.

Confidence intervals, hypothesis testing, probabilistic models…

Data mining (1990s?)

Google web search (1998)

Netflix Prize (2006)

10% improvement 2009

Statistical/computational learning theory

Theorems from statistics and functional analysis

Artificial intelligence (1950s--?)

Artificial neurons, McCullouch & Pitts (1943)

Turing test (1950)

Chinook checkers (champion in 1994)
(solved in 2007)

Deep Blue chess (1996)

AlphaGo (2016)

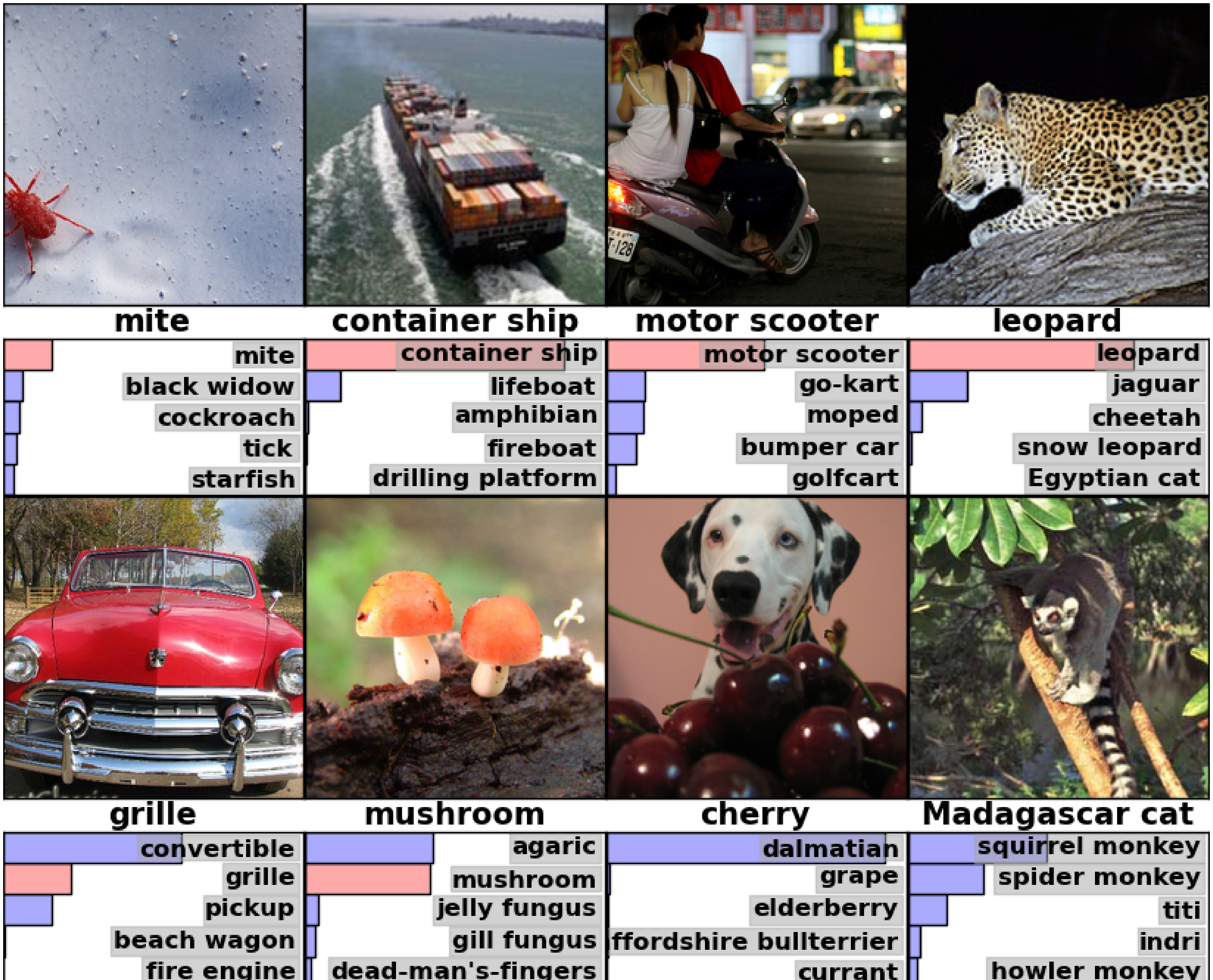
"AI winter" … AI spring?

Machine learning

"*Algorithms that find structure in big datasets, using empirical models and regularizations.*" (?)

Breiman, Leo. Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author). Statist. Sci. 16 (2001), no. 3, 199--231. doi:10.1214/ss/1009213726. http://projecteuclid.org/euclid.ss/1009213726.

Image labeling



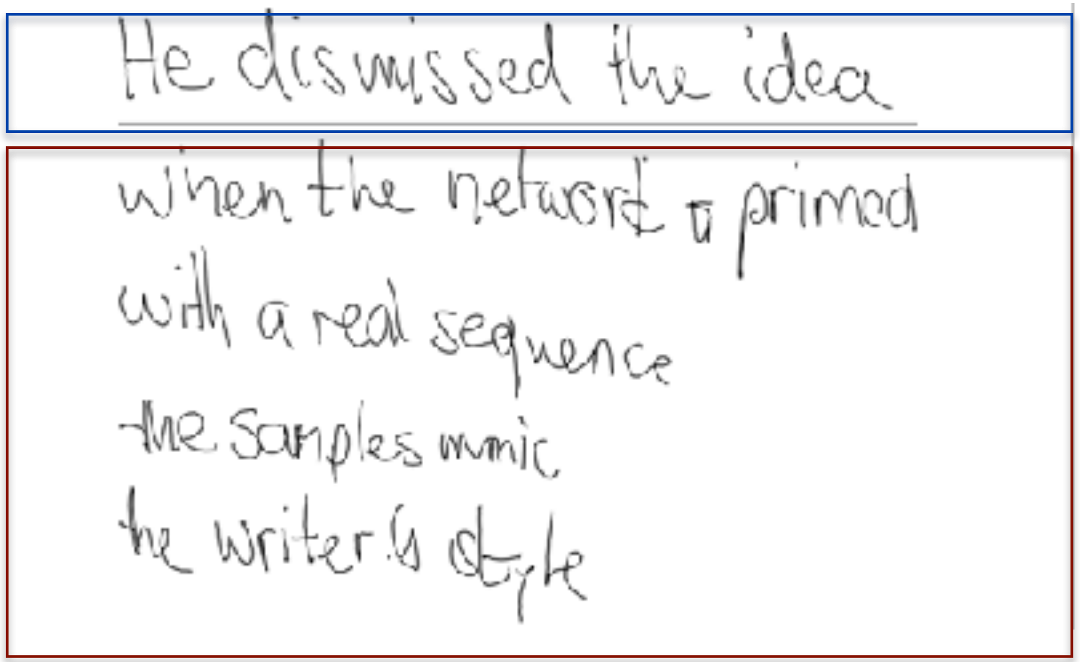
Transfer learning

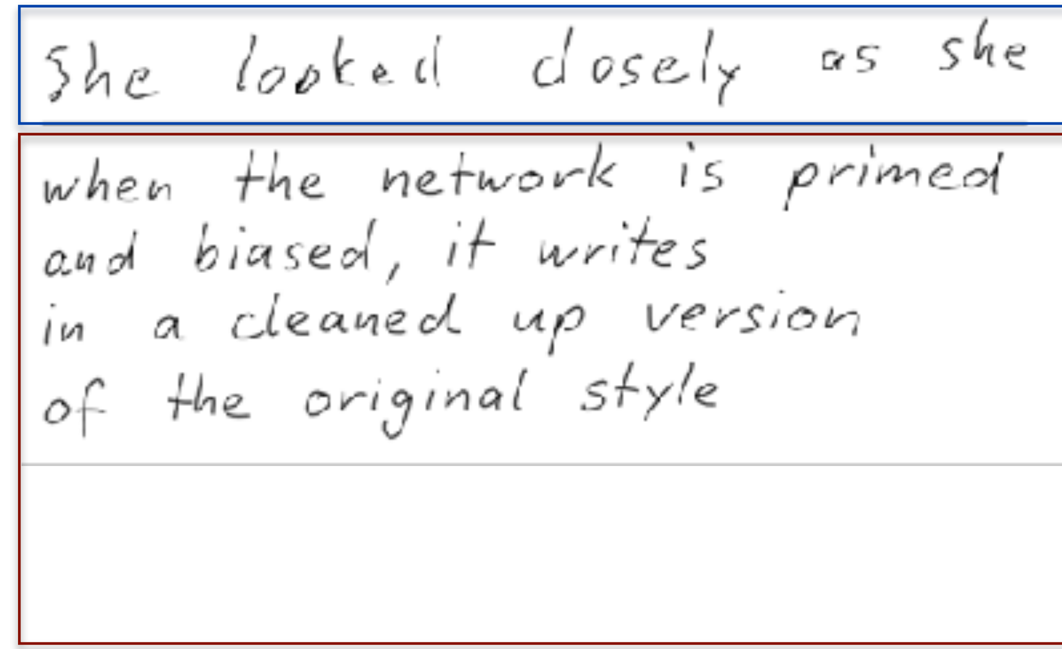
Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style." arXiv preprint arXiv:1508.06576 (2015).

Handwriting synthesis

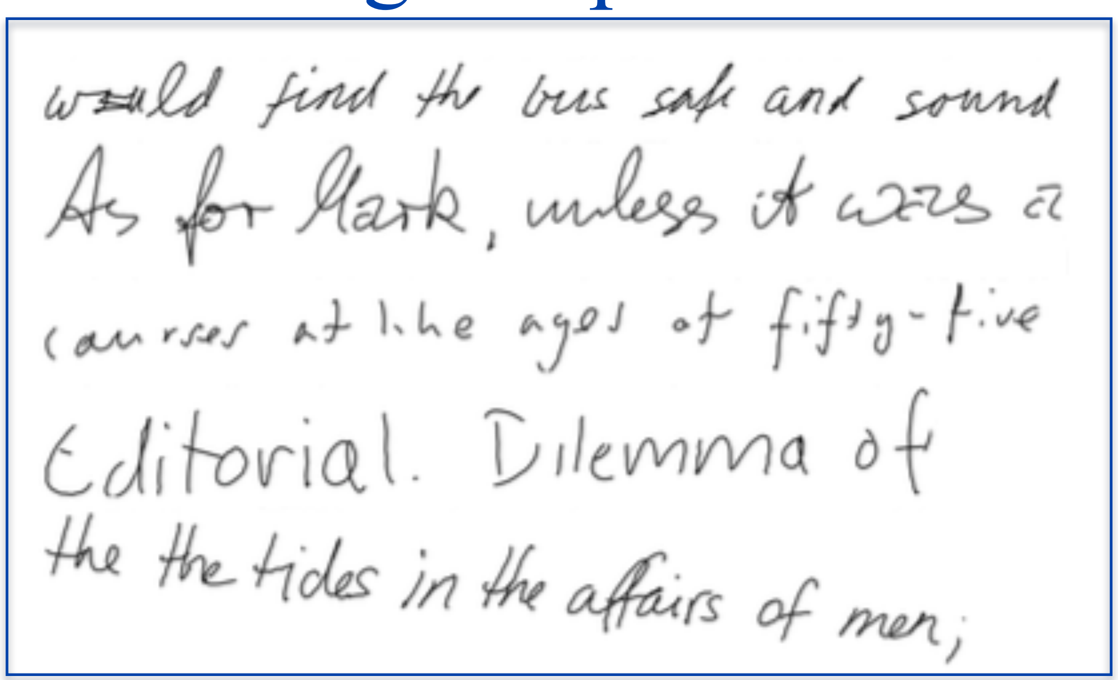
Human input

Generated output

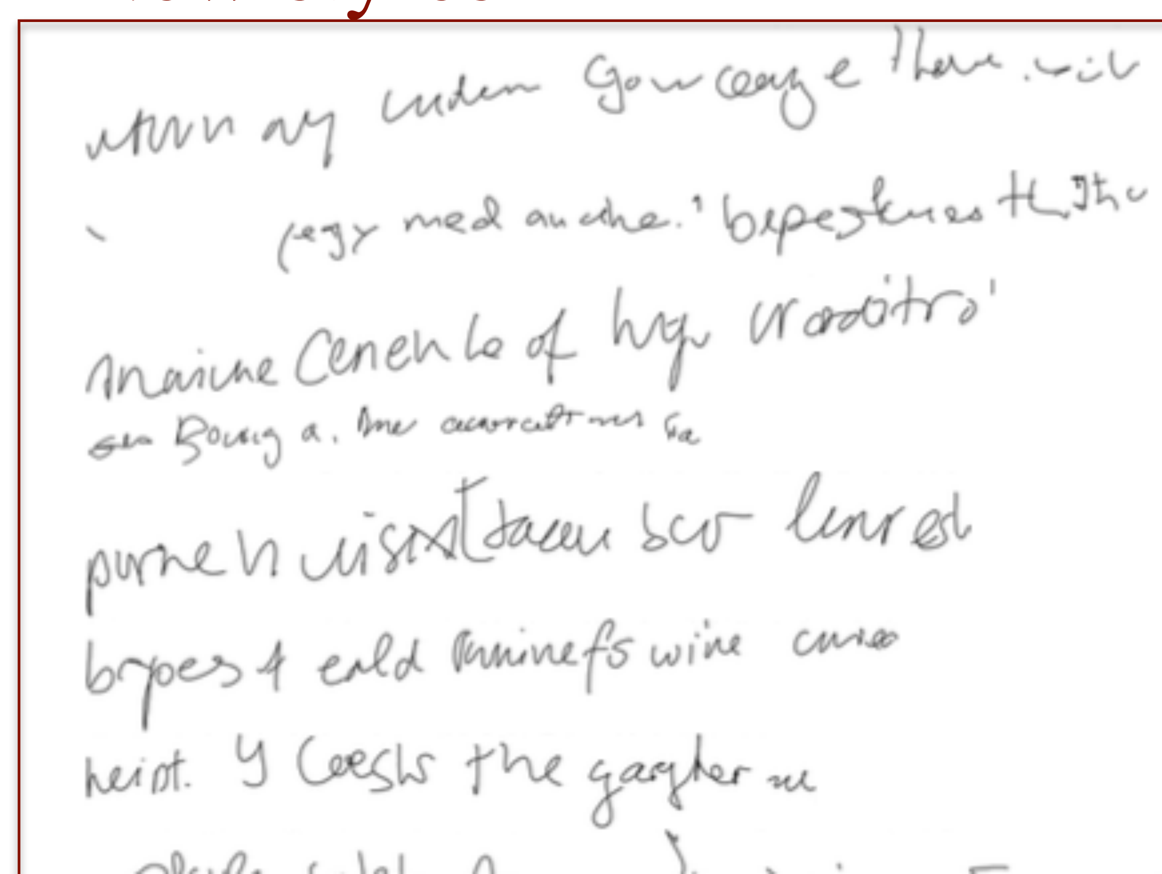




Training samples



New styles



A. Graves. Generating sequences with recurrent neural networks. CoRR, abs/1308.0850, 2013.

Professional level Go play

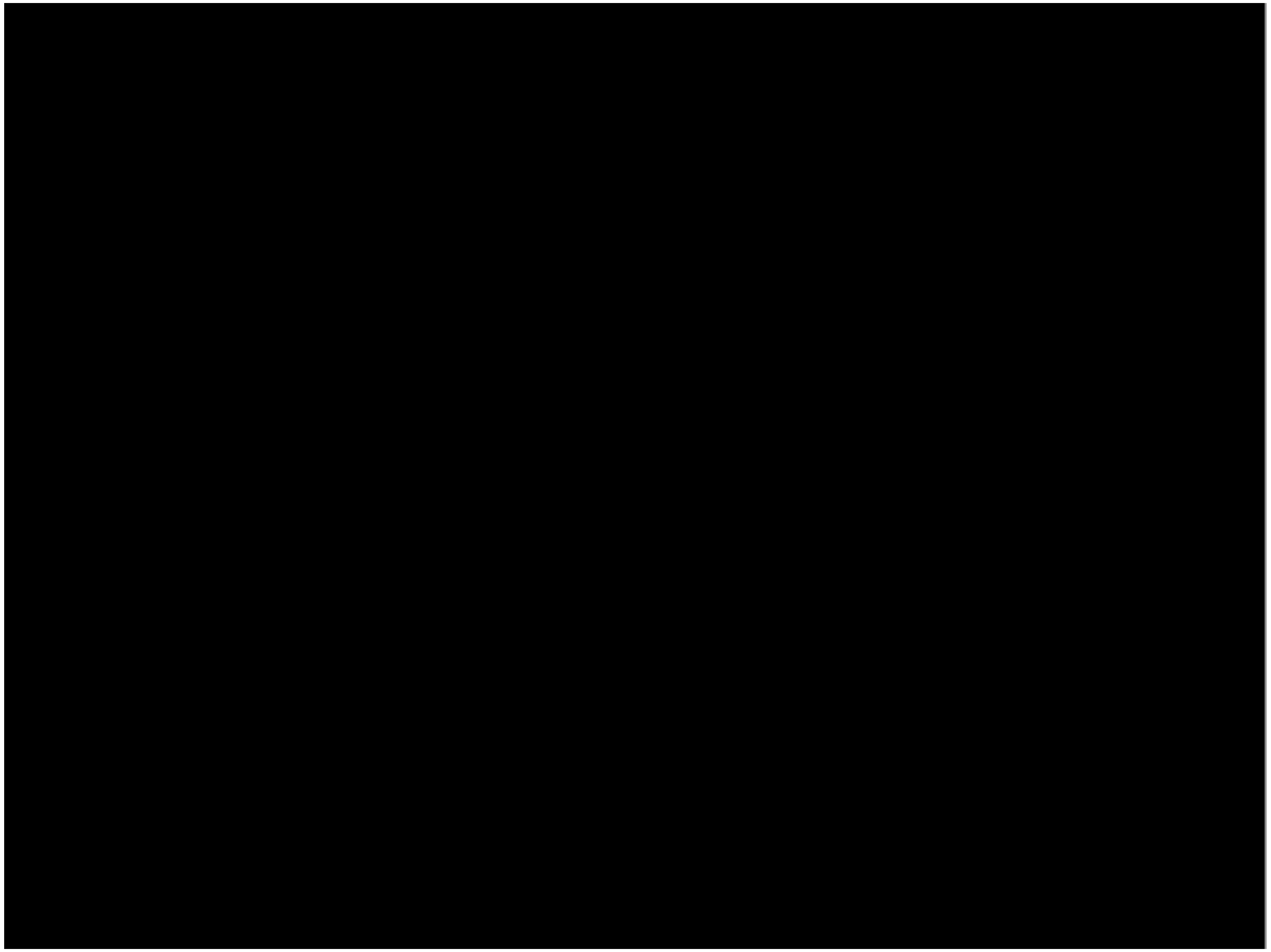
Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." Nature 529.7587 (2016): 484-489.

~~Trained using a large database of professional games~~

Subsequent learning through self-play.

AlphaZero, 2017

Learning to play video games



Google’s DeepMind AI beats 49 Atari games, exceeds human performance

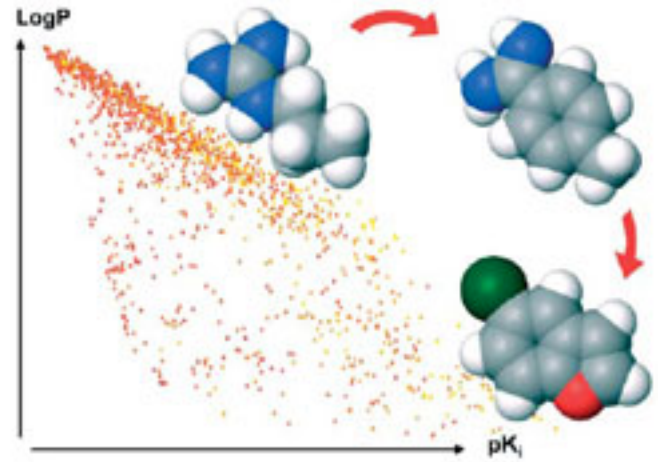
Training technique: Positive reinforcement learning

Mnih et al, “Human-level control through deep reinforcement learning”, Nature **518** 529--533 (2015)

ML for physics, interesting ideas

Chemo-informatics

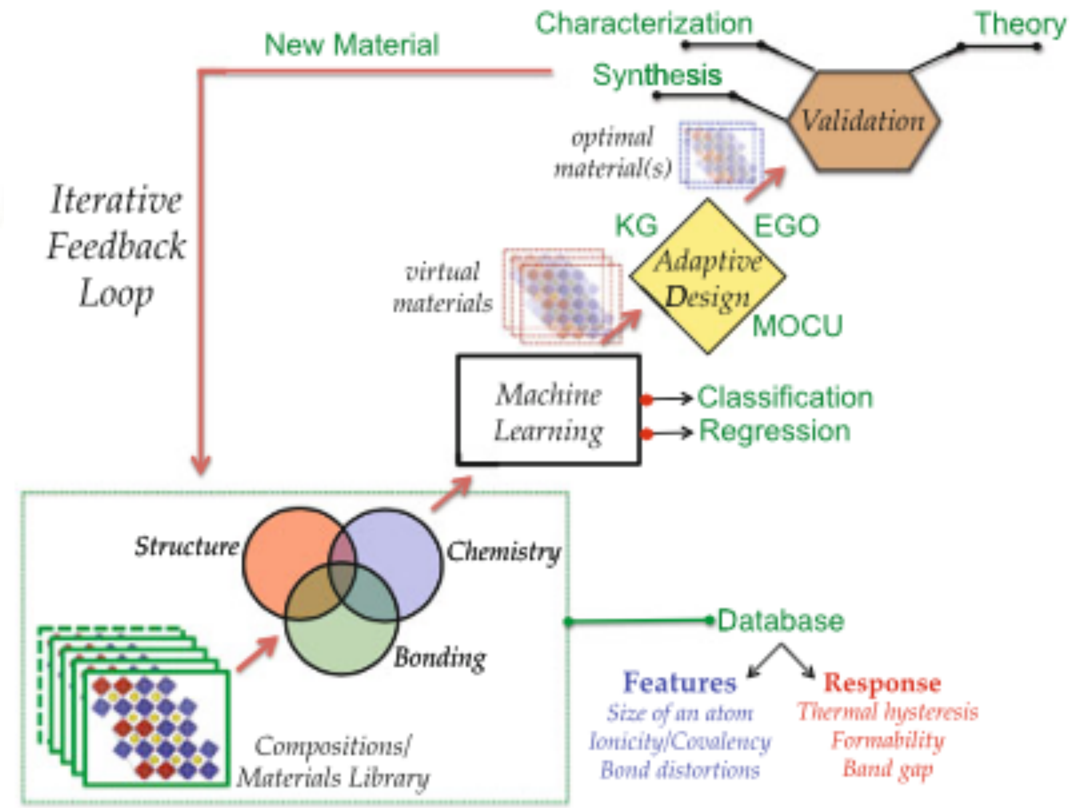
Links between chemical structures and activity, molecular finger-printing



Materials informatics

Design of new functional materials

Lookman et al, "A perspective on Materials Informatics: State-of-the-art and Challenges" (2016)



Statistical physics

MD / DEM potentials

Coarse grained molecular dynamics

Effective models for fluids

Microstructure / phase field modeling

Geophysics

Earthquake early warning

Seismic inversion

Flow in fractured media

…

Types of Machine Learning

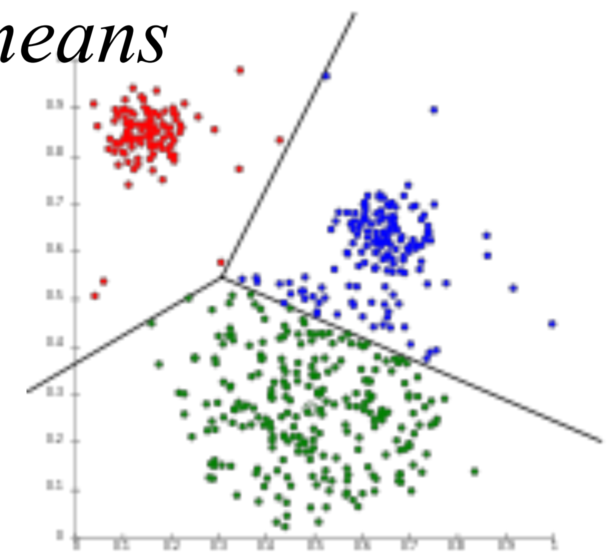
- *Unsupervised Learning*: Learn structure of unlabeled data.
- *Supervised Learning*: Learn the map between inputs and outputs.
- *Reinforcement Learning*: Learn to perform tasks using a reward scheme.

...

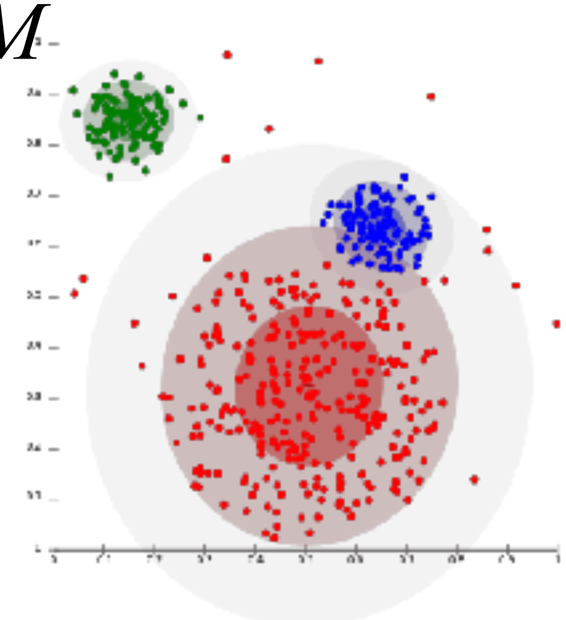
Unsupervised learning

Clustering

k-means

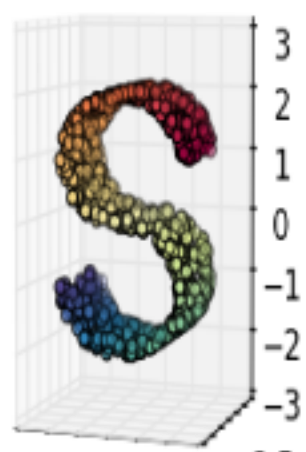


EM

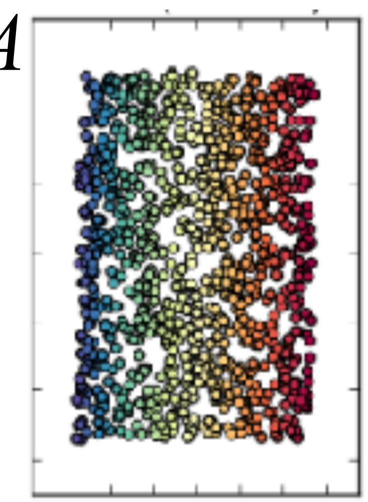


...

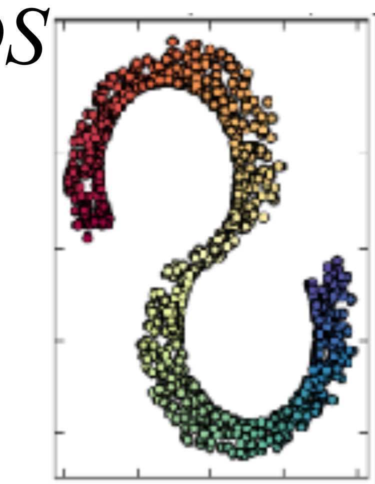
Manifold learning



LTSA

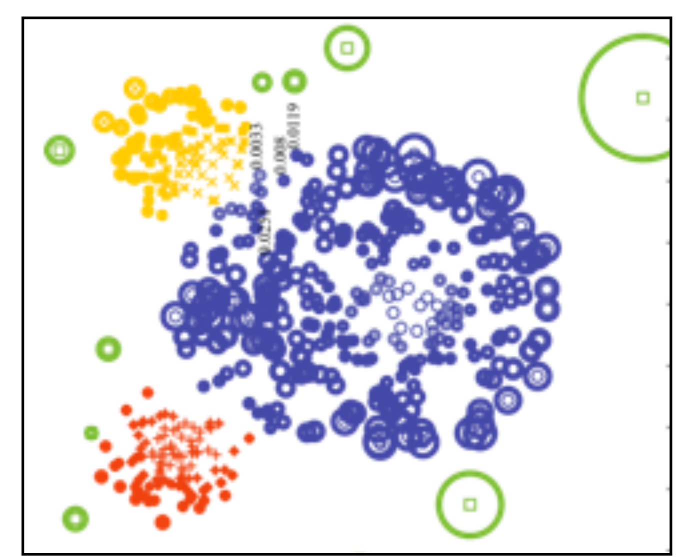


MDS



...

Anomaly detection



Supervised learning

Labeled dataset

$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots\}$

Goal: Learn map

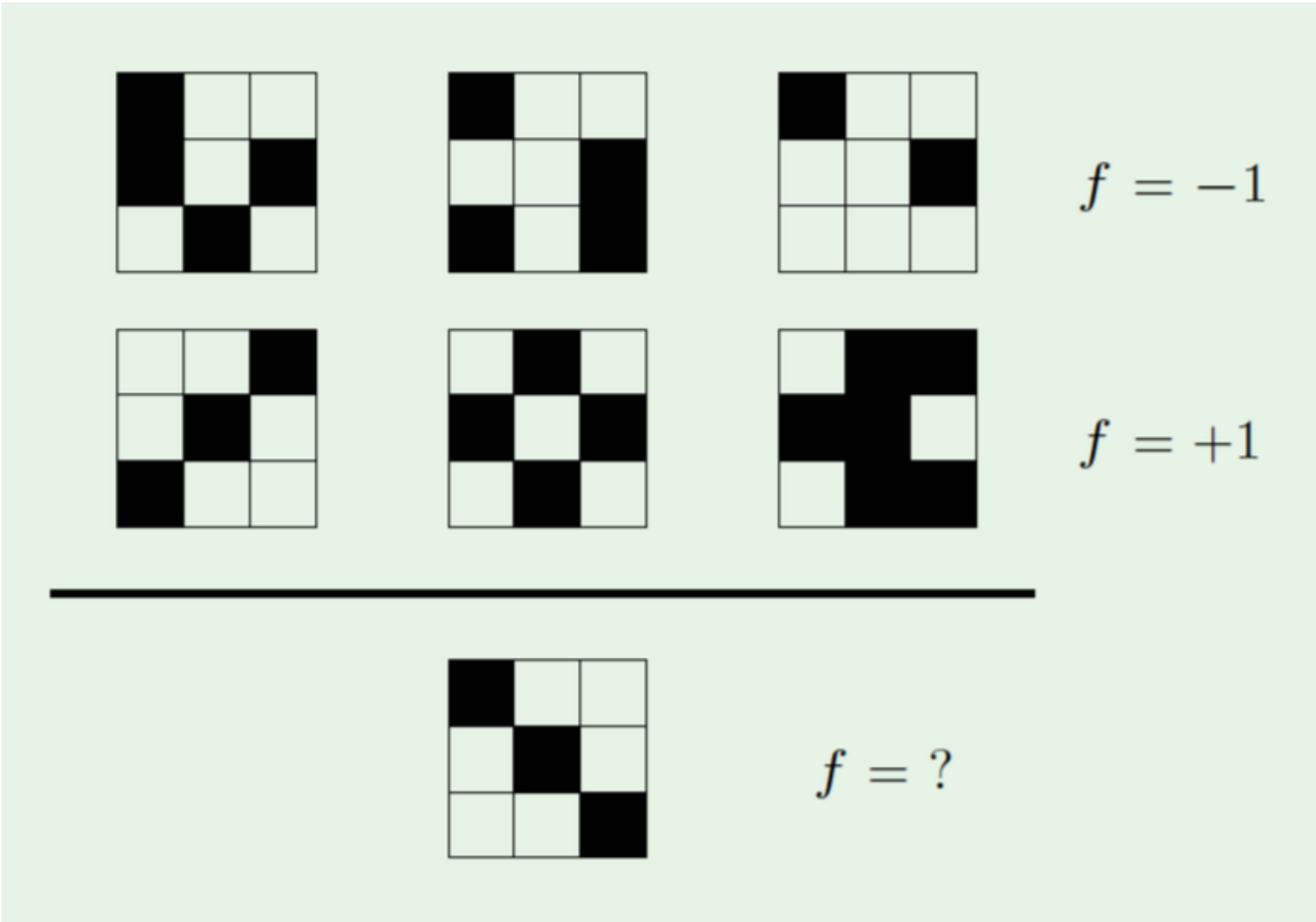
$\boldsymbol{x} \longrightarrow y = f(\boldsymbol{x})$

60,000 handwritten digits (MNIST data)

Labels

0
1
2
3
4
5
6
7
8
9

Puzzle -- no free lunch



Y. S. Abu-Mostafa's online class "Learning from Data" (edX CS1156x).

Step by step: Linear regression

Source:

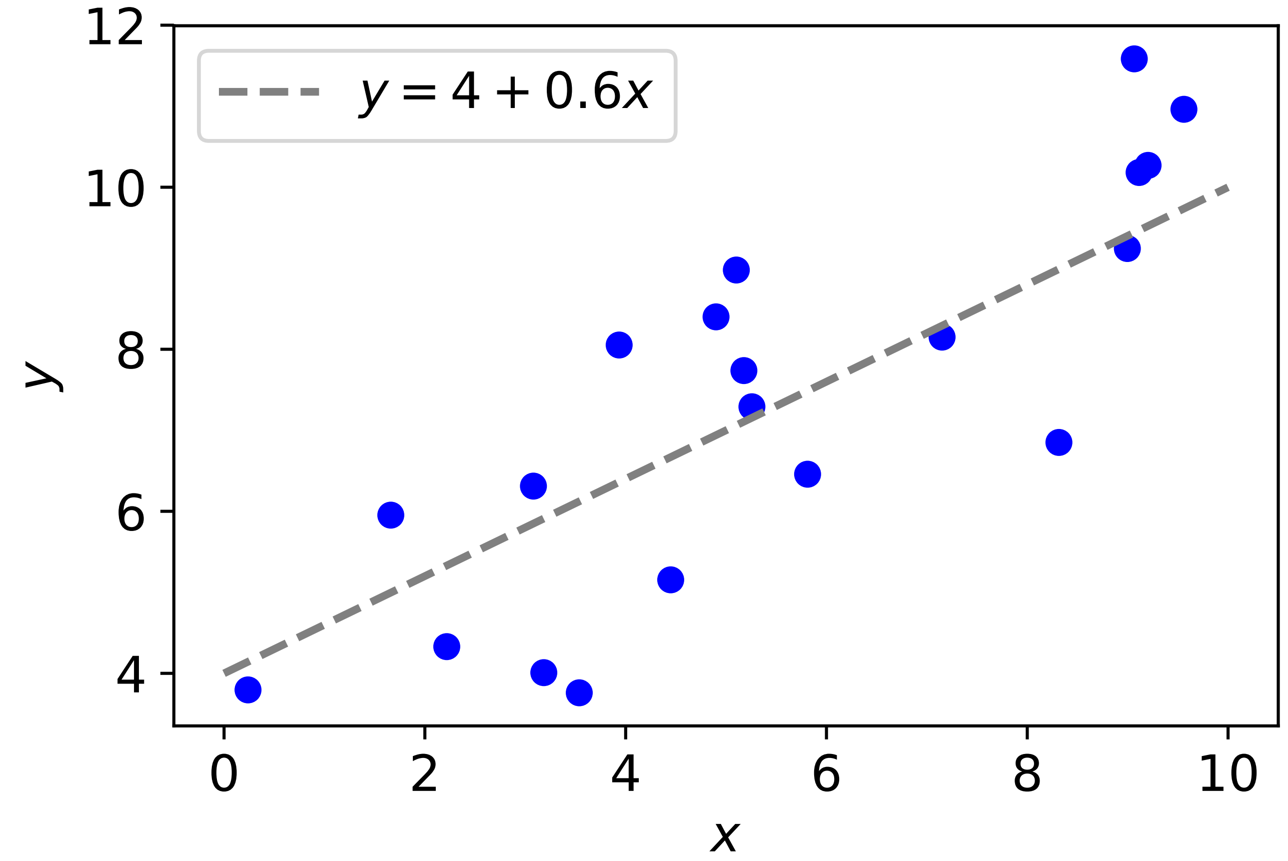
$$y_i = 4 + 0.6x_i + \epsilon_i$$

Noise term:

$$\epsilon_i \sim \mathcal{N}(\mu = 0, \sigma = 1)$$

Goal: Build model

$$\hat{y}(x) = ???$$



Step by step: Linear regression

Source:

$$y_i = 4 + 0.6x_i + \epsilon_i$$

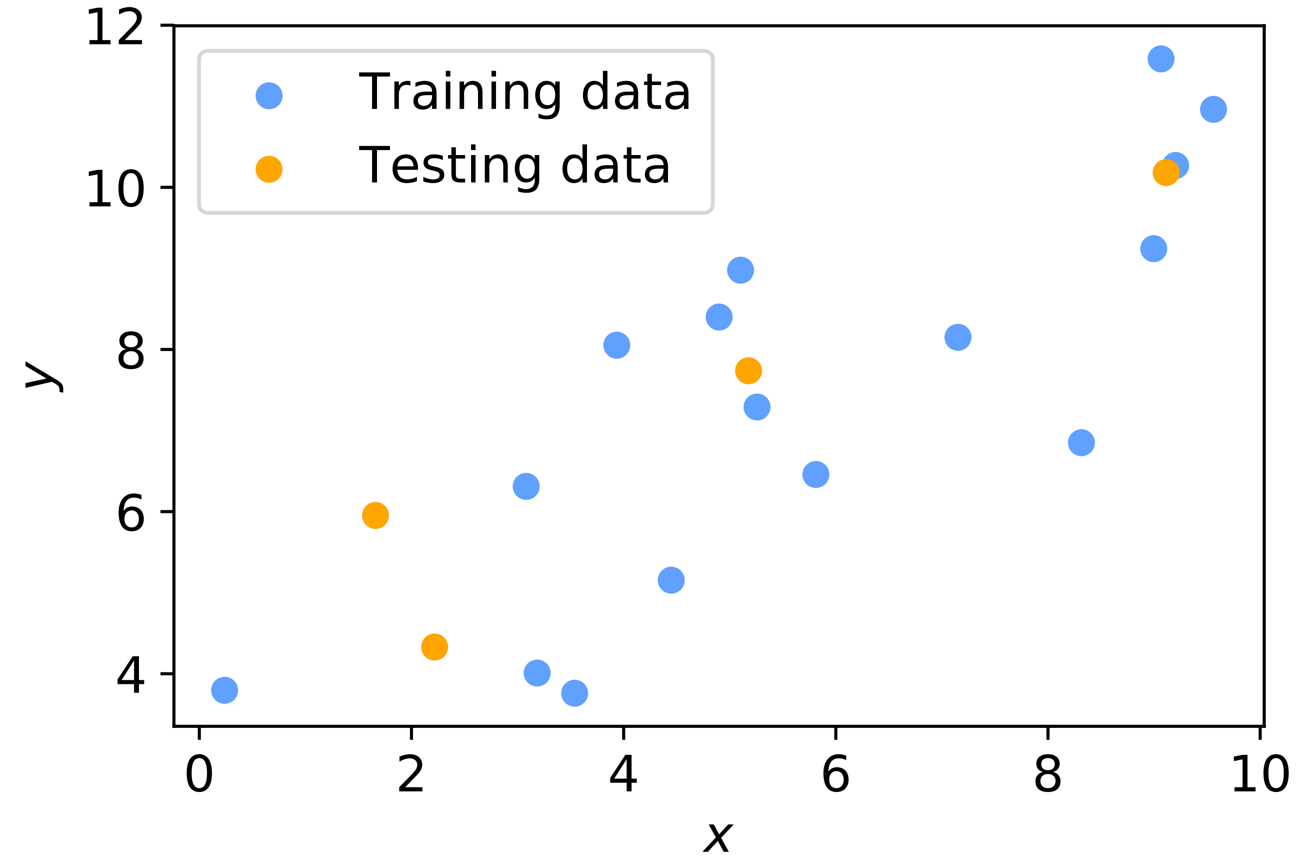
Noise term:

$$\epsilon_i \sim \mathcal{N}(\mu = 0, \sigma = 1)$$

Goal: Build model

$$\hat{y}(x) = ???$$

Step 1: Split data (80/20)



Step by step: Linear regression

Source:

$$y_i = 4 + 0.6x_i + \epsilon_i$$

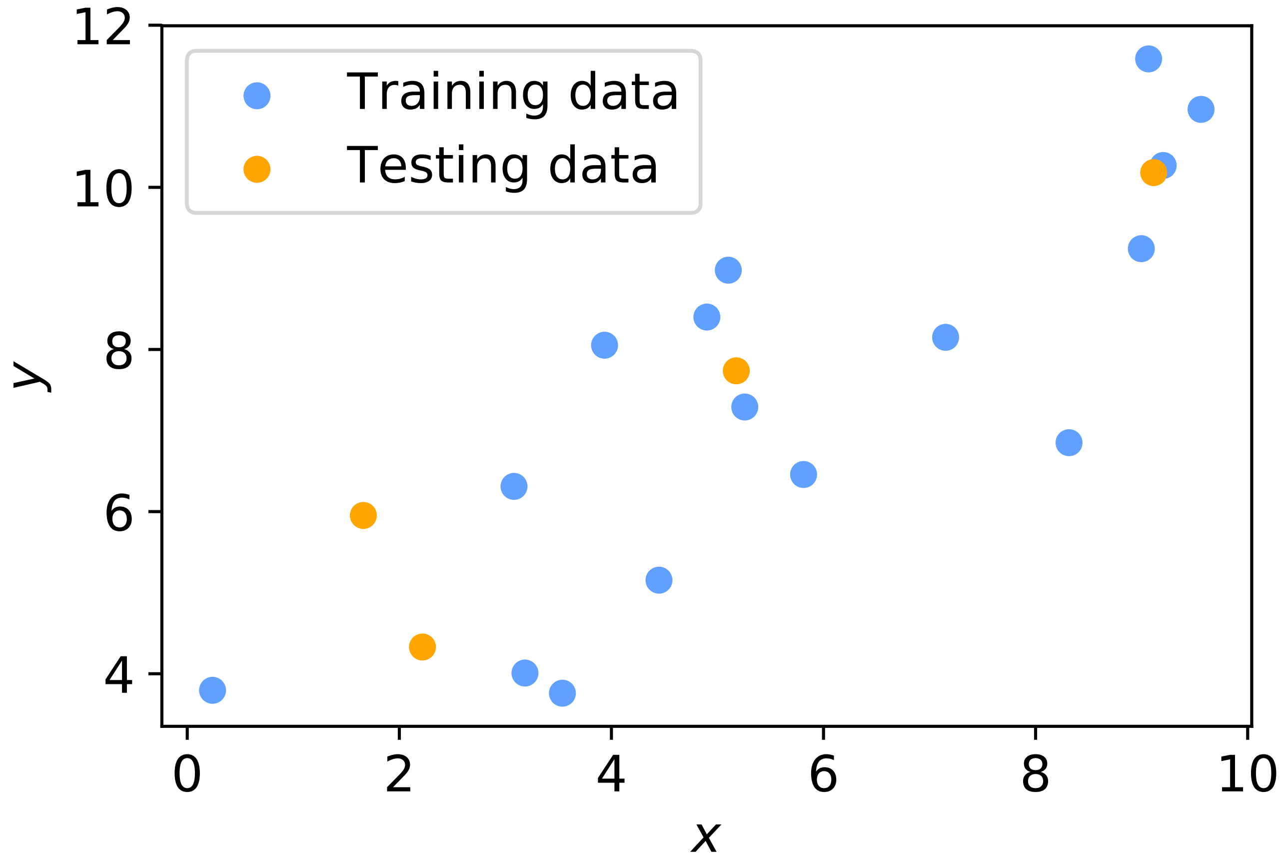
Noise term:

$$\epsilon_i \sim \mathcal{N}(\mu = 0, \sigma = 1)$$

Goal: Build model

$$\hat{y}(x) = ???$$

<u>Step 1</u>: Split data (80/20)



<u>Step 2</u>: Define **model**

$$\hat{y}(x) = \hat{\beta}_0 + \hat{\beta}_1 x$$

and **cost function** to optimize

$$\mathscr{L} = \sum_i (y_i - \hat{y}_i)^2$$

Step by step: Linear regression

Source:

$$y_i = 4 + 0.6x_i + \epsilon_i$$

Noise term:

$$\epsilon_i \sim \mathcal{N}(\mu = 0, \sigma = 1)$$

Goal: Build model

$$\hat{y}(x) = ???$$

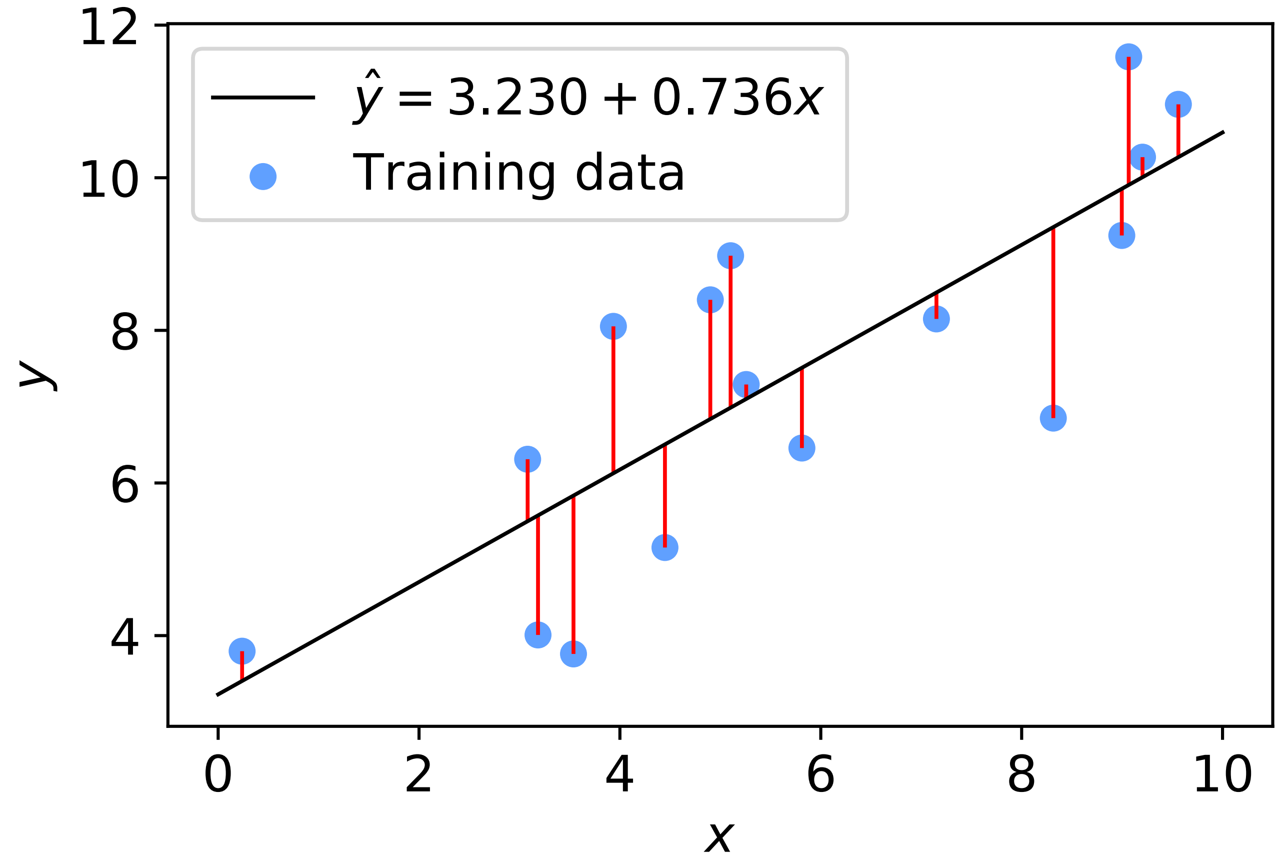
Step 1: Split data (80/20)

Step 2: Define model

$$\hat{y}(x) = \hat{\beta}_0 + \hat{\beta}_1 x$$

and cost function to optimize

$$\mathcal{L} = \sum_i (y_i - \hat{y}_i)^2$$



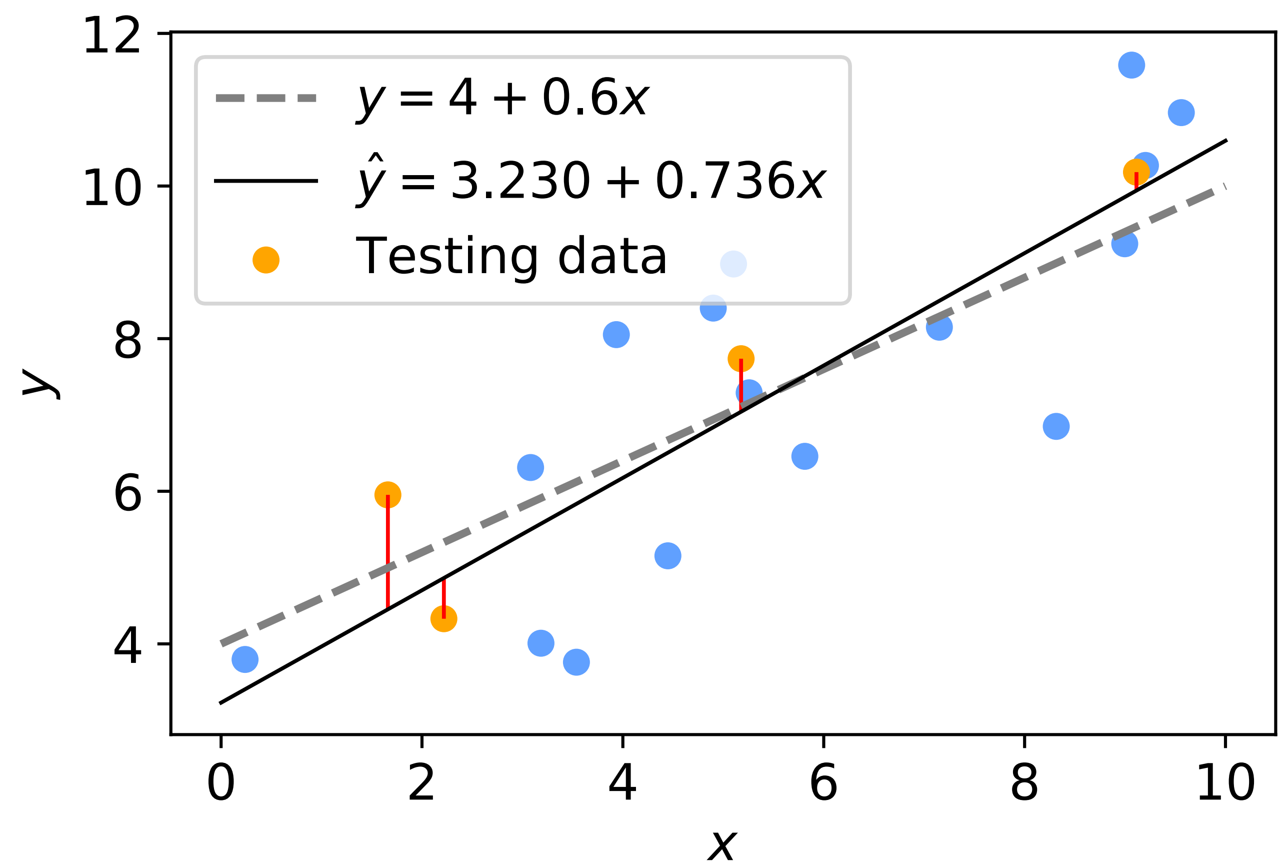
Step 3: Optimize on training data

$$\hat{\beta} = (X^T X)^{-1} X^T \mathbf{y}$$

X_{ij} is the j^{th} feature of point x_i

Step by step: Linear regression

Step 1: Split data (80/20)



Step 2: Define model

$$\hat{y}(x) = \hat{\beta}_0 + \hat{\beta}_1 x$$

and cost function to optimize

$$\mathscr{L} = \sum_i (y_i - \hat{y}_i)^2$$

Step 3: Optimize on training data $\hat{\beta} = (X^T X)^{-1} X^T \mathbf{y}$

Step 4: Measure performance on test data $R^2_{\text{train}} = 0.66$ $R^2_{\text{test}} = 0.84$

Performance metric

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)}{\sum_i (y_i - \bar{y}_i)} = \begin{cases} 1 \text{ if perfect} \\ 0 \text{ if unpredictive} \end{cases}$$

Linear regression, take 2

Source:

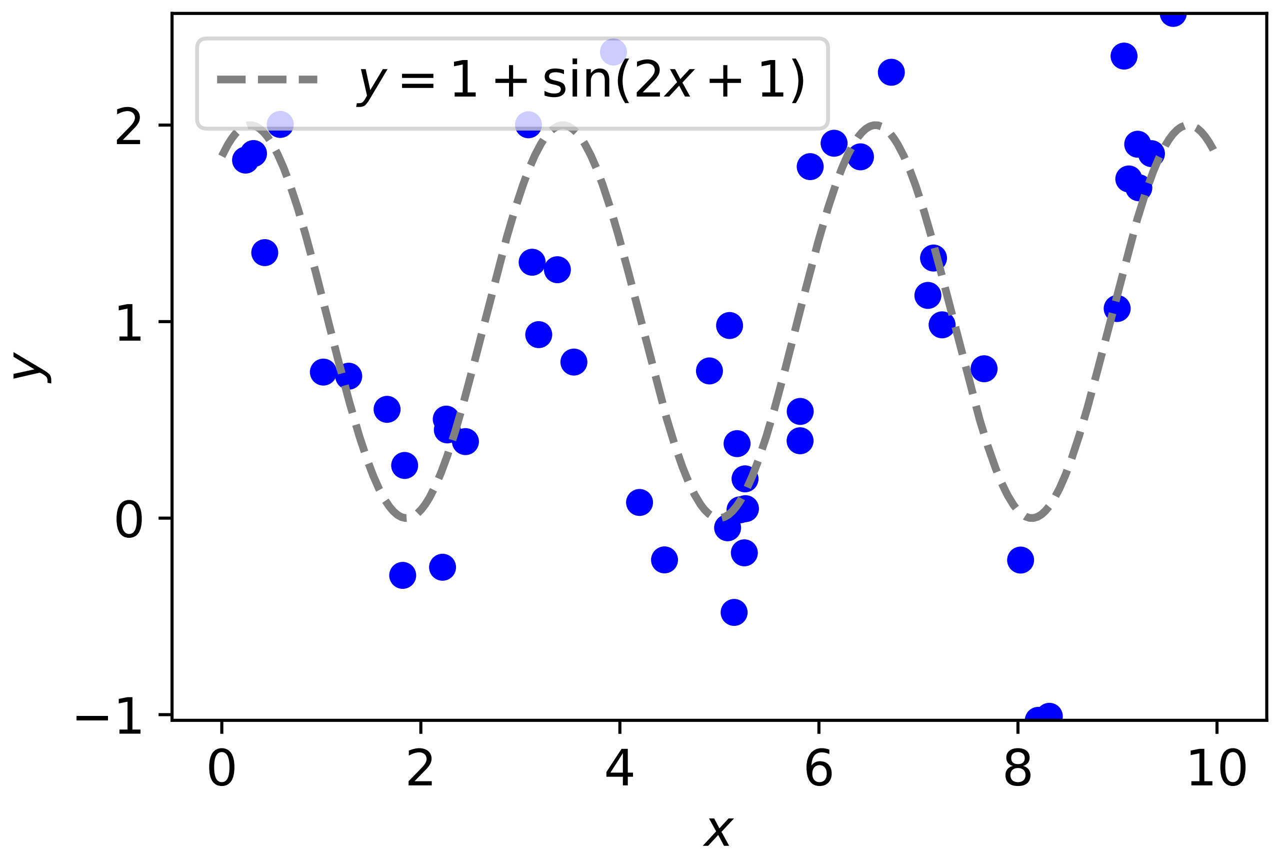
$$y_i = 1 + \sin(x_i + 1) + \epsilon_i/2$$

Noise term:

$$\epsilon_i \sim \mathcal{N}(\mu = 0, \sigma = 1)$$

Goal:

$$\hat{y}(x) = ???$$



Linear regression, take 2

Source:

$$y_i = 1 + \sin(x_i + 1) + \epsilon_i/2$$

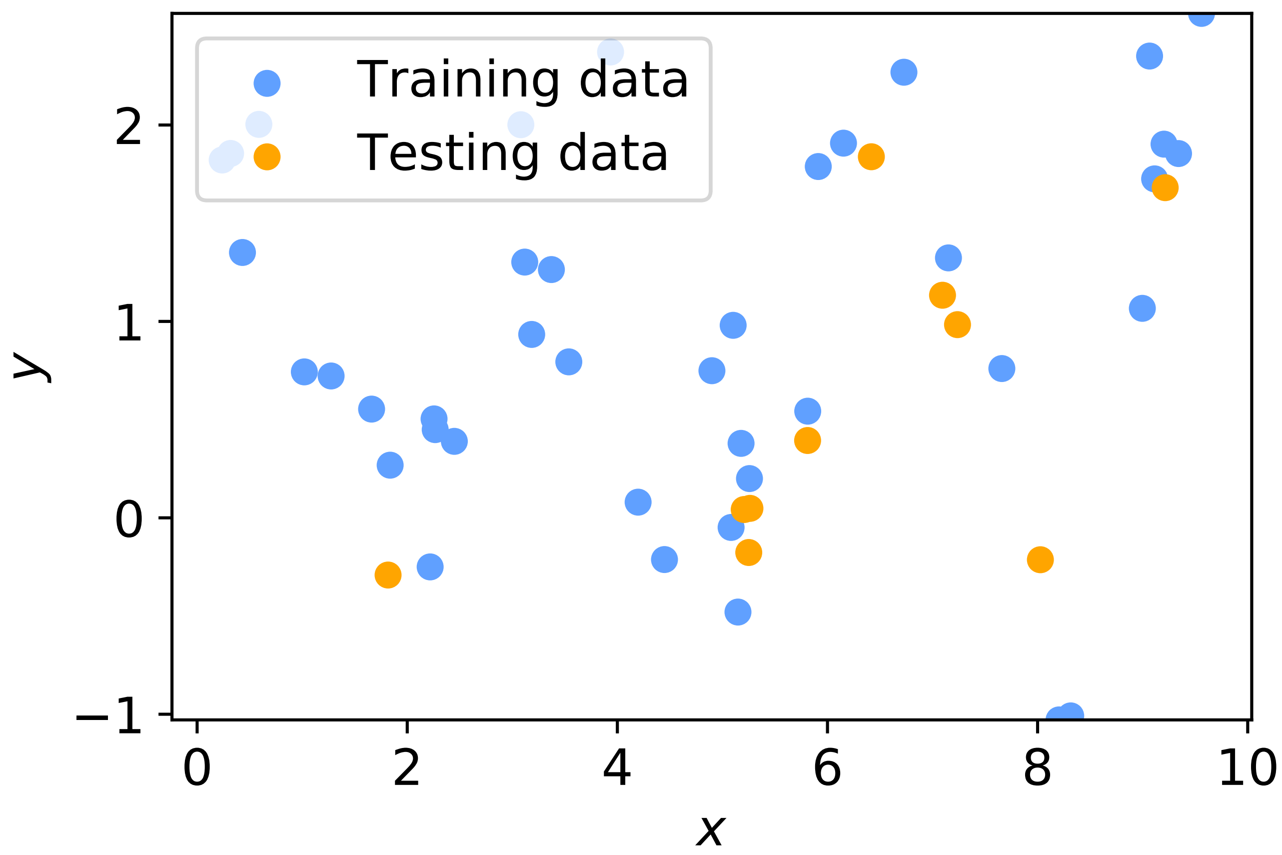
Noise term:

$$\epsilon_i \sim \mathcal{N}(\mu = 0, \sigma = 1)$$

Goal:

$$\hat{y}(x) = ???$$

Step 1: Split data (80/20)



Linear regression, take 2

Source:

$y_i = 1 + \sin(x_i + 1) + \epsilon_i/2$

Noise term:

$\epsilon_i \sim \mathcal{N}(\mu = 0, \sigma = 1)$

Goal:

$\hat{y}(x) = ???$

Step 1: Split data (80/20)

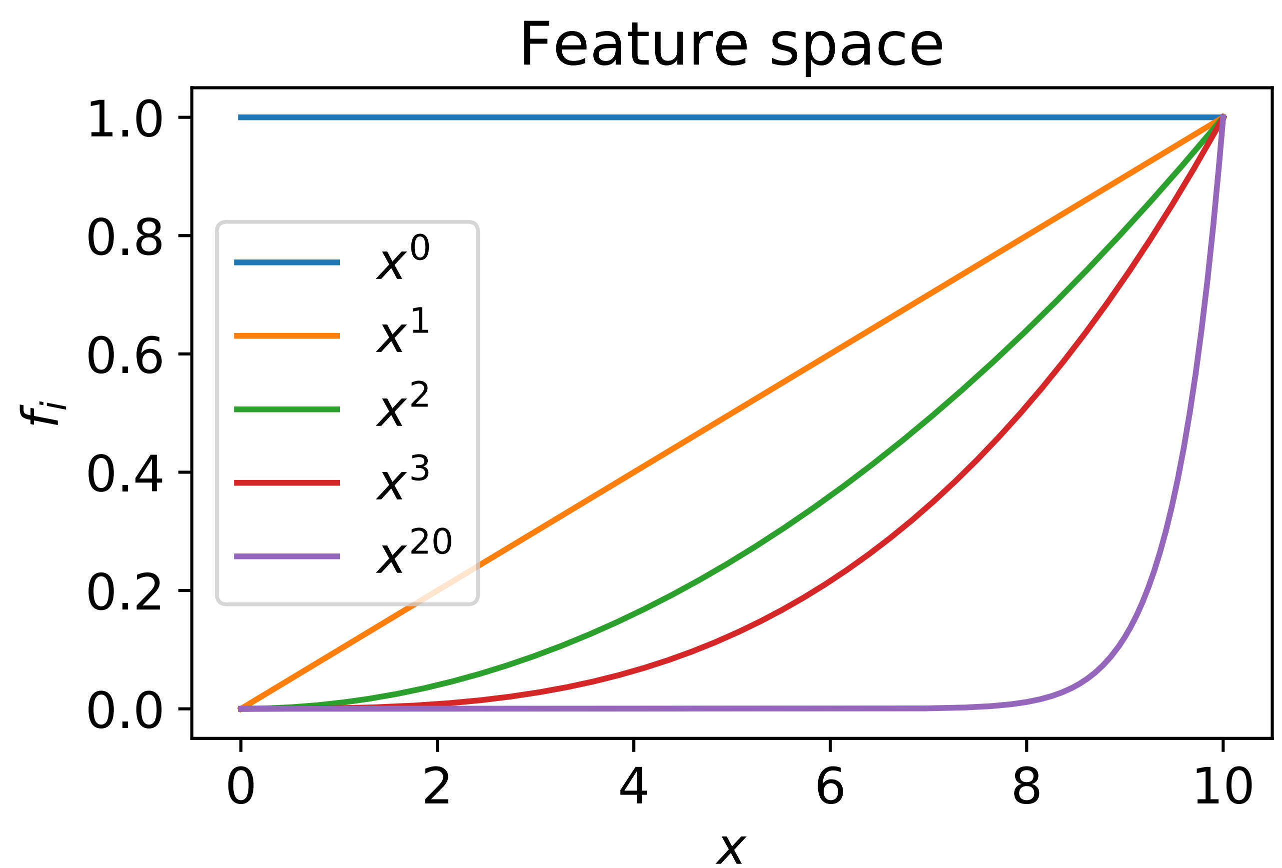
Step 2: Define **feature space**

$$f_i = x^i \in \{1, x, x^2, \ldots, x^N\},$$

for **linear** model

$$\hat{y}(x) = \sum_{i=0}^{N} \beta_i f_i(x),$$

and sum-of-squares cost function $\mathscr{L} = \sum_i (y_i - \hat{y}_i)^2$



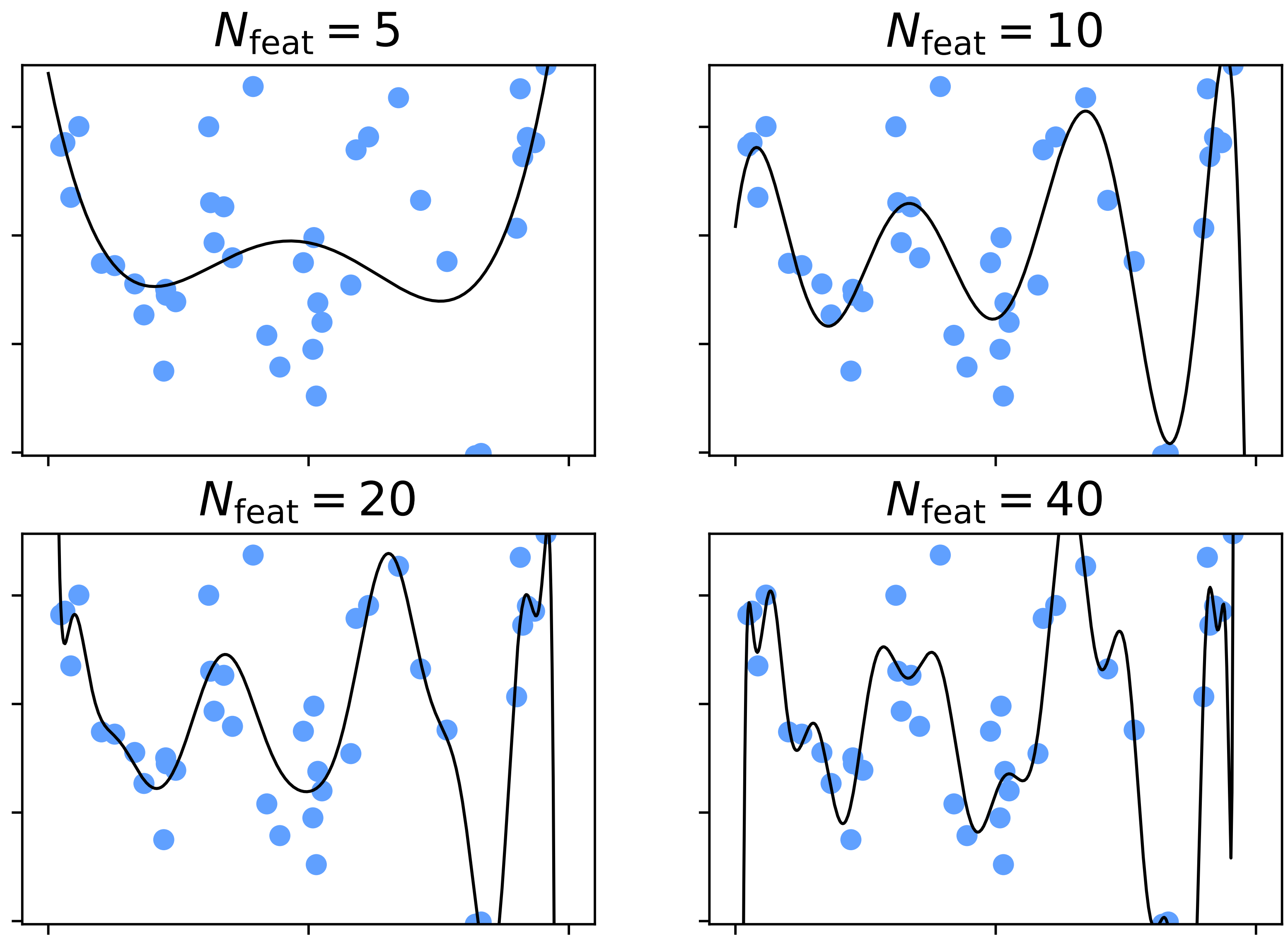
Linear regression, take 2

Step 1: Split data (80/20)

Step 2: Define model

$$\hat{y}(x) = \sum_{i=0}^{N} \beta_i f_i(x)$$

$$f_i = x^i$$



Step 3: Optimize on training data

$$\hat{\beta} = (X^T X)^{-1} X^T \mathbf{y}$$

$$X_{ij} = f_j(x_i)$$

… **how to evaluate** various cutoff N?

Linear regression, take 2

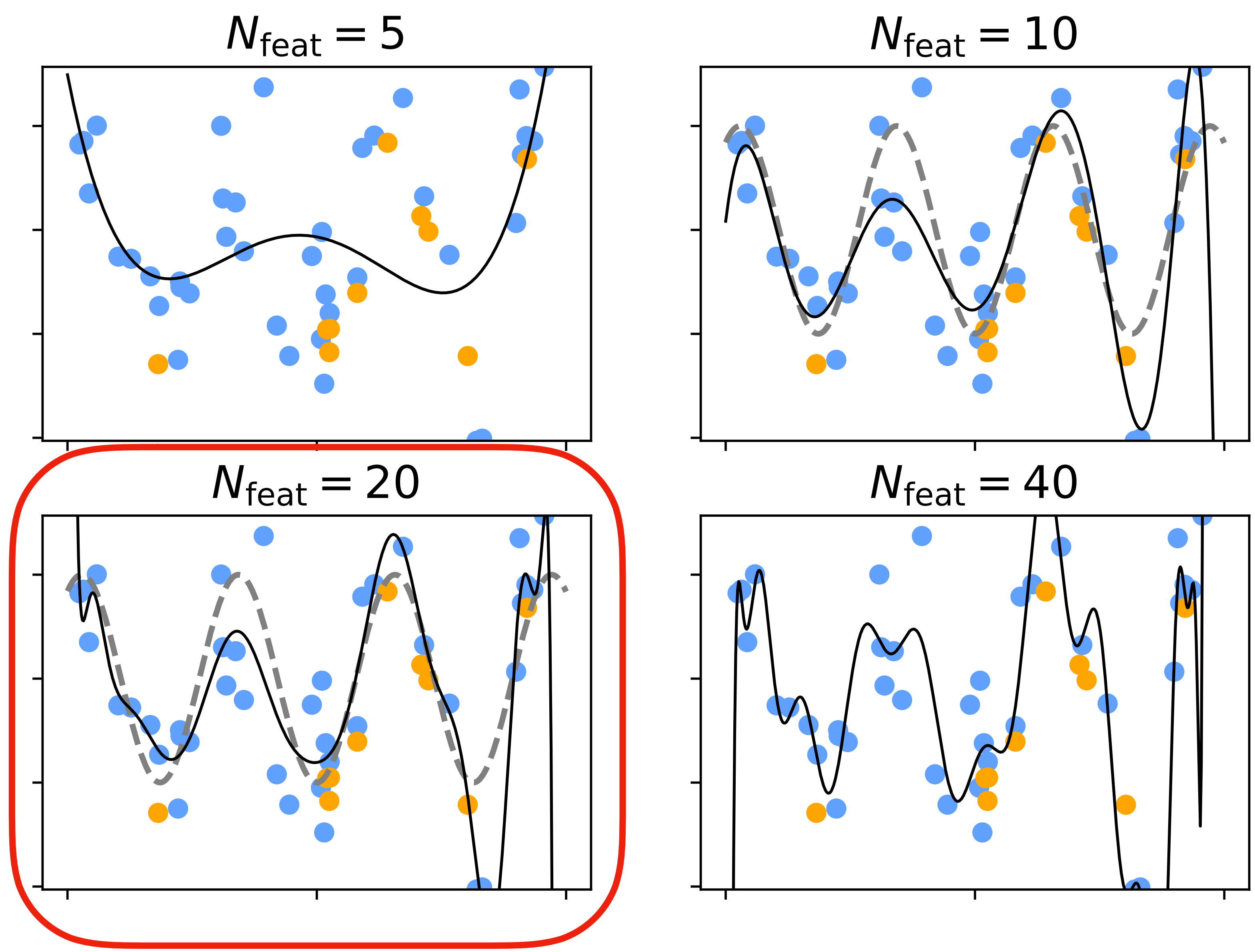
Step 1: Split data (80/20)

Step 2: Define model

Step 3: Optimize on training data

Step 4: Measure R^2 scores on validation data

N	Train	Valid.
5	0.27	-0.07
10	0.73	0.59
20	0.79	0.71
40	0.83	0.42



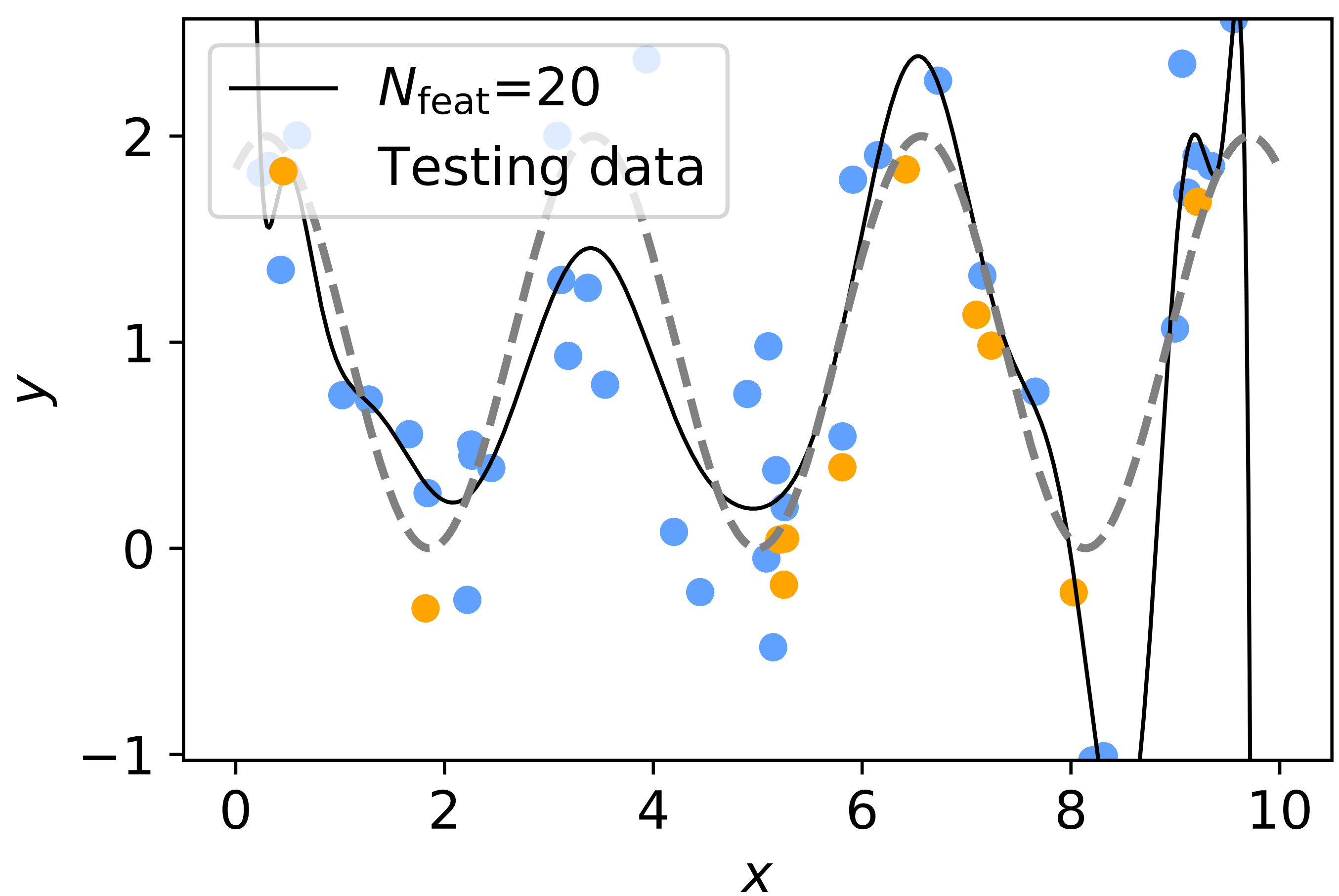
Linear regression, take 2

Step 1: Split data (80/20)

Step 2: Define model

Step 3: Optimize on training data

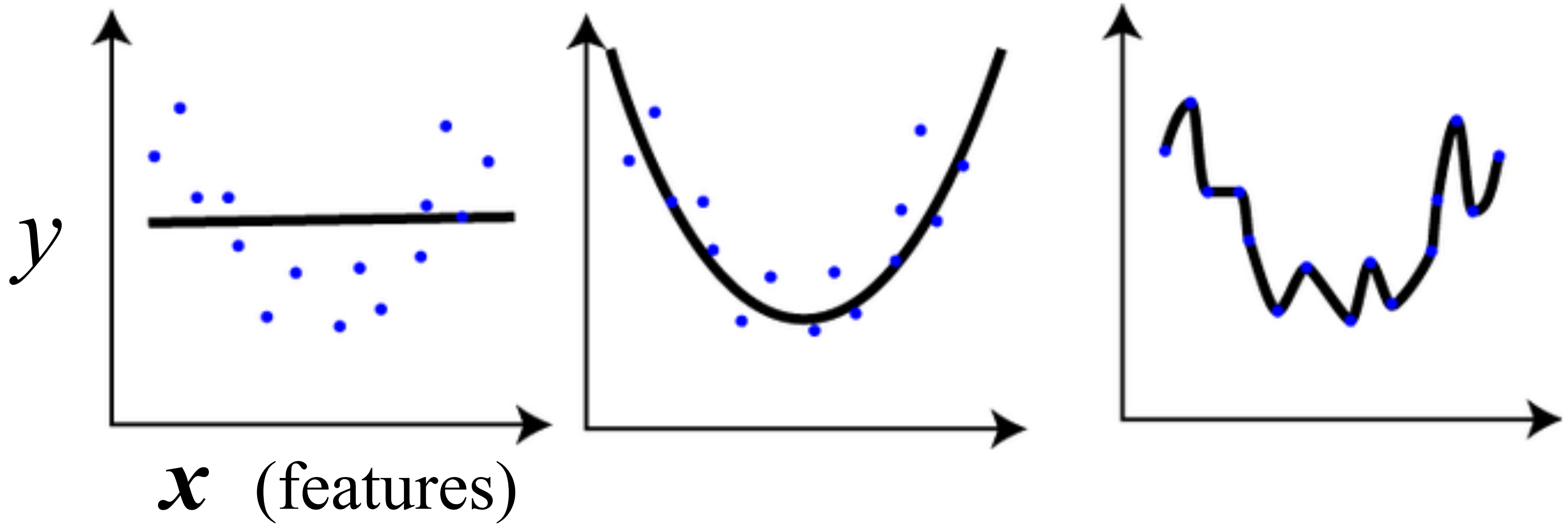
Step 4: Select polynomial order N with validation data



Final scores

	Training	Validation	Actual error
R^2	0.79	0.71	???

A recap



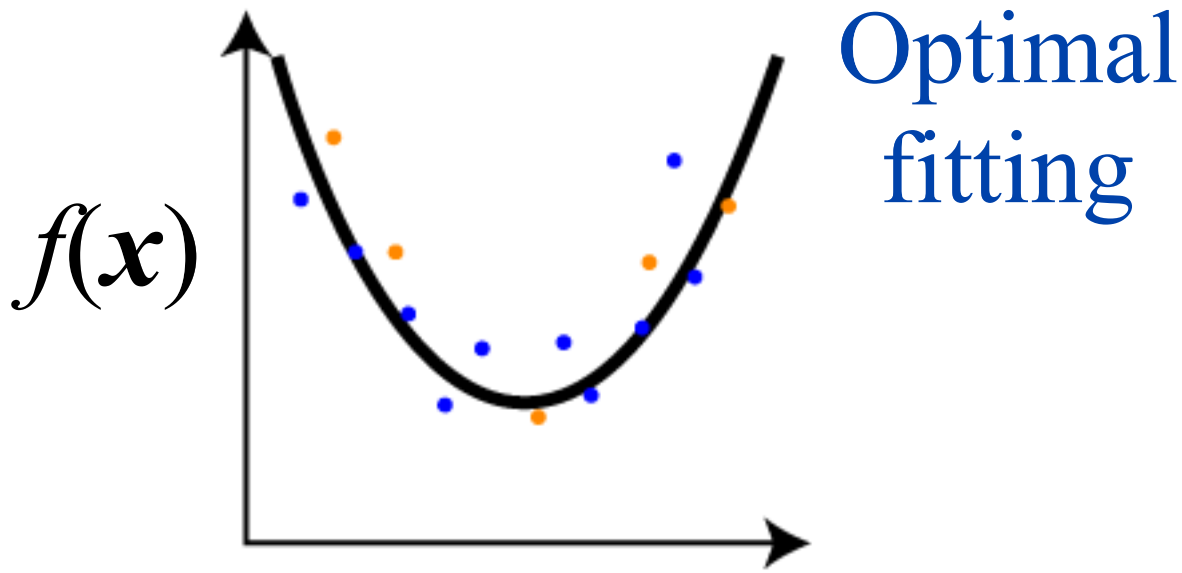
Model capacity

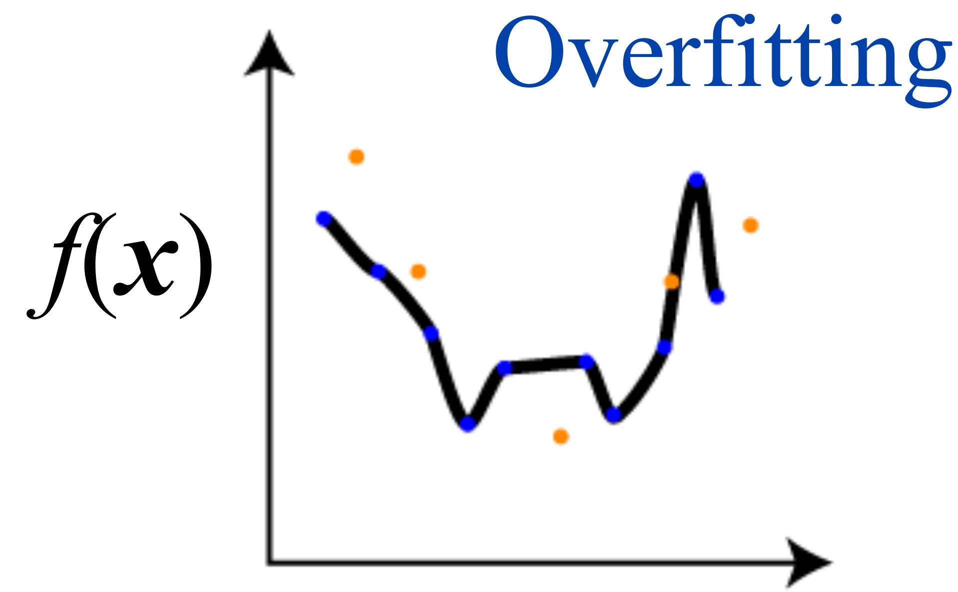
Underfitting → Overfitting

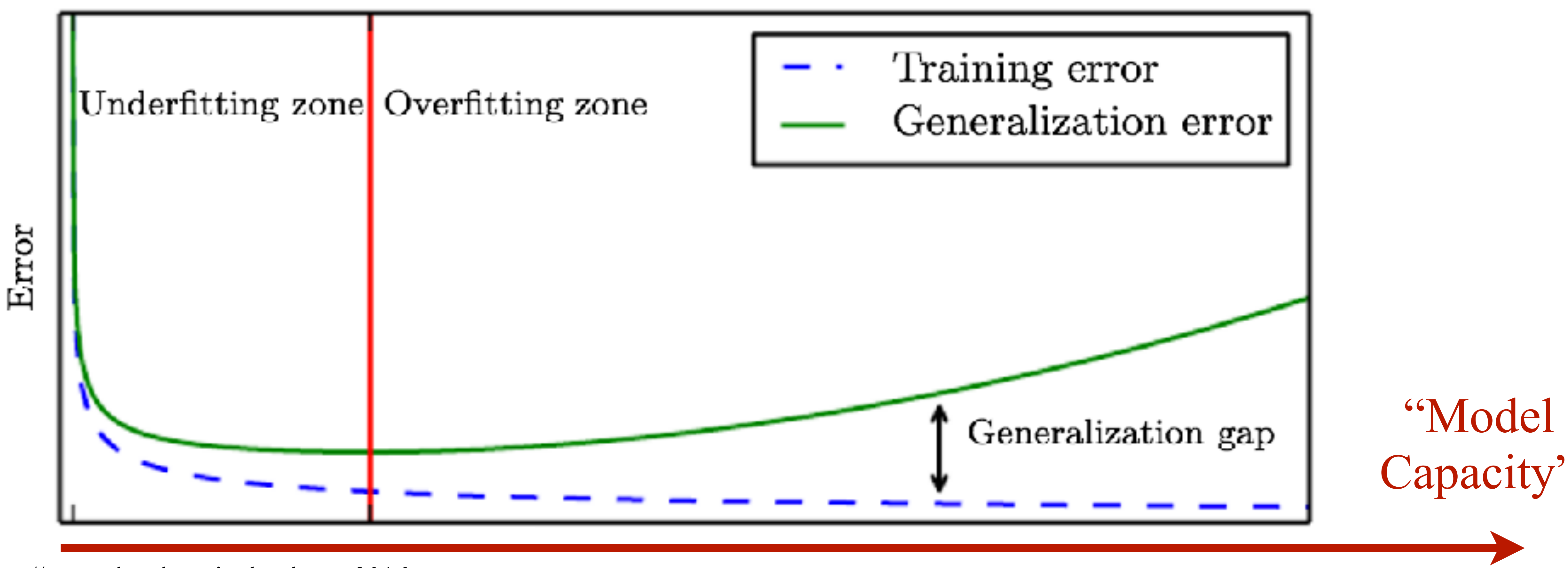
Proper *regularization* of model is context dependent

https://shapeofdata.wordpress.com/2013/03/26/general-regression-and-over-fitting/

- **Step 1**: Randomly split data in ***Training*** and ***Testing*** sets
- **Step 2**: Optimize model from **training** data
- **Step 3**: Estimate *generalization error* on **testing** data



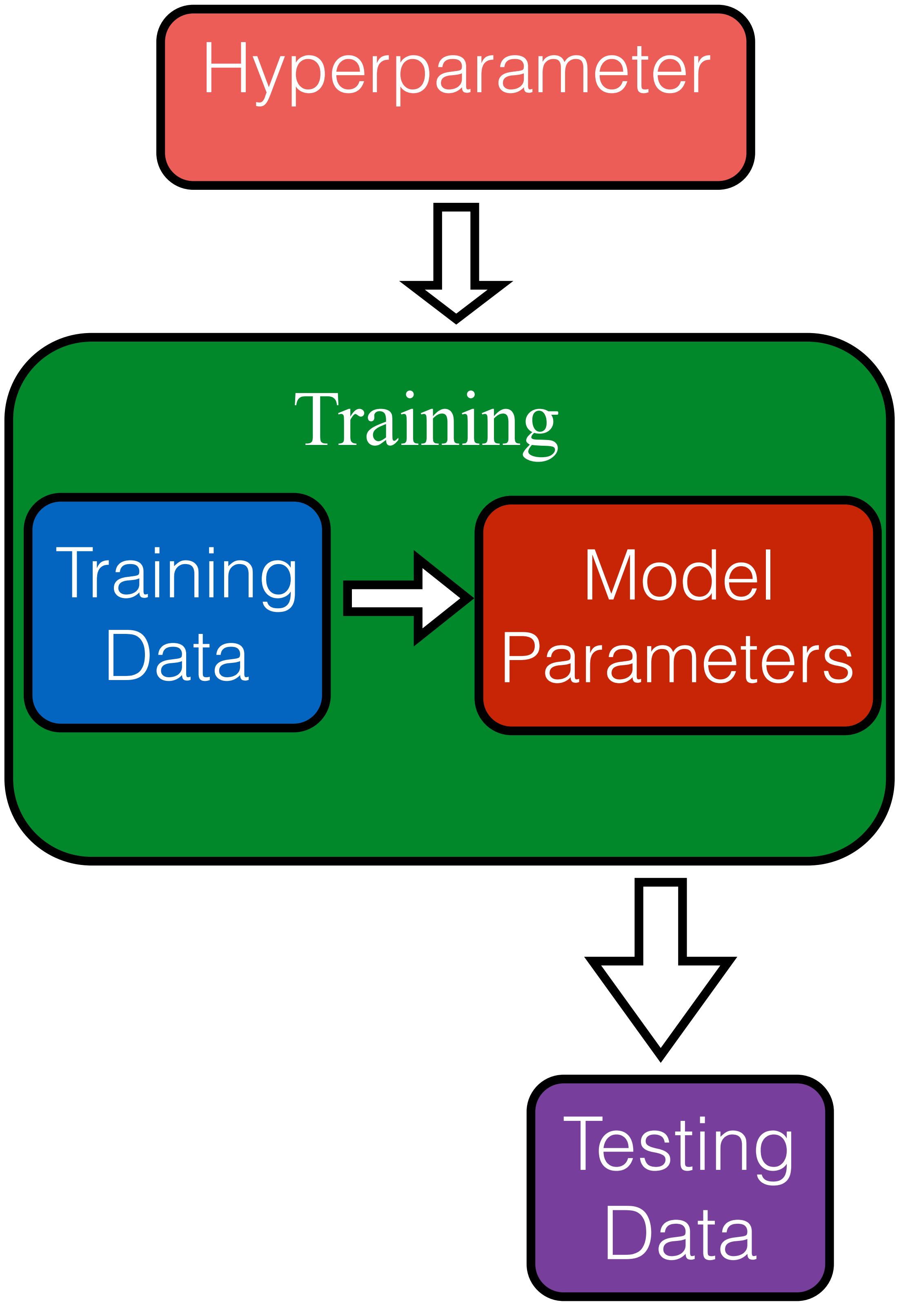




http://www.deeplearningbook.org 2016

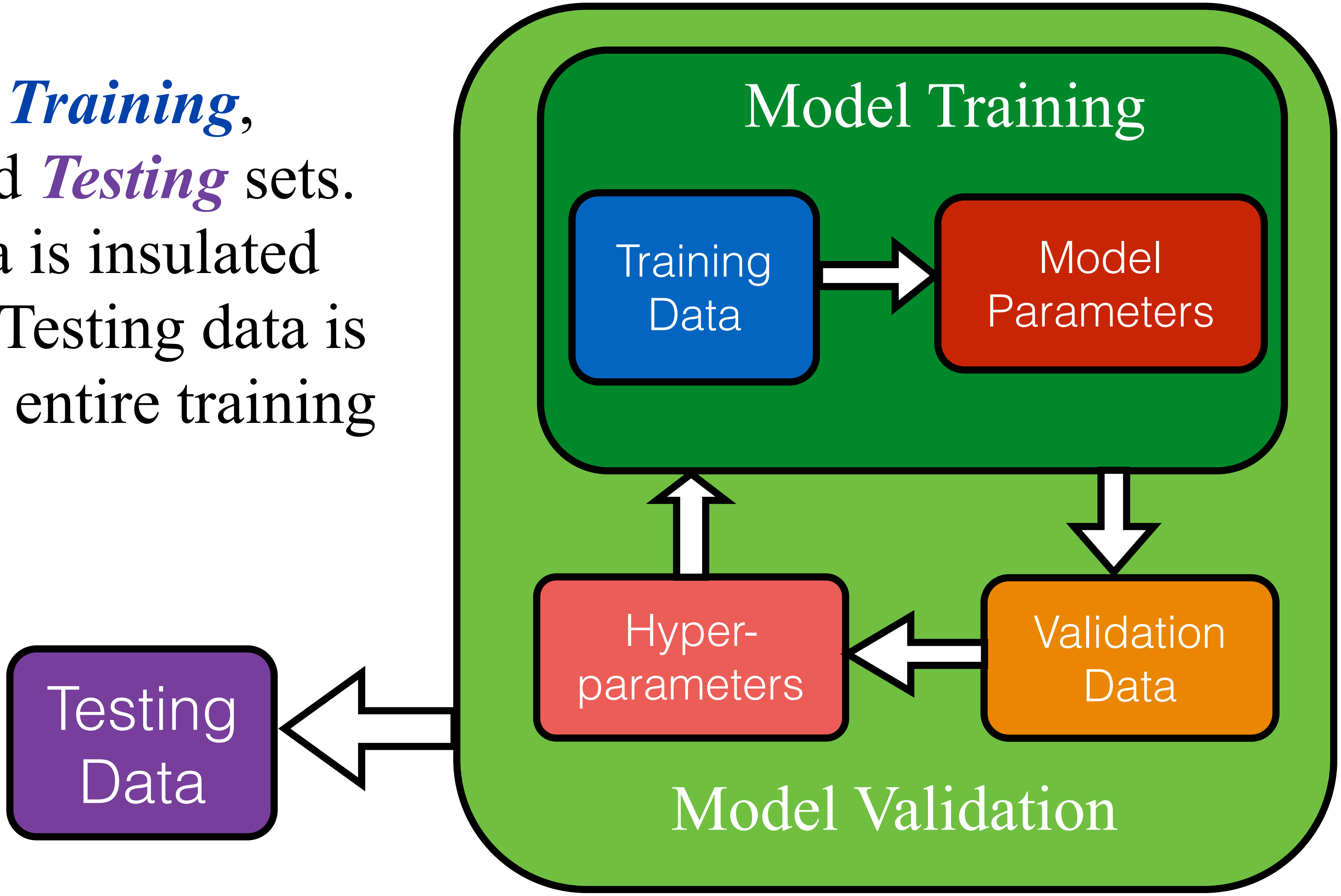
Hyperparameters

- Selected before training
- Often control model capacity (e.g. forcing smoothness)
- For example: **order of polynomial fitting**



Proper model selection loop

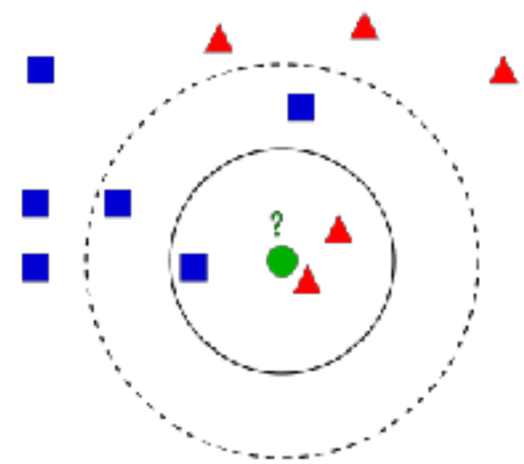
Data split into ***Training***, ***Validation***, and ***Testing*** sets. Validation data is insulated from training. Testing data is insulated from entire training process.



Some interesting ML algorithms

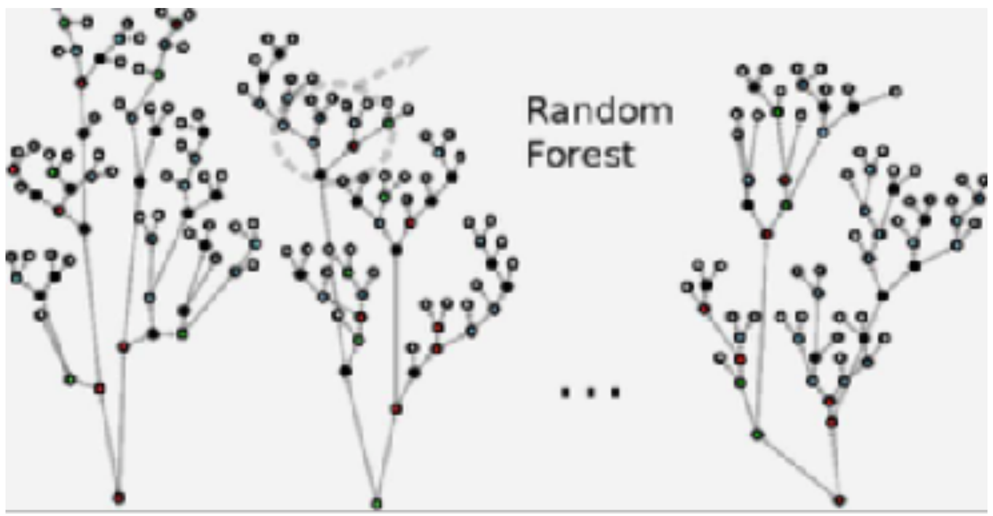
Non-parametric kernel methods

- k-Nearest neighbors
- Support vector machines
- Gaussian process regression



Random forest

- Collection of decision trees



Neural networks

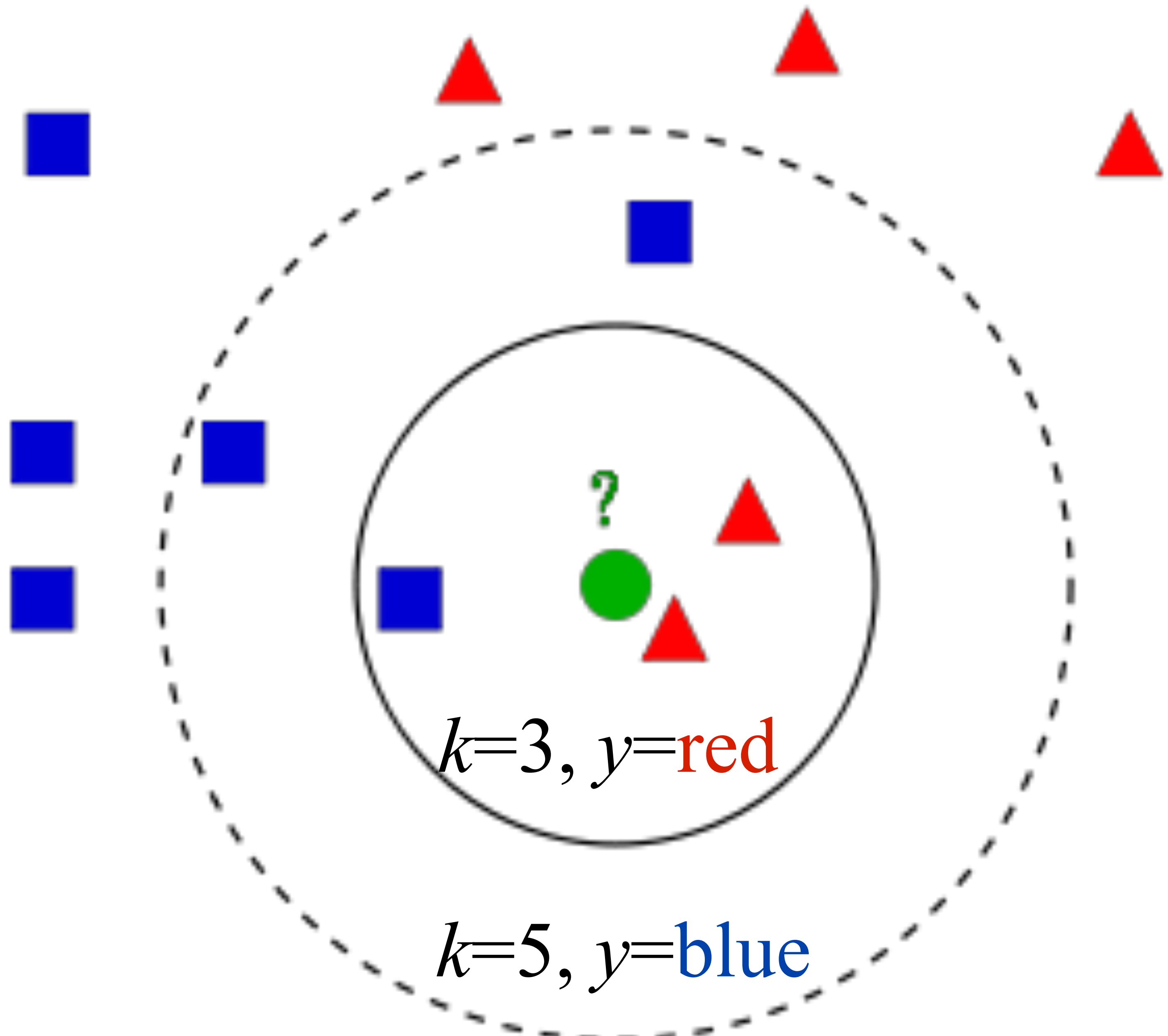
- Deep convolutional nets
- …

Next talk

Kernel methods

Define kernel $K(x_1,x_2)$ to measure *similarity* between x_1 and x_2.

***k*-NN algorithm**: Majority vote of *k*-nearest neighbors.



Effectively *interpolates* nearby data.

Model *grows automatically* with new data.

Other methods: Support Vector Machines, Kernel Ridge Regression, …

Linear (ridge) regression $\hat{\beta} = (XX^T + \lambda)^{-1}X\mathbf{y}$

$$= X^T(X^TX + \lambda)^{-1}\mathbf{y}$$

Points x_i, x_j

Features $\mathbf{f}(x_i) = [f_1(x_i)\ldots f_N(x_i)]$

$\mathbf{f}(x_j) = [f_1(x_j)\ldots f_N(x_j)]$

Similarities $(X^TX)_{i,j} = \mathbf{f}_i \cdot \mathbf{f}_j$

Kernel trick: bypass features to directly define

$$(X^TX)_{ij} = K(x_i, x_j)$$

Arbitrary kernel K completely defines model

$$\boxed{\hat{y}(x) = \sum_i \alpha_i K(x_i, x)}$$

Linear regression, take 3
(Gaussian process version)

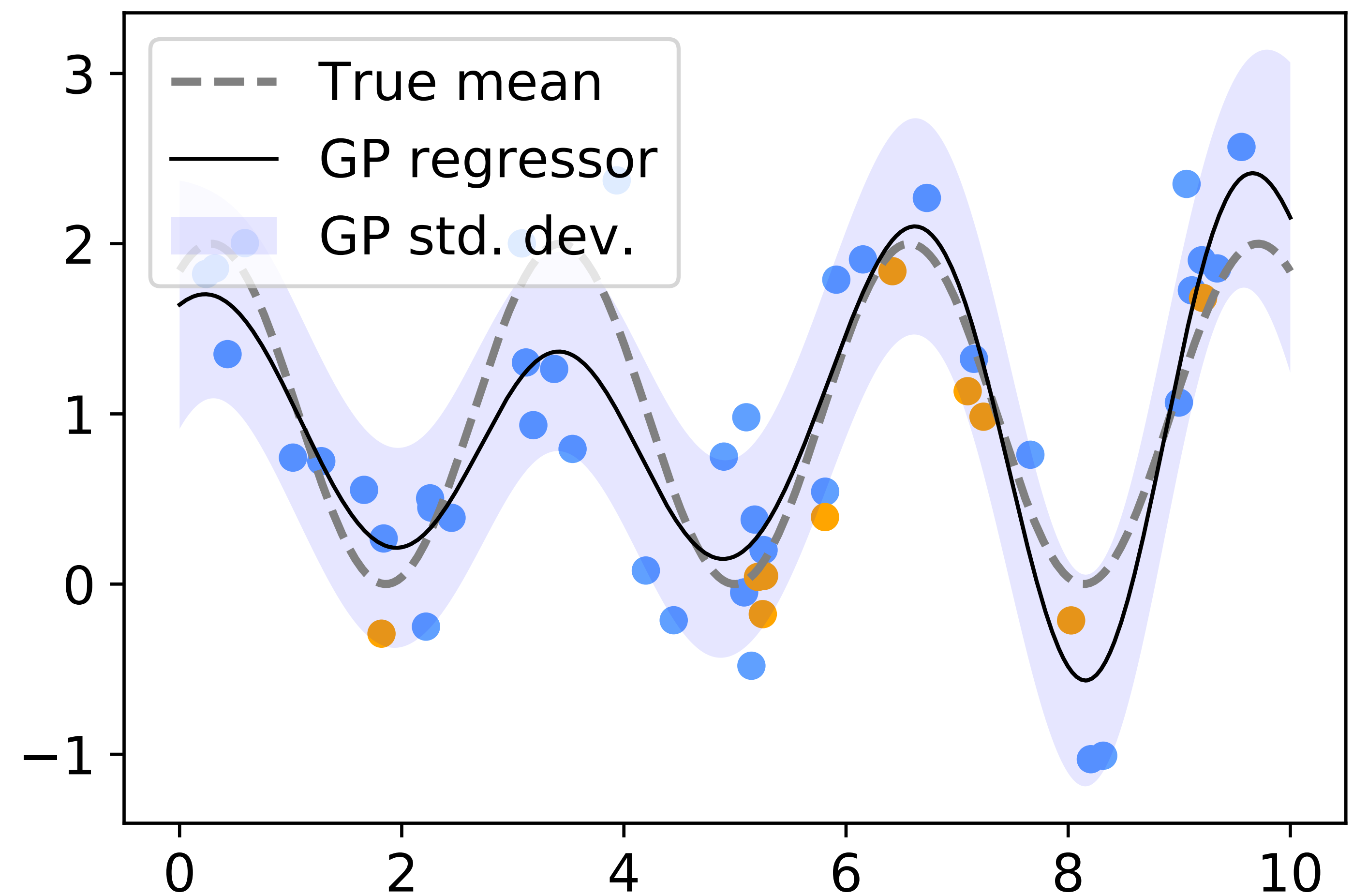
Step 1: Split data (80/20)

Step 2: Define **kernel**

$$K(x_1, x_2) = c_1 e^{-c_2|x_1 - x_2|^2} + c_3\delta_{i,j}$$

Step 3: Optimize on training data.
GP automatically handles hyperparameters c_1, c_2, c_3 !

Step 4: Evaluate performance on testing data.



	Training	Actual error
R^2	0.75	0.71

Linear regression, take 3

By the way, this is super easy in scikit-learn:

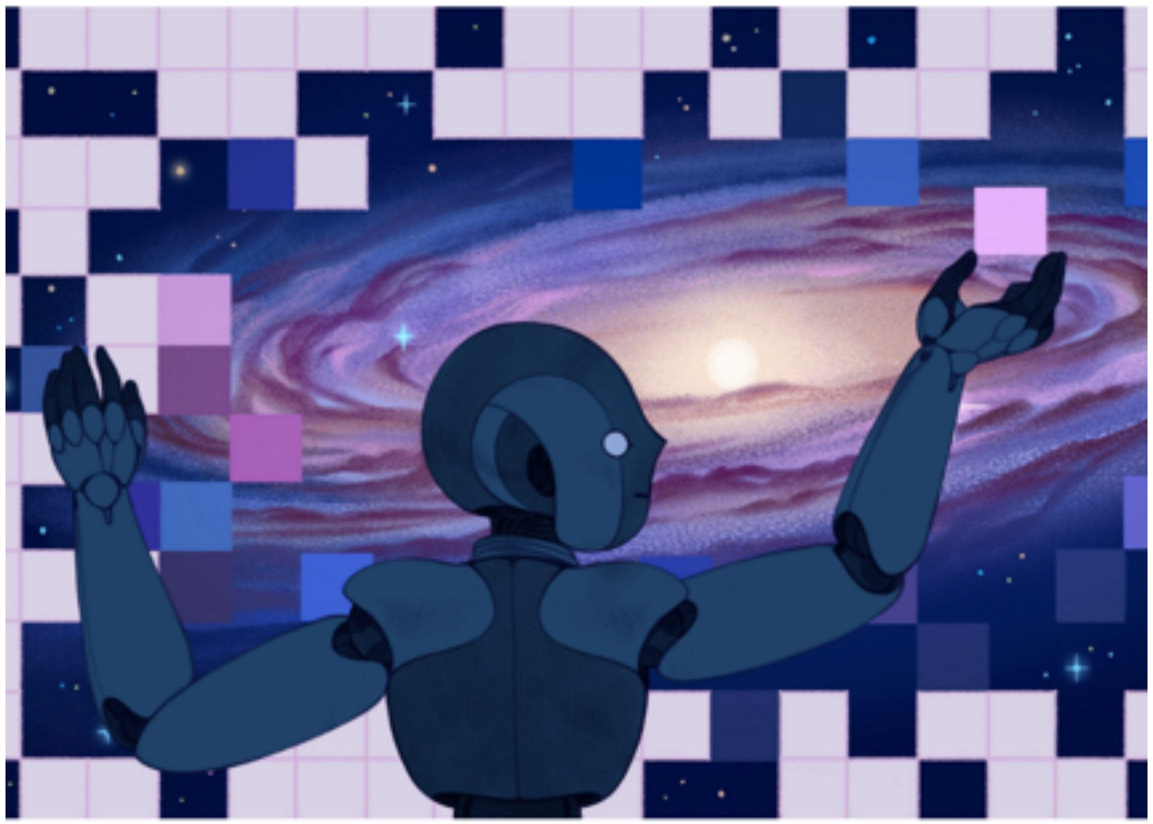
```python
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import WhiteKernel, RBF

gp_kernel = 1.0 * RBF(1) + WhiteKernel(1)
gpr = GaussianProcessRegressor(gp_kernel)

regr = gpr.fit(train[0,:], train[1,:])
print("Kernel params ", regr.kernel_)
print("Training score %f" % regr.score(train[0,:], train[1, :]))
print("Test score     %f" % regr.score(test[0,:], test[1, :]))
```

```
Kernel params  1.41**2 * RBF(length_scale=0.891) + WhiteKernel(noise_level=0.285)
Training score 0.746590
Test score     0.708480
```

Where *can't* we use ML?



Rachel Suggs for Quanta Magazine

MACHINE LEARNING

How Artificial Intelligence Is Changing Science

By DAN FALK

March 11, 2019

The latest AI algorithms are probing the evolution of galaxies, calculating quantum wave functions, discovering new chemical compounds and more. Is there anything that scientists do that can't be automated?

https://www.quantamagazine.org/how-artificial-intelligence-is-changing-science-20190311/

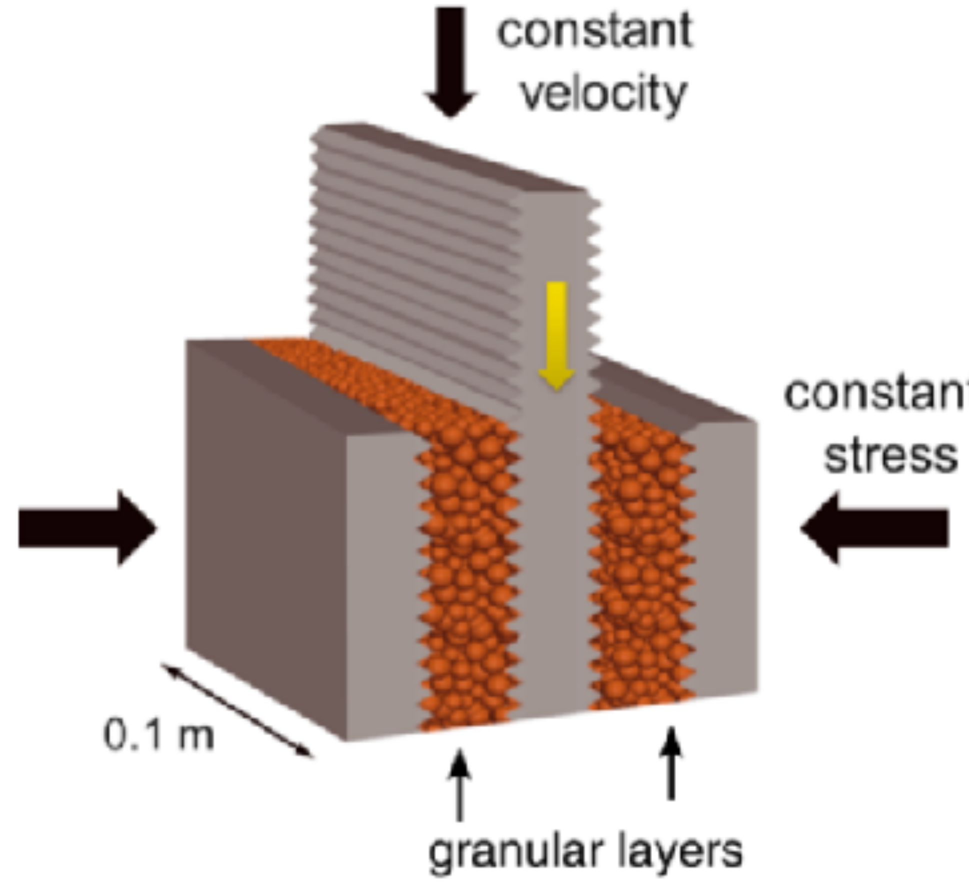
Predicting Lab-quakes

Geophys. Res. Lett. 44, 9276 (2017) [arXiv:1702.05774]

Claudia H.

Bertrand R.-L.

Nick L.



Source: Johnson, P., et al. (2013), Acoustic emission and microslip precursors to stick-slip failure in sheared granular material, Geophys. Res. Lett., 40 5627–5631

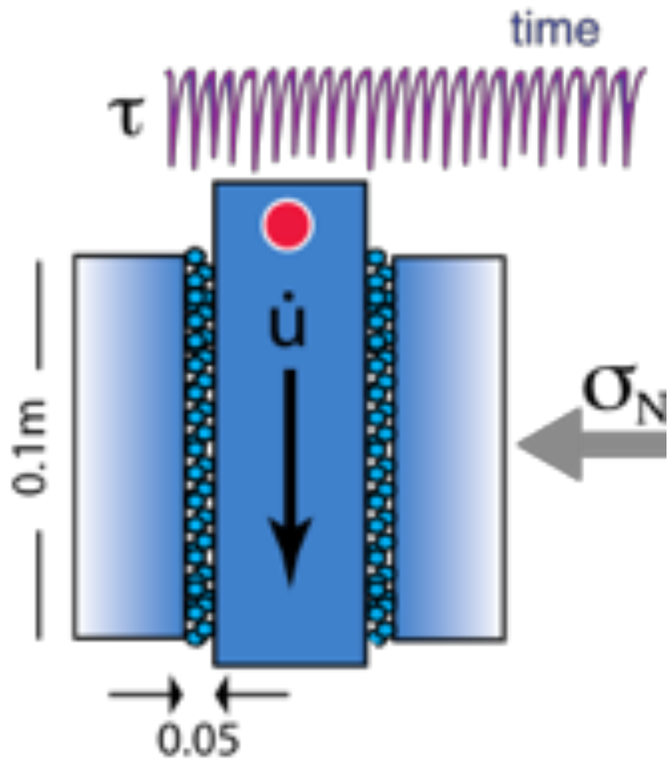
Media buzz

https://www.scientificamerican.com/article/can-artificial-intelligence-predict-earthquakes/

https://www.technologyreview.com/s/603785/machine-learning-algorithm-predicts-laboratory-earthquakes/

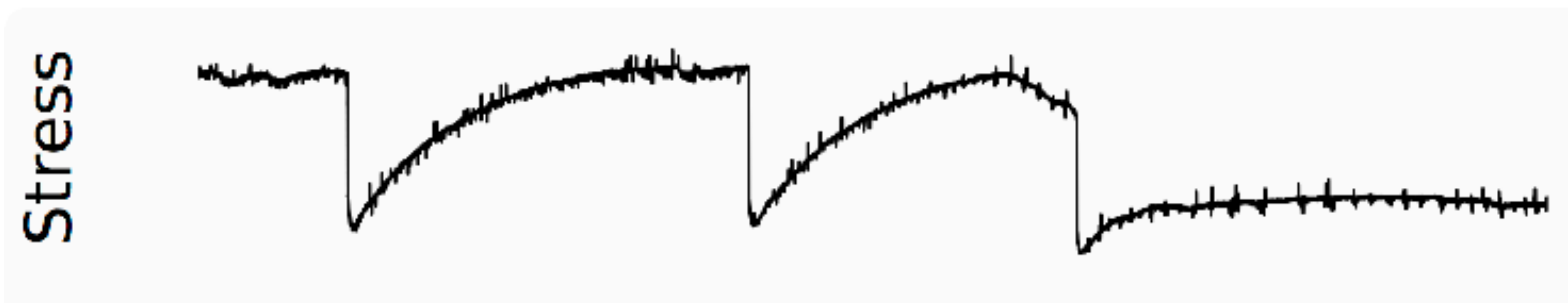
http://cacm.acm.org/news/213876-can-artificial-intelligence-predict-earthquakes/fulltext

http://www.msn.com/en-us/weather/topstories/could-artificial-intelligence-help-predict-earthquakes/ar-AAmZKLe



- Central "loader" plate pushed down at constant velocity
- Normal force applied on side plates
- Glass beads ("gouge") between plates

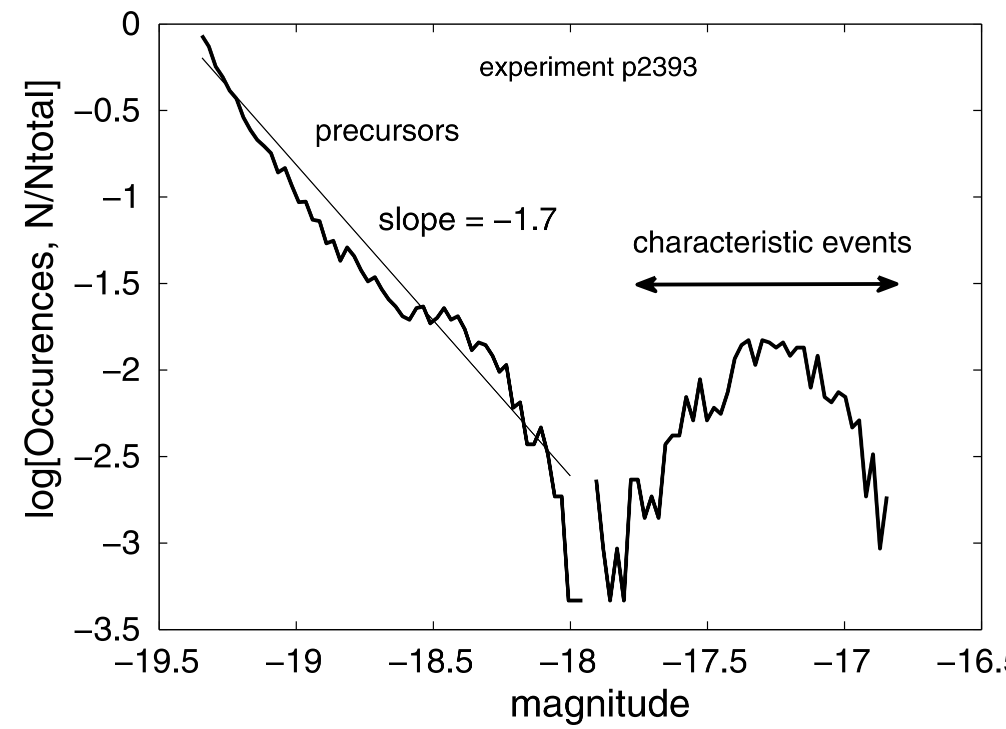
- Force ("shear stress") on driving block

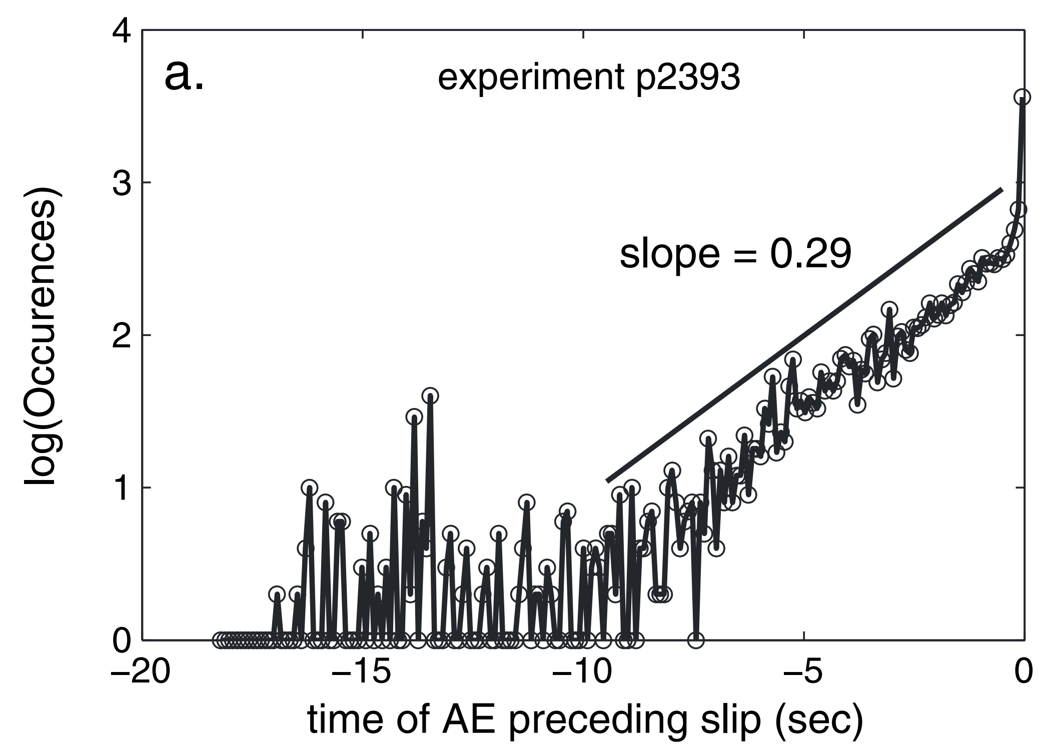


- Acoustic emission

Precursor activity

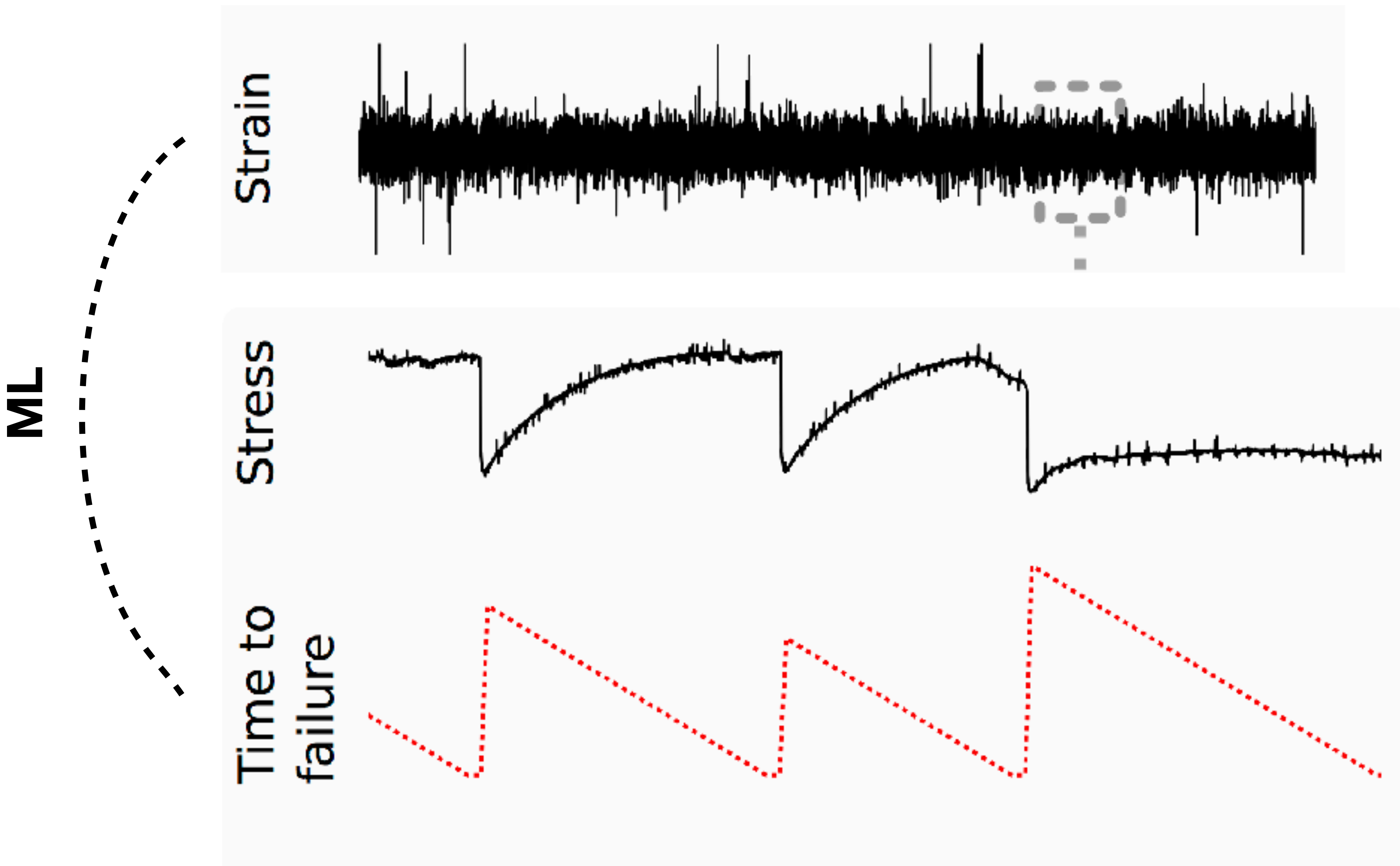
- Impulsive precursors follow Gutenberg–Richter (power law) decay
- Rate of precursors grows exponentially before characteristic event (lab-quake)





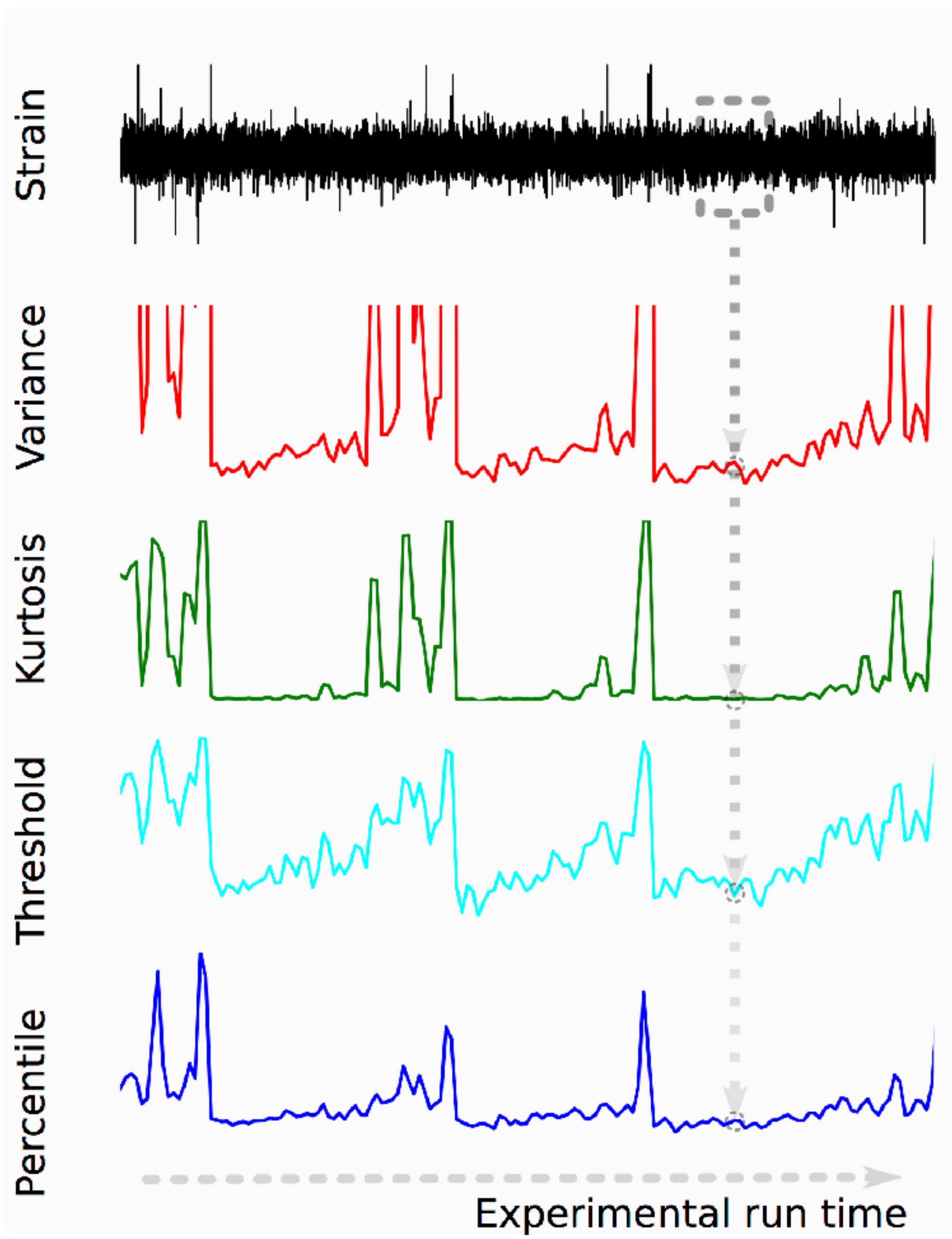
Johnson, P. A., et al. "Acoustic emission and microslip precursors to stick−slip failure in sheared granular material." Geophysical Research Letters 40.21 (2013): 5627-5631.

Goal: Predict time until next failure (stress drop) from *local* window



Features

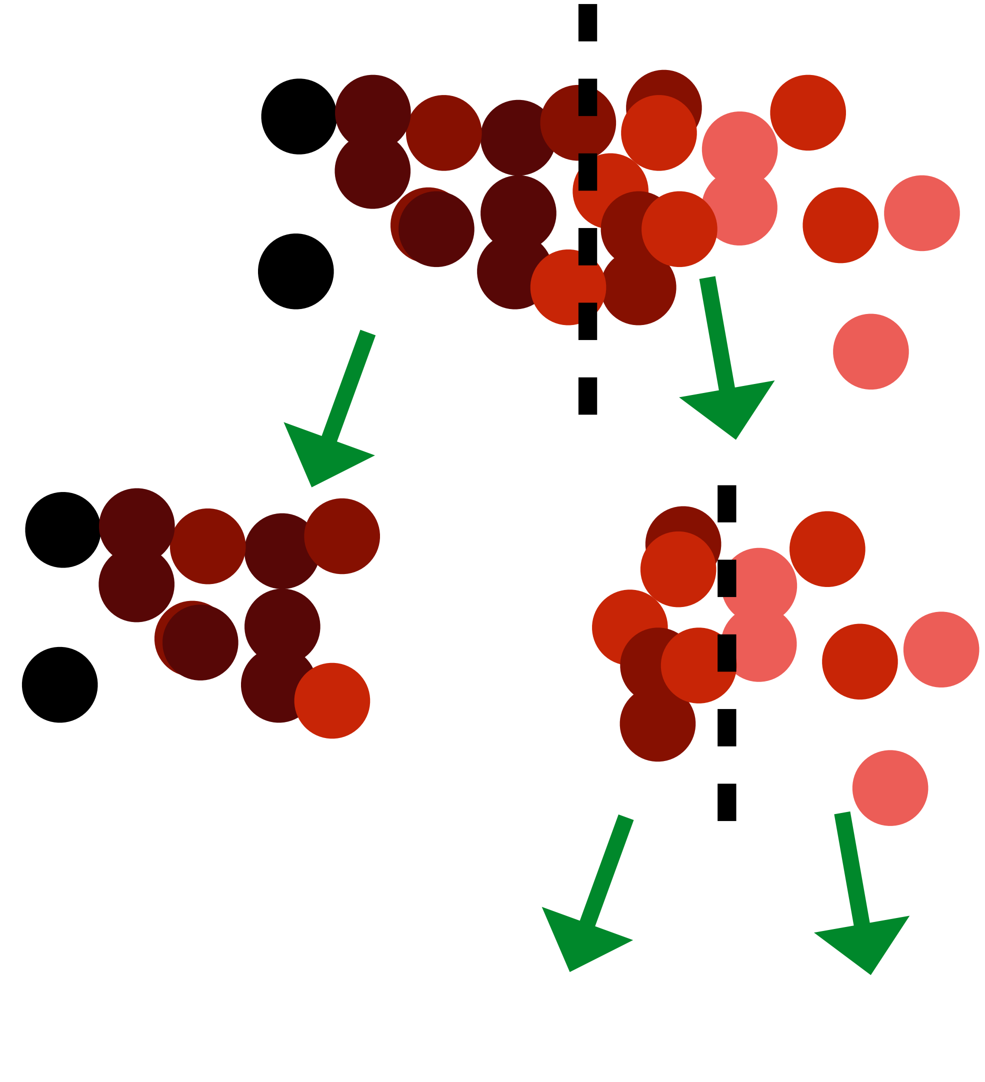
- “Now” prediction, window size ~1/10 cycle
- Extract from acoustic emissions



- Centered moments: variance, skew, kurtosis…
- Amplitude maximum, minimum, extreme quantiles
- Counts over and under various thresholds
- Time correlation measures — power spectrum, autocorrelation…

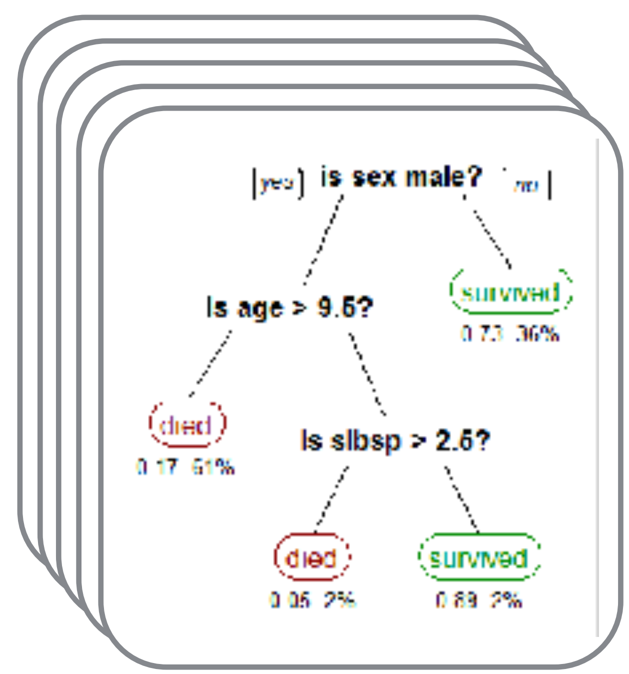
Decision Trees

- Recursive splitting of training data
- Splits maximize difference between the two branches of the training data
- Leaves predict sample average of training data



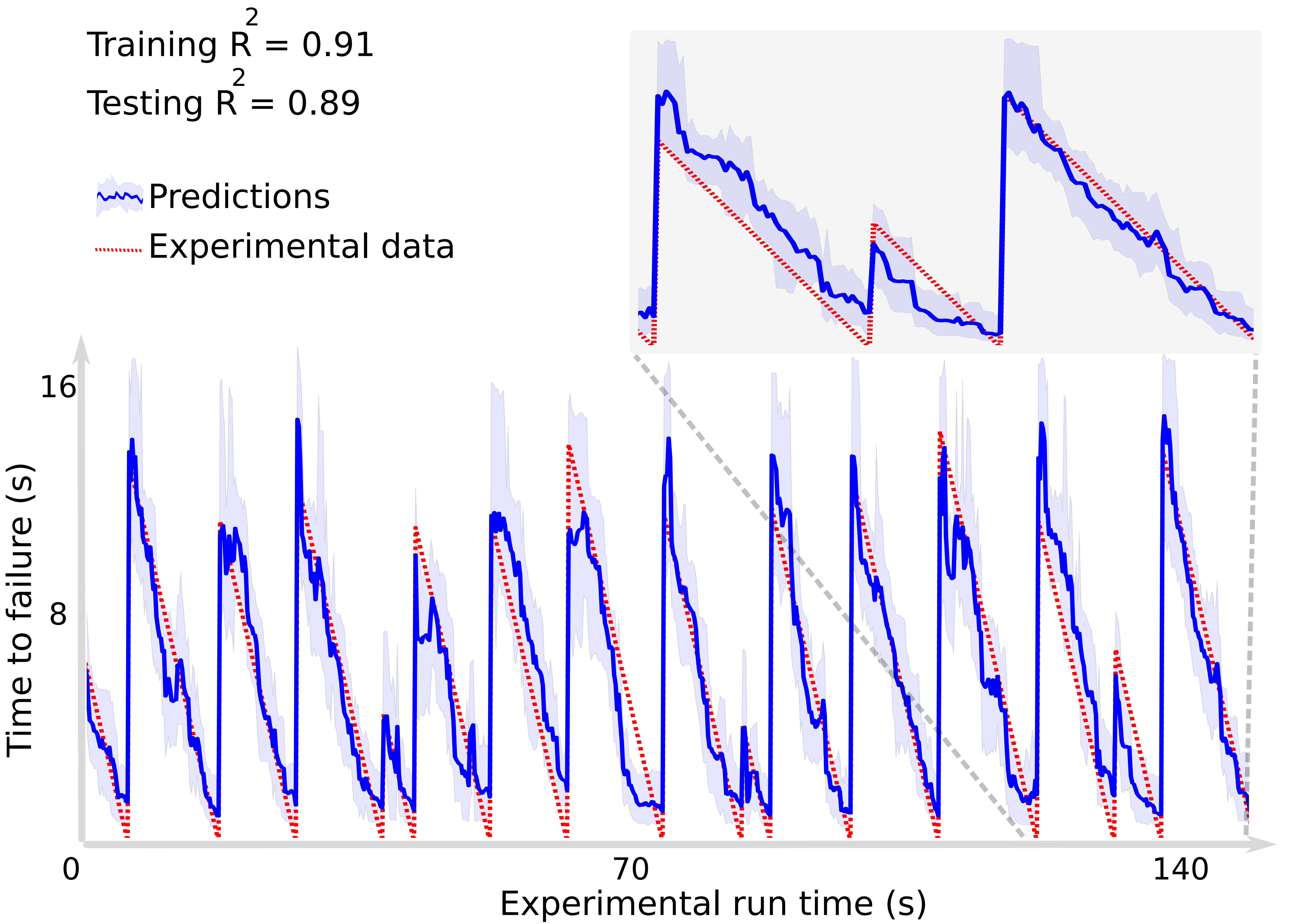
Random forest

- Average over many decision trees

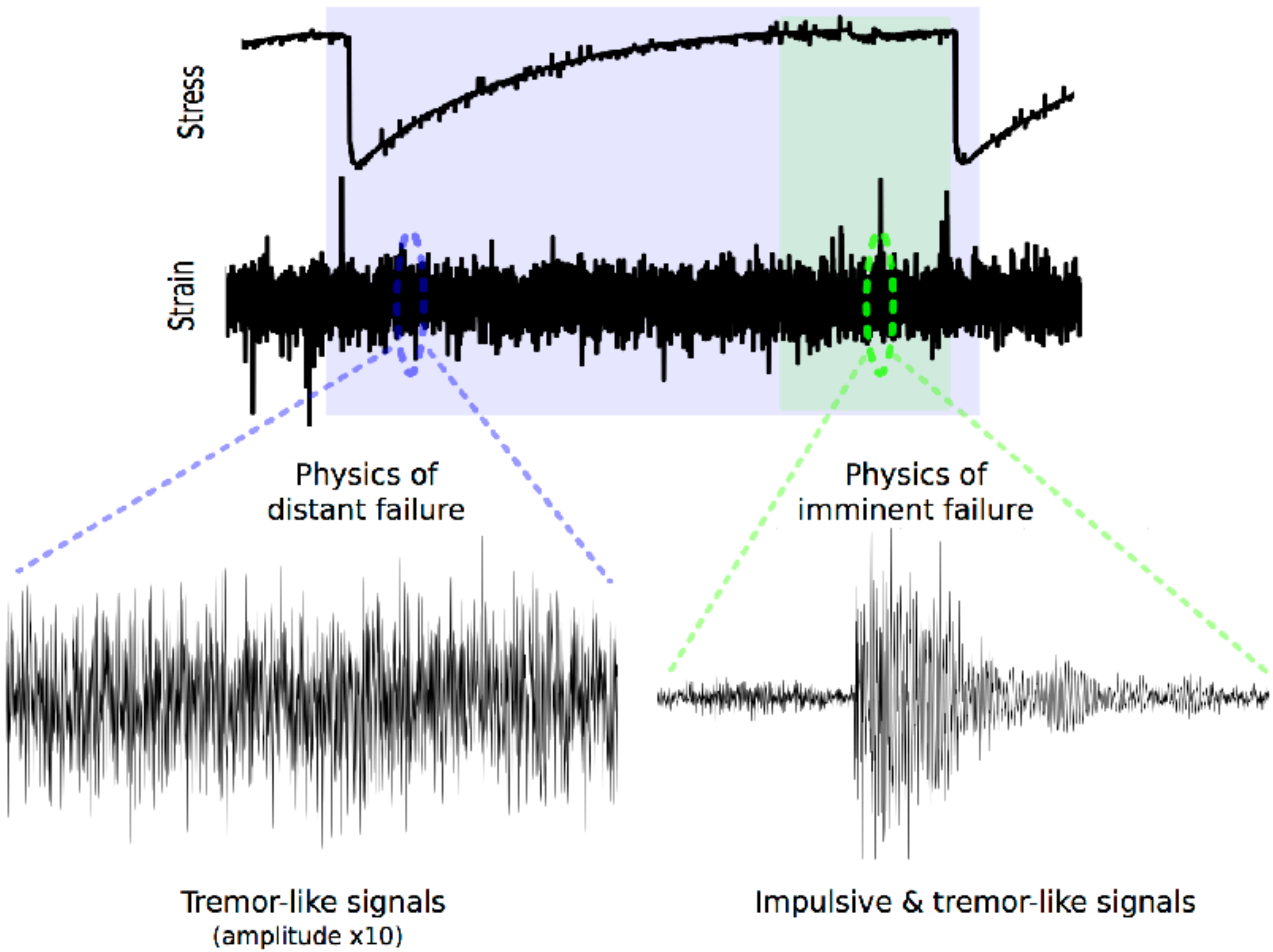


Eg. Survival odds of passengers of the *Titanic*

Predictions



Physics of failure



ML, a bird's eye view:

- *Good data* is crucial (even more important than algorithms).
- *Training* is simply optimization of a *cost function*.
- The essence of machine learning is empirical tuning of model complexity (hyperparameter selection) using validation data.
- Keep *test data* separate from training/validation data!
- scikit-learn.org is a great place to start. ***Gaussian Process*** and ***Random Forest*** methods are particularly easy.
- Neural networks (next talk) are amazingly powerful with large datasets, but take a lot more fiddling.