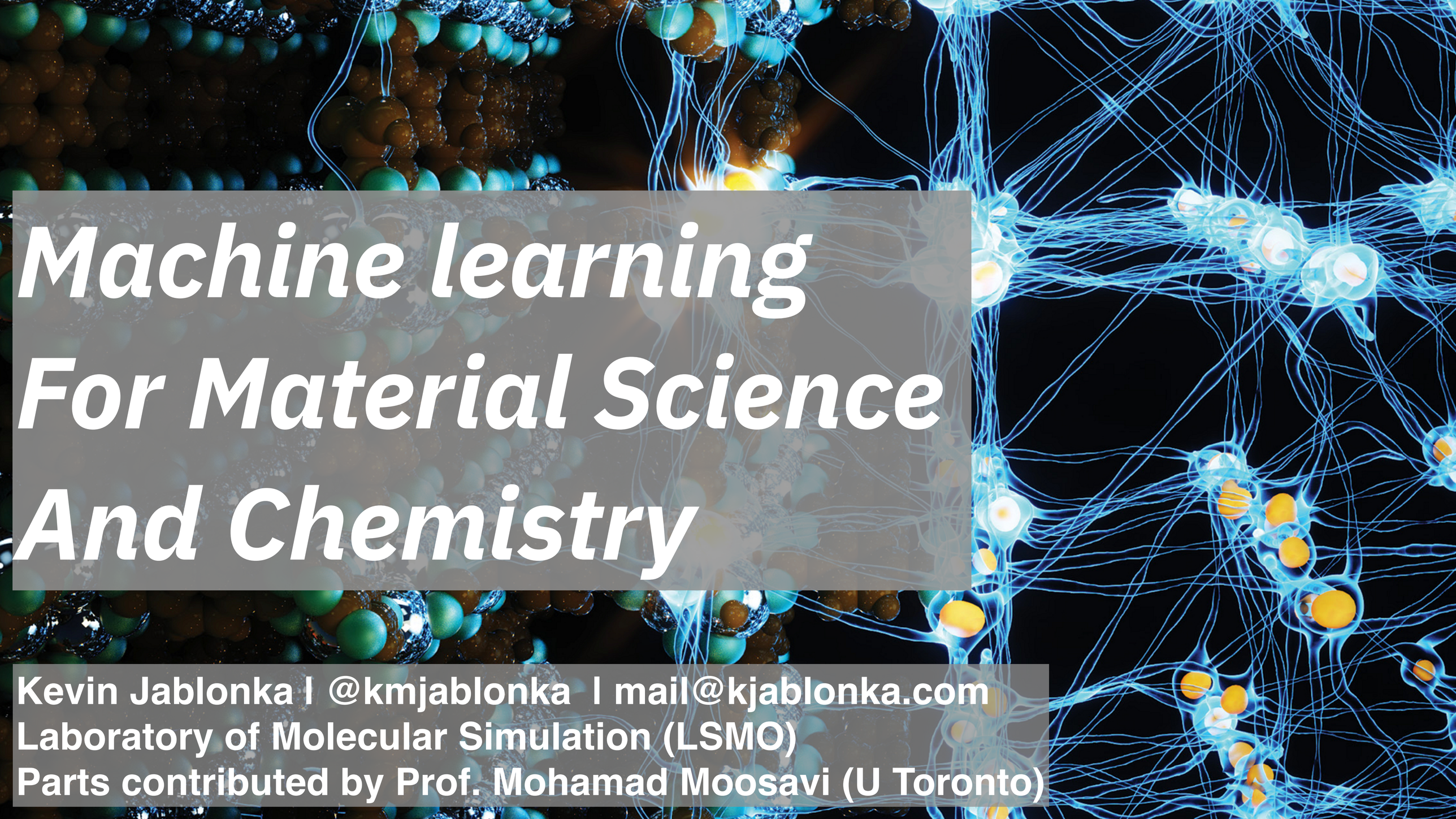
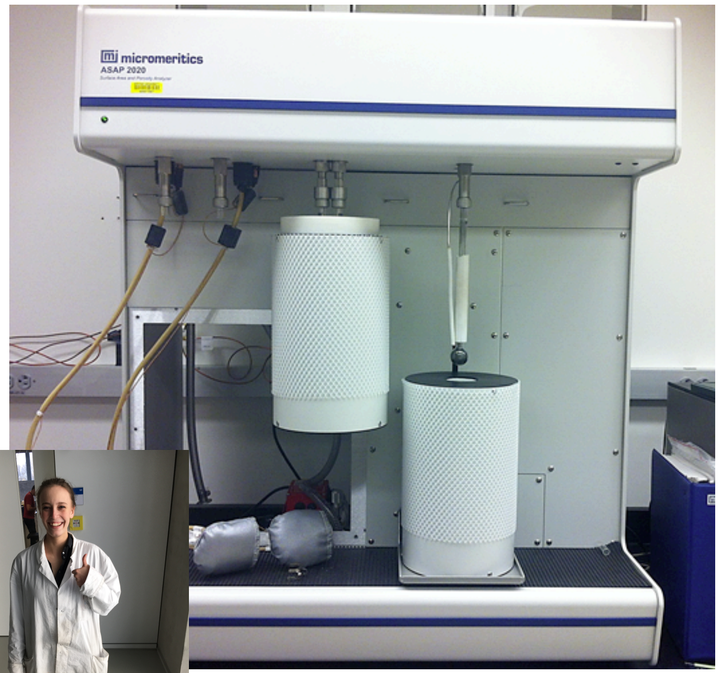


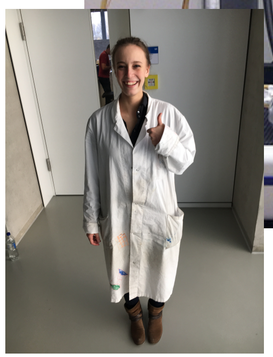
Machine learning For Material Science And Chemistry

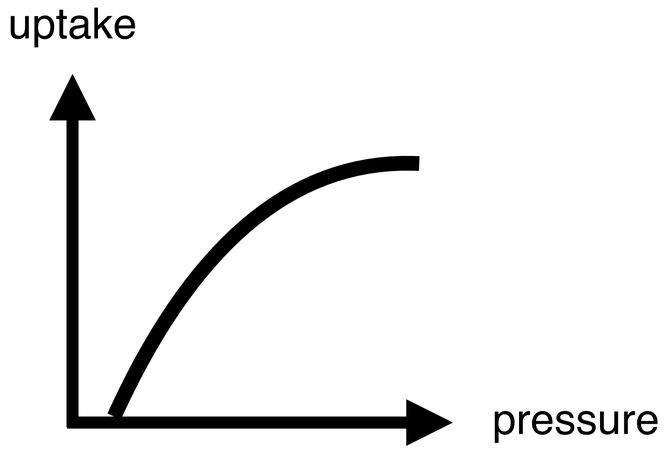
Kevin Jablonka I @kmjablonka I mail@kjablonka.com
Laboratory of Molecular Simulation (LSMO)
Parts contributed by Prof. Mohamad Moosavi (U Toronto)

Testing a MOF for carbon capture

Experiment: Weeks/Months



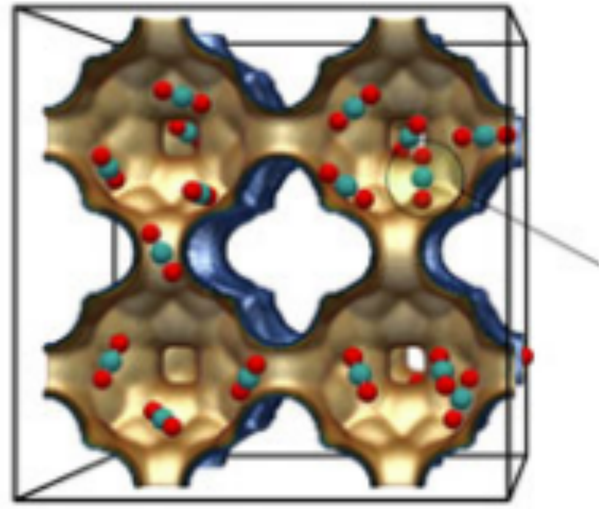




Computer simulations: Hours/Days

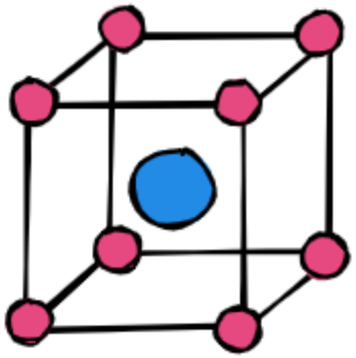
$$\frac{\mathrm{acc}(o \to n)}{\mathrm{acc}(n \to o)} = \mathrm{e}^{-\beta\left[U_n(\ \mathrm{s}^N;\mathbf{h}) - U_o(\mathbf{s}^N;\mathbf{h})\right]}$$

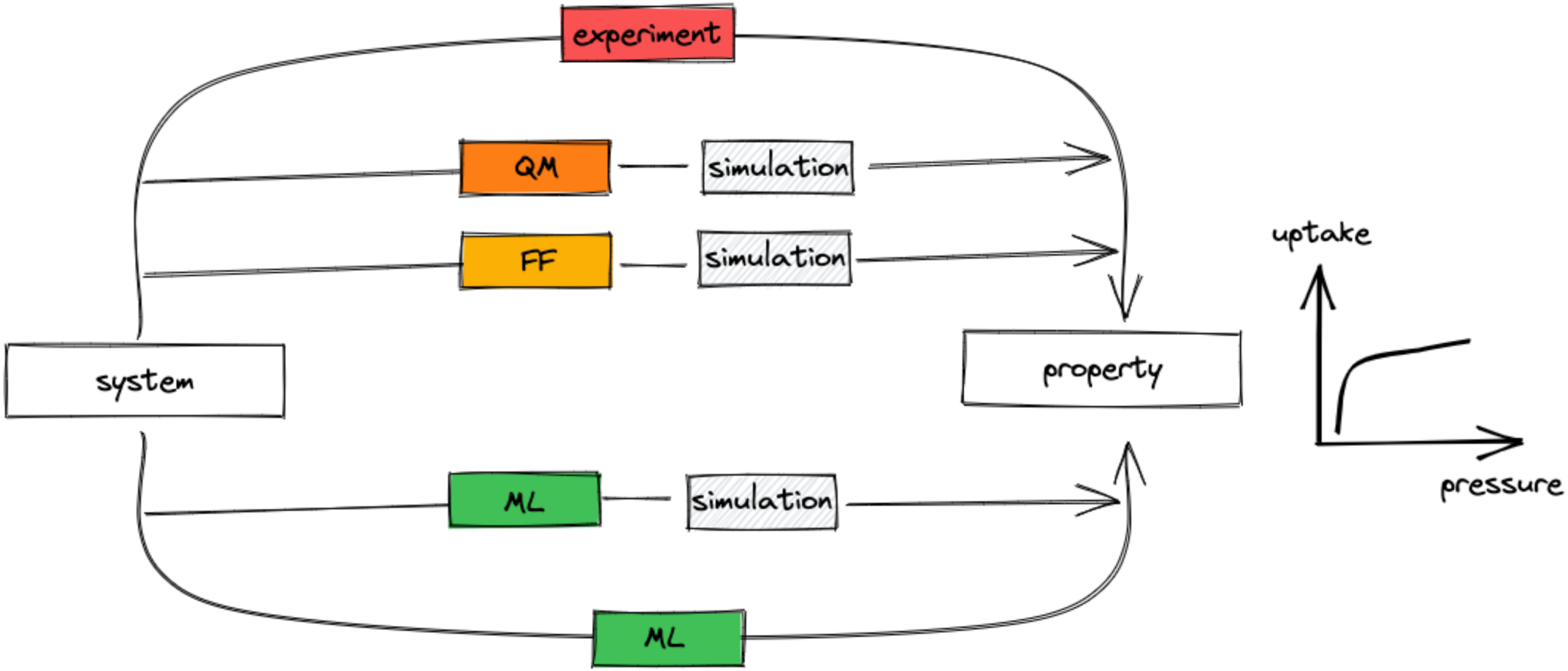
$$\frac{\mathrm{acc}(o \to n)}{\mathrm{acc}(n \to o)} = \mathrm{e}^{-\beta\left[U_n(\ \mathrm{s}^{N+1};\mathbf{h}) - U_o(\mathbf{s}^N;\mathbf{h})\right]}$$



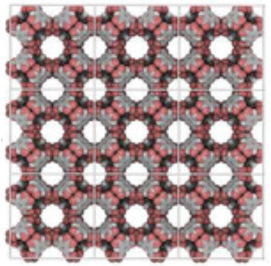
Molecular Simulation **2013**, *39* (14–15), 1253–1292.

Where does ML fit in this picture?





Supervised ML workflow



$$X = \begin{pmatrix} x^{1,1} & x^{1,1} & \dots & x^{1,p} \\ x^{2,1} & x^{2,1} & \dots & x^{2,p} \\ \vdots & \ddots & & \vdots \\ x^{n,1} & x^{n,2} & \dots & x^{n,p} \end{pmatrix} \quad Y = \begin{pmatrix} y^1 \\ y^2 \\ \vdots \\ y^n \end{pmatrix}$$

p features

n observations

Cu1 Cu 0.04746(3) 0.70528(2) 0.5000
Cu2 Cu 0.06207(2) 0.60834(2) 0.5000
Cu3 Cu 0.377374(18) 0.327333(17) 0.145781(12)
Cu4 Cu 0.337355(17) 0.272144(17) 0.196102(12)
O1 O 0.11047(12) 0.62203(11) 0.46304(8)
O2 O 0.09801(12) 0.70448(11) 0.46334(7)
O3 O 0.17739(12) 0.81124(11) 0.37594(7)
O4 O 0.21888(12) 0.77657(11) 0.33090(7)
O5 O 0.27044(11) 0.59561(11) 0.21620(7)
O6 O 0.32403(11) 0.56455(10) 0.17643(7)
O7 O 0.35334(12) 0.38519(11) 0.17199(8)
O8 O 0.32649(12) 0.33741(11) 0.21710(7)
O9 O 0.14566(12) 0.28556(11) 0.33326(8)
O10 O 0.10519(11) 0.23780(11) 0.37266(7)
O11 O 0.00204(12) 0.30975(11) 0.46366(7)

1. Feeding structures into models

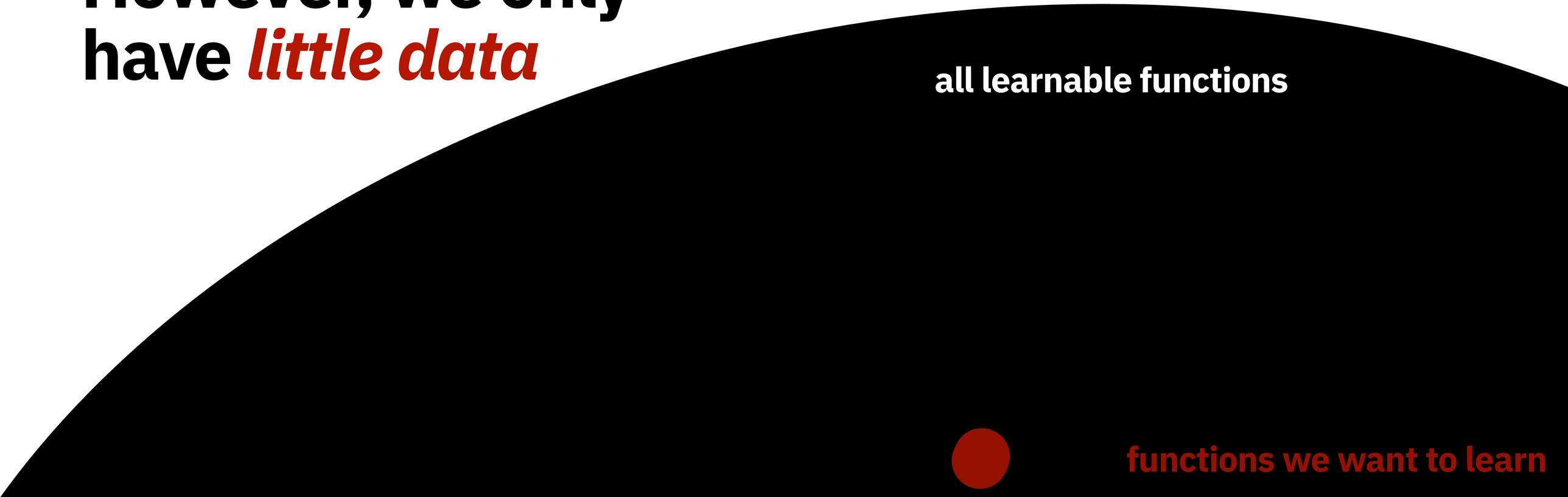
Learning Objectives:

- Need for featurization
- Equivariances/invariances
- Typical assumptions
- Examples of descriptors

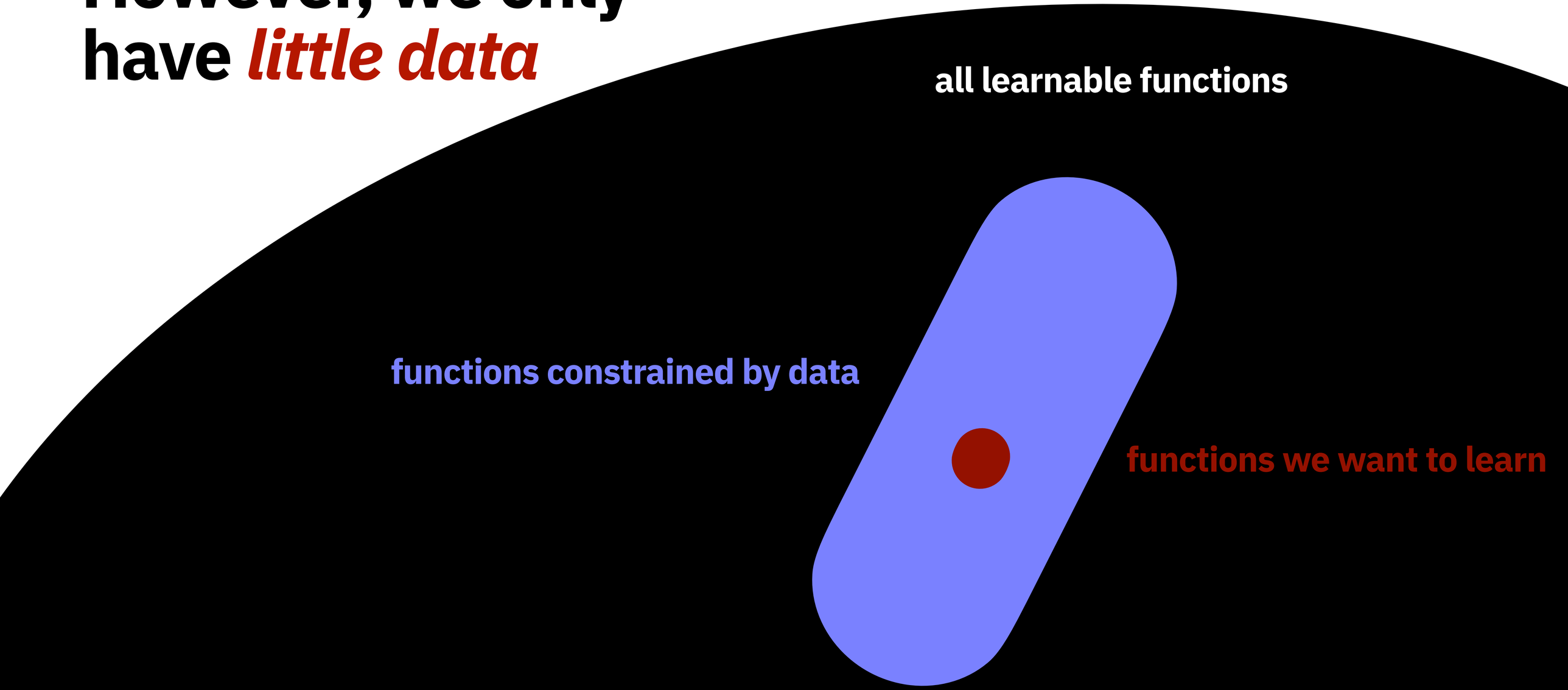
However, we only have *little data*

all learnable functions

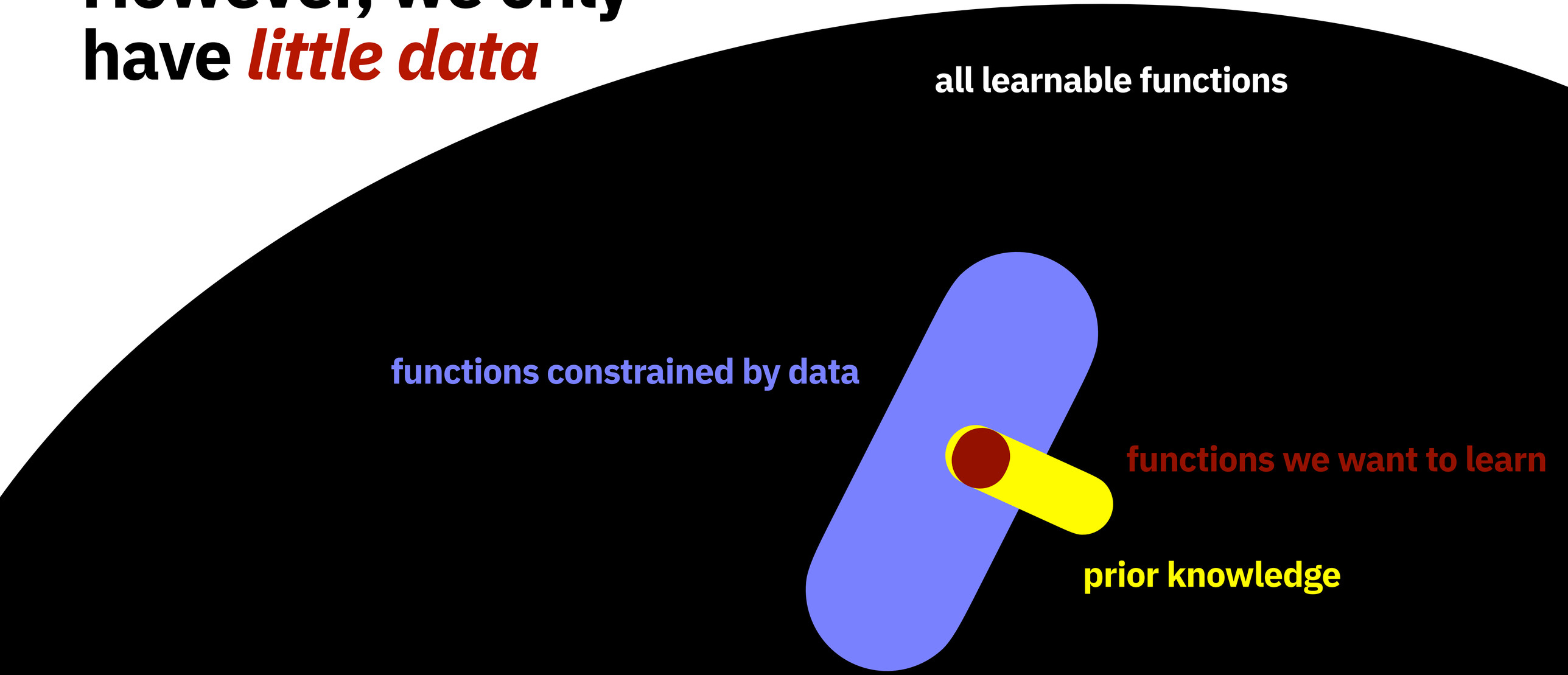
However, we only have *little data*



However, we only have *little data*



However, we only have *little data*



Coding example

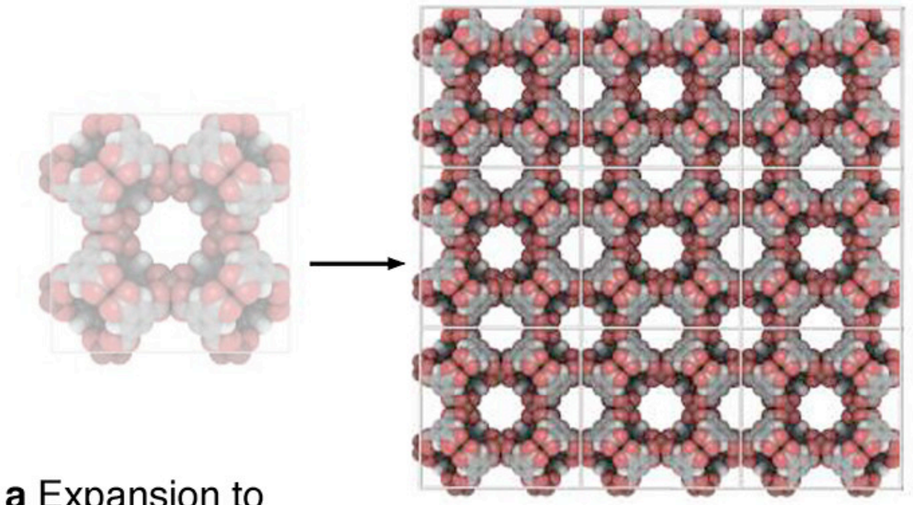
Inductive biases and wishlist

We would like to satisfy basic symmetries. Typically invariance/ equivariance wr.t.

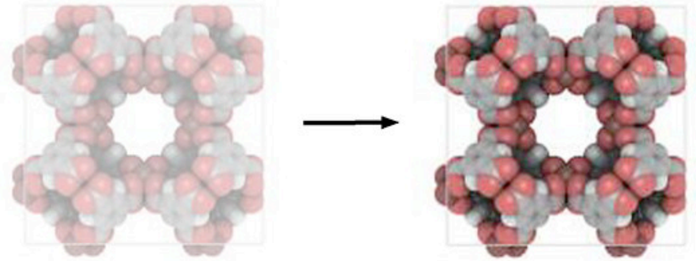
- translation
- rotation
- permutation
- expansion to supercell

Additionally, in some cases we want

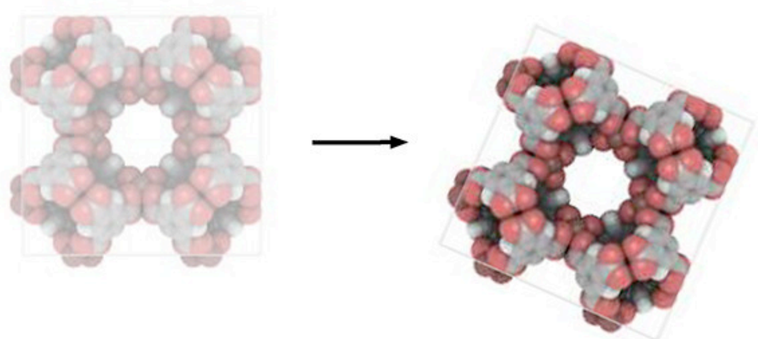
- differentiability
- invertibility



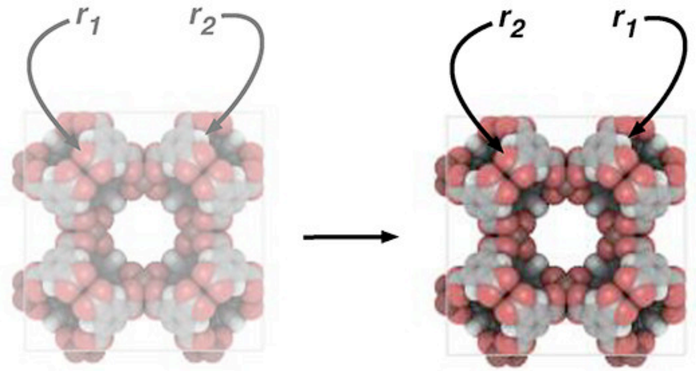
a Expansion to supercell.



b Translation.



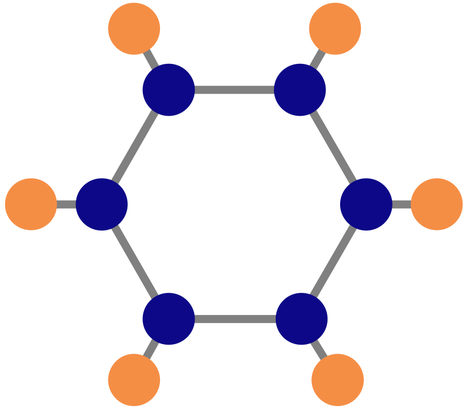
c Rotation.



c Permutation.

Chem. Rev. **2020**, *120* (16), 8066–8129.

Three ways to make model symmetry aware



H	-0.21463	0.97837	0.33136
C	-0.38325	0.66317	-0.70334
C	-1.57552	0.03829	-1.05450
H	-2.34514	-0.13834	-0.29630

Coordinates are sensitive to translations, rotations, and permutation.

Data Augmentation

Throw data at the problem and see what you get!

Invariant Inputs
Equivariant Inputs

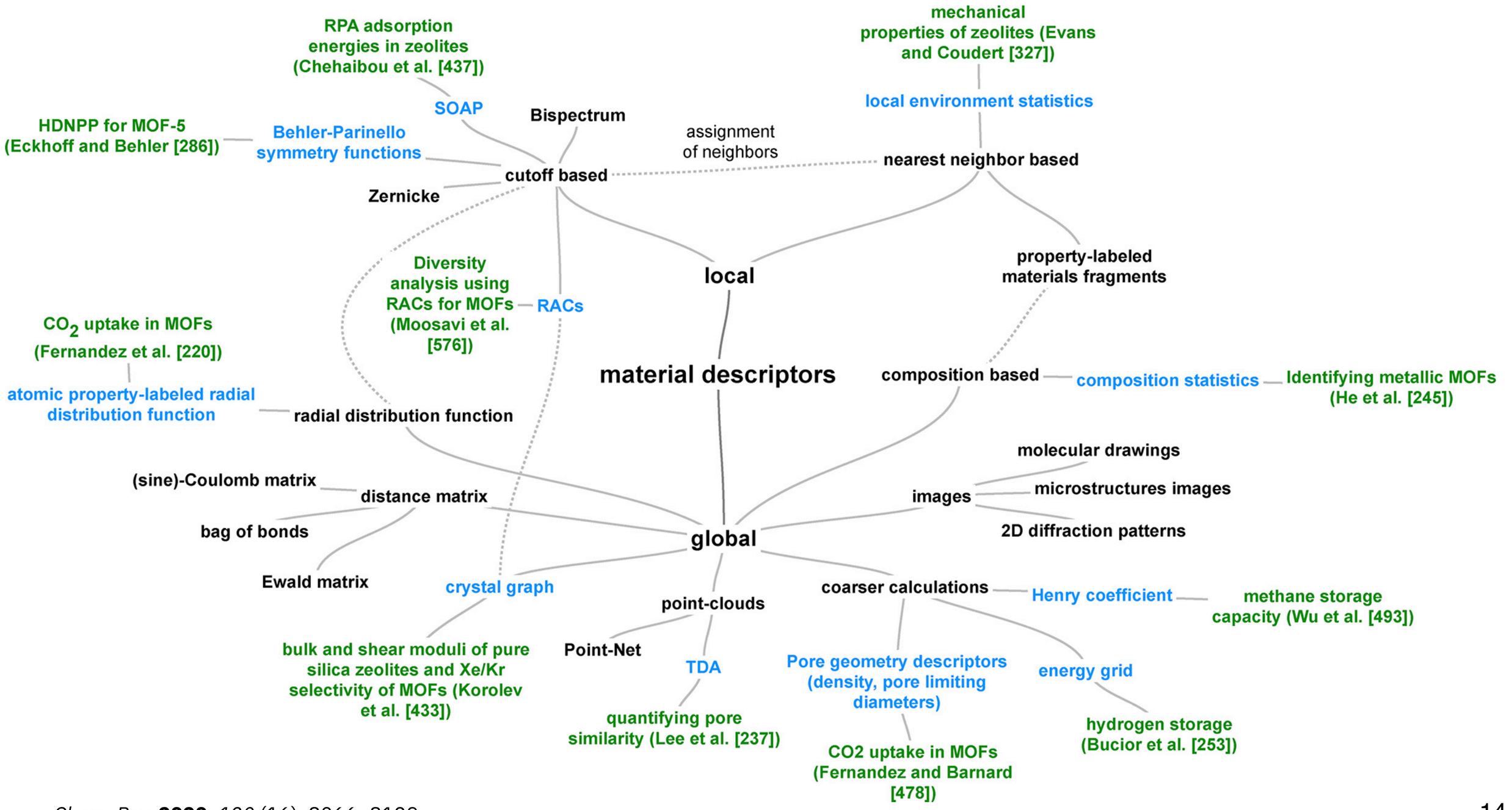
13

Convert your data to invariant representations so the model can't possibly mess it up.

Invariant models
Equivariant models

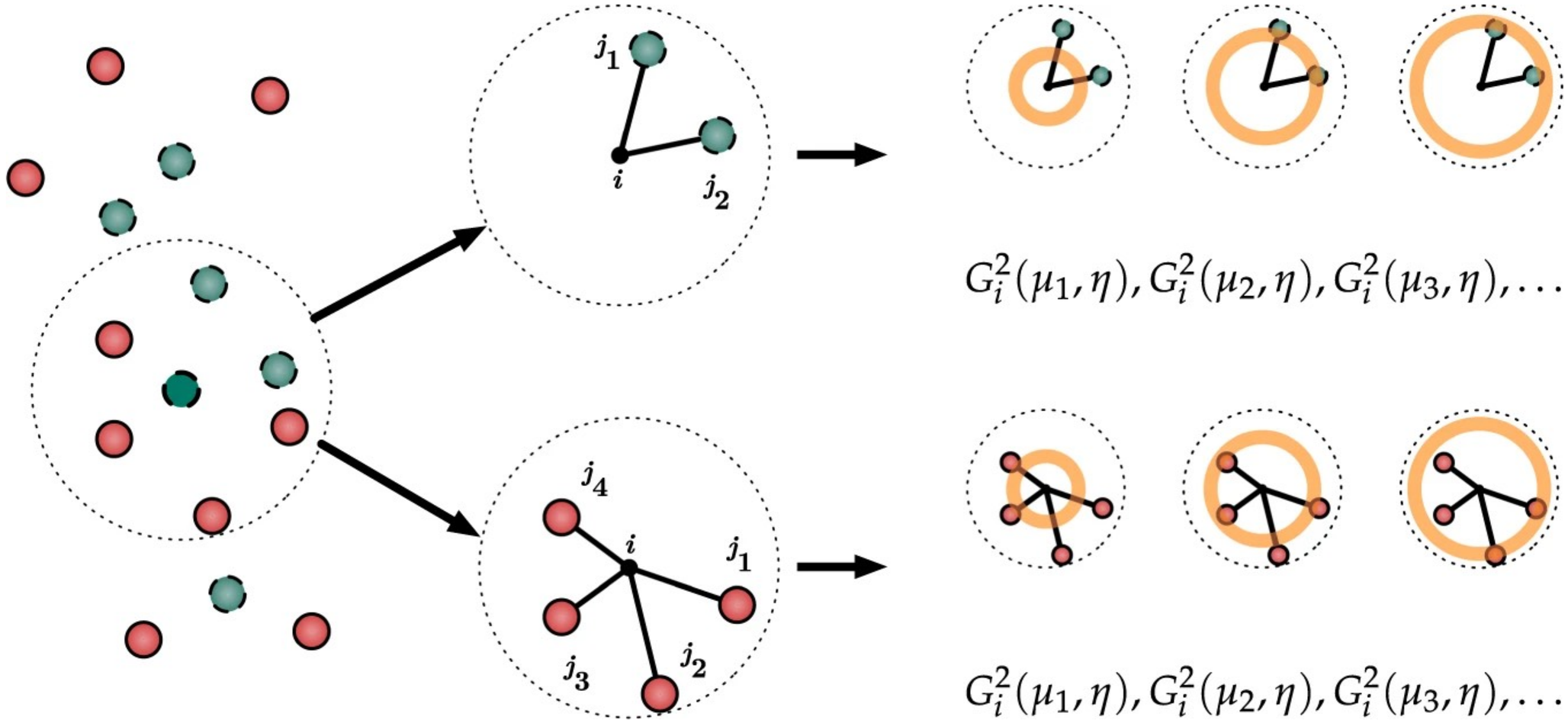
Make a model that can naturally handle coordinates as input (equivariant neural networks)

(Slide adapted from Tess Smidt)

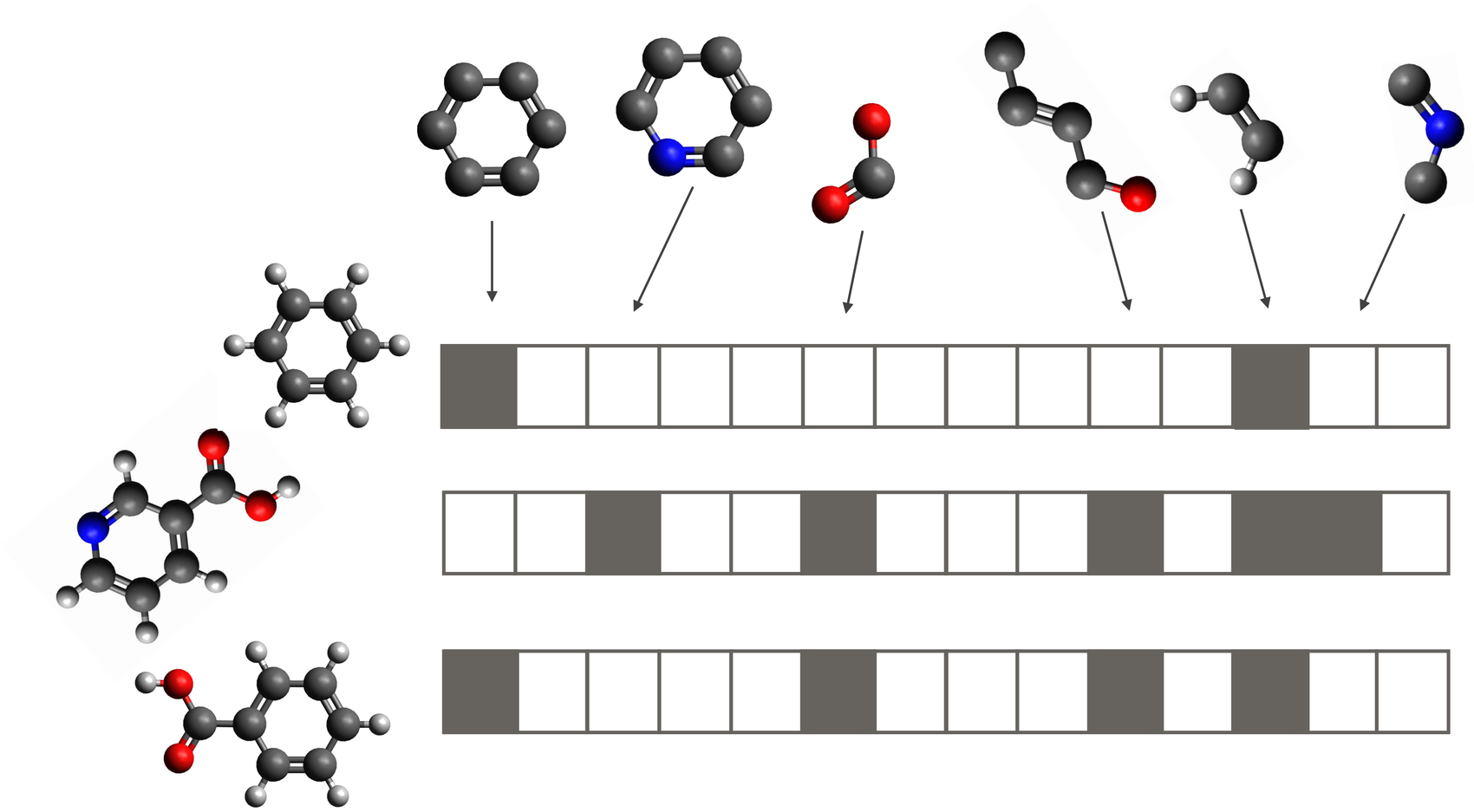


Chem. Rev. **2020**, *120* (16), 8066–8129.

Symmetry functions - Locality Approximation

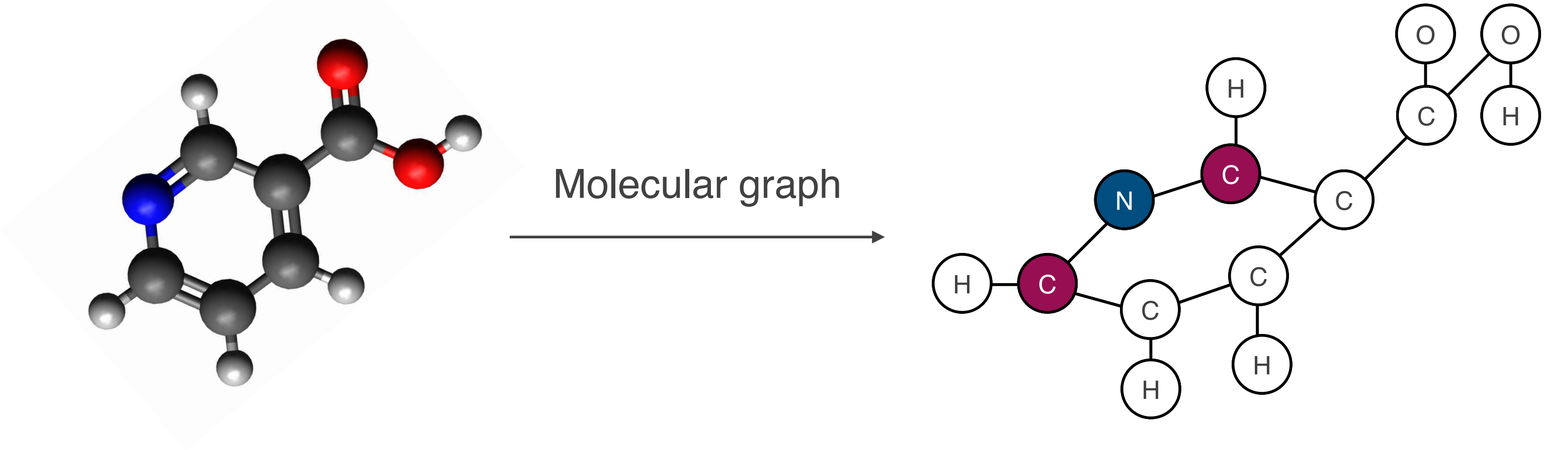


Fragment based fingerprints



Molecule graph summarization

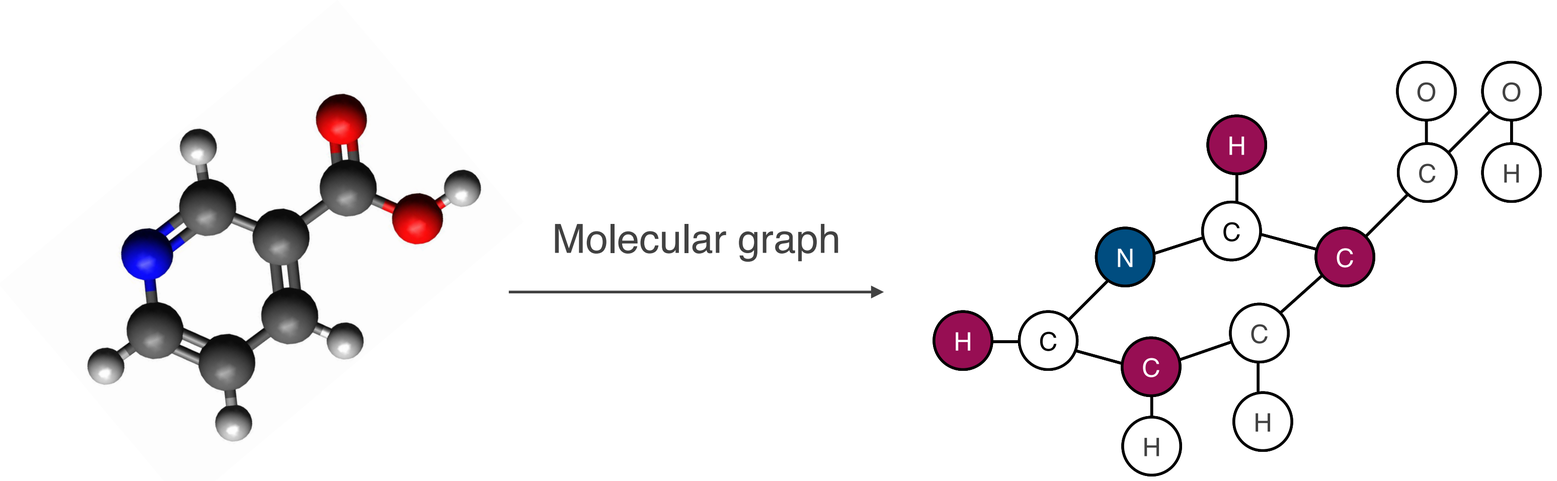
Create fix-length descriptor by capturing local correlations of molecule graph



$$\underset{all}{\overset{[N]}{\chi}}{}_1^{diff} = \sum_{i}^{[N]} \sum_{j}^{all} (\chi_i - \chi_j)\delta(d_{i,j},1)$$

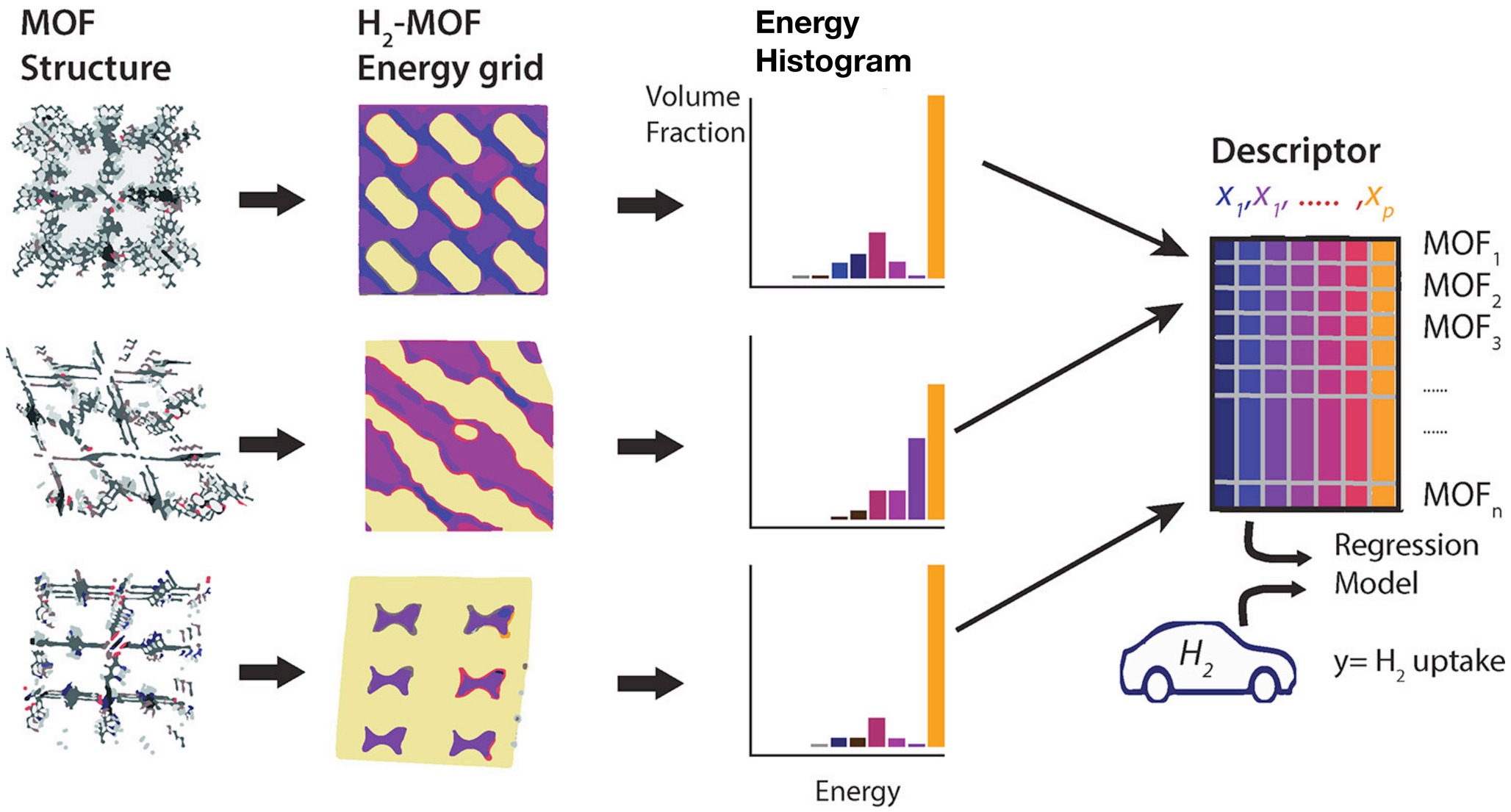
Molecule graph summarization

Create fix-length descriptor by capturing local correlations of molecule graph



$$ {}^{[N]}_{all}\chi_1^{diff} = \sum_{i}^{[N]} \sum_{j}^{all} (\chi_i - \chi_j)\delta(d_{i,j},2) $$

Cheaper surrogate



1. Feeding structures into models

Learning Objectives:

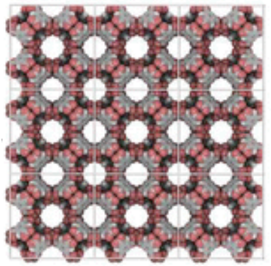
- Need for featurization
- Equivariances/invariances
- Typical assumptions
- Examples of descriptors

2. Training a basic model

Learning Objectives:

- Supervised ML workflow
- Empirical risk estimate
- Test data

Supervised ML workflow



$$X = \begin{pmatrix} x^{1,1} & x^{1,1} & \dots & x^{1,p} \\ x^{2,1} & x^{2,1} & \dots & x^{2,p} \\ \vdots & \ddots & & \vdots \\ x^{n,1} & x^{n,2} & \dots & x^{n,p} \end{pmatrix} \quad Y = \begin{pmatrix} y^1 \\ y^2 \\ \vdots \\ y^n \end{pmatrix}$$

p features

n observations

How do we measure progress towards our goal

Goal: Find good function *f* in function space $\mathcal{F}$

How: Minimize the expected risk of *f* $R(f) = E_Z[L(z, f)] = \int_Z L(z, f)p(z)\,\mathrm{d}z$

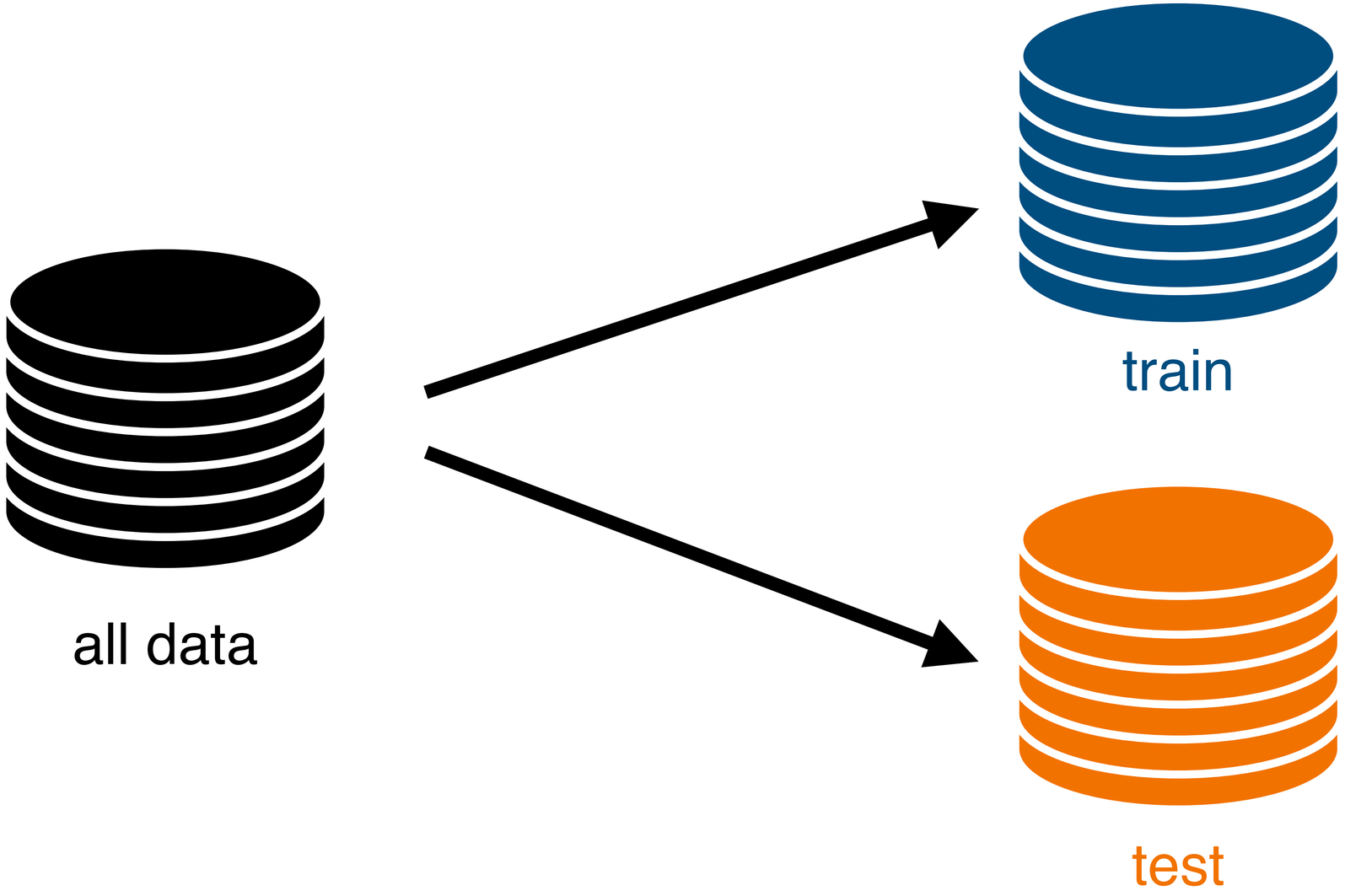
However *p(z) unknown!*

Fix: Empirical Risk $\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n} L(z_i, f)$

Given a finite amount of training data, derive a relation for an infinite domain!

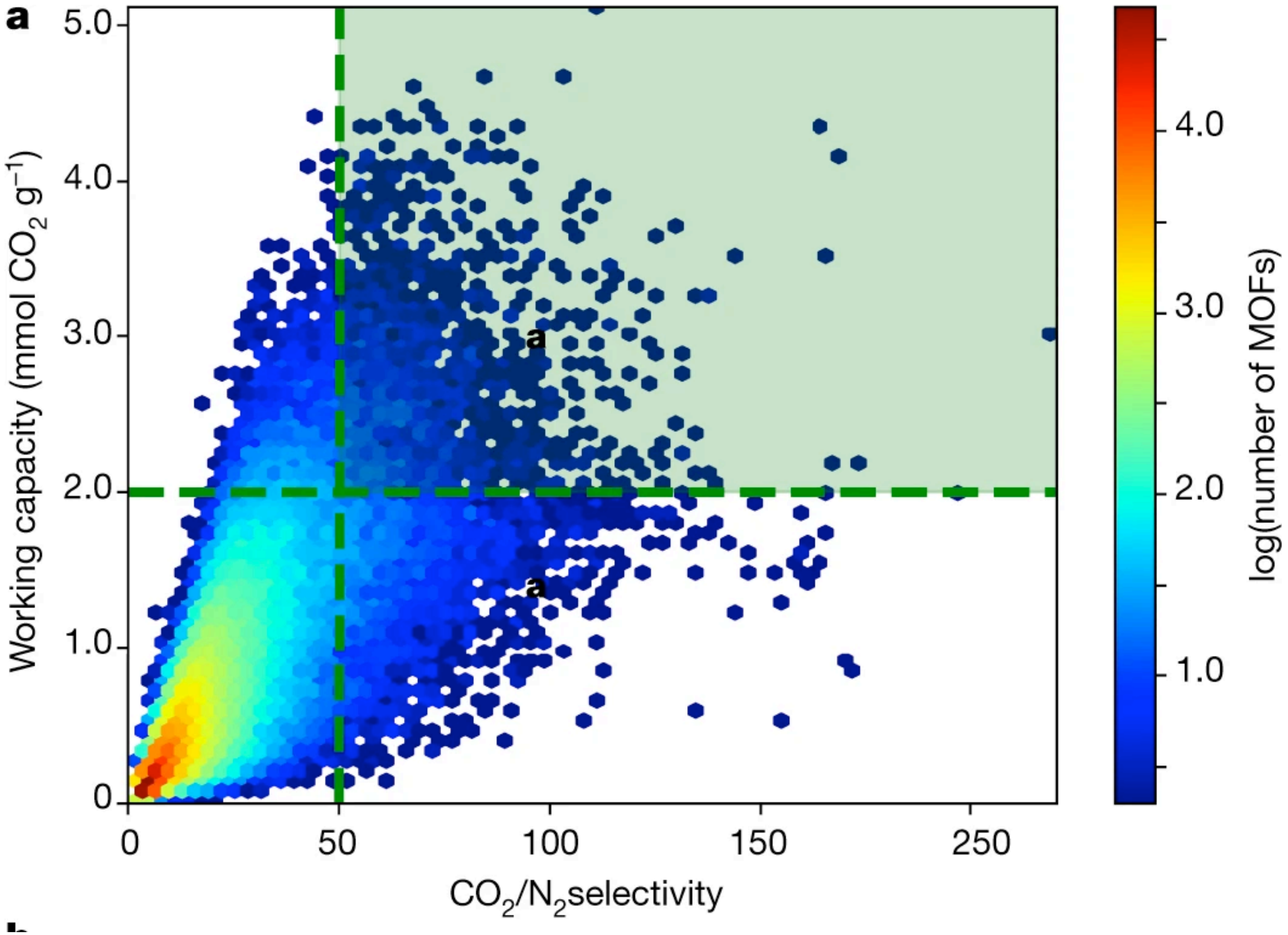
Selecting the datasets over which to compute the empirical risk is a key challenge!

First step: Put aside some test data you do not look at



First step: Put aside some test data you do not look at

A random sample from such a datasets might contain no or little top performing materials.



2. Training a basic model

Learning Objectives:

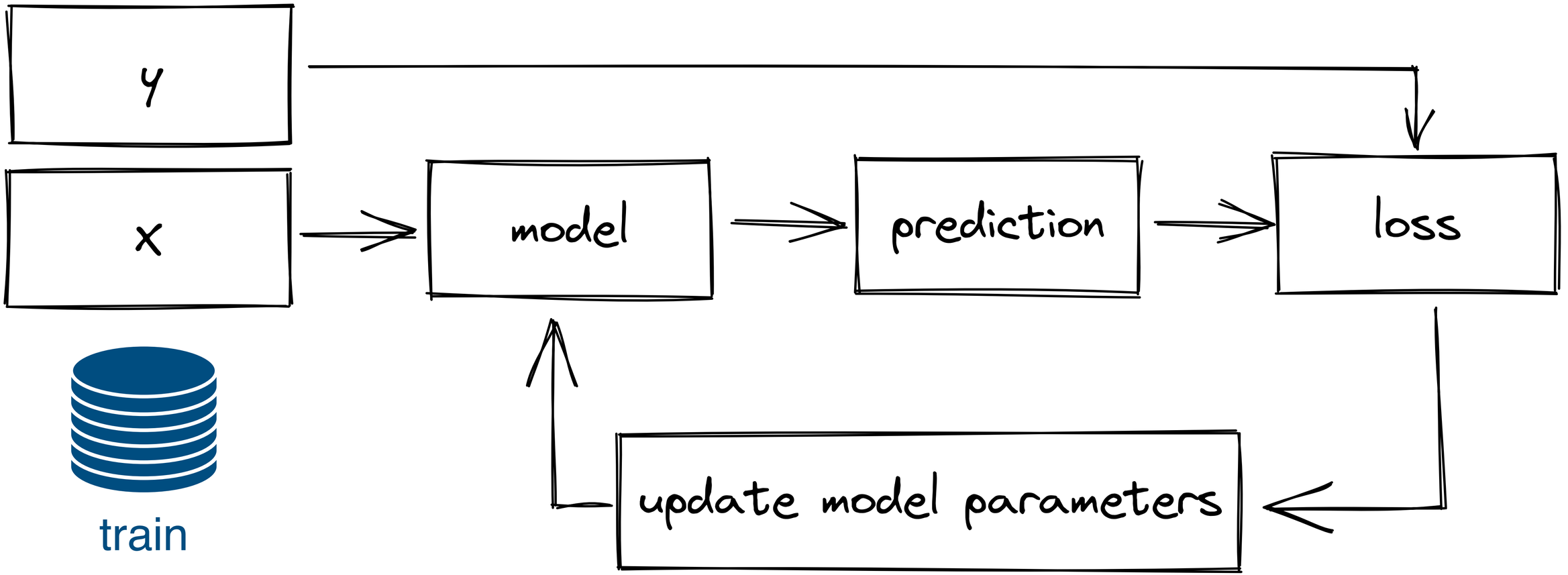
- Supervised ML workflow
- Empirical risk estimate
- Test data

3. Regression

Learning Objectives:

- Matrix formulation of linear regression
- Recasting non-linear regression in matrix notation

Choose trainable weights that minimize empirical risk



Simple linear regression

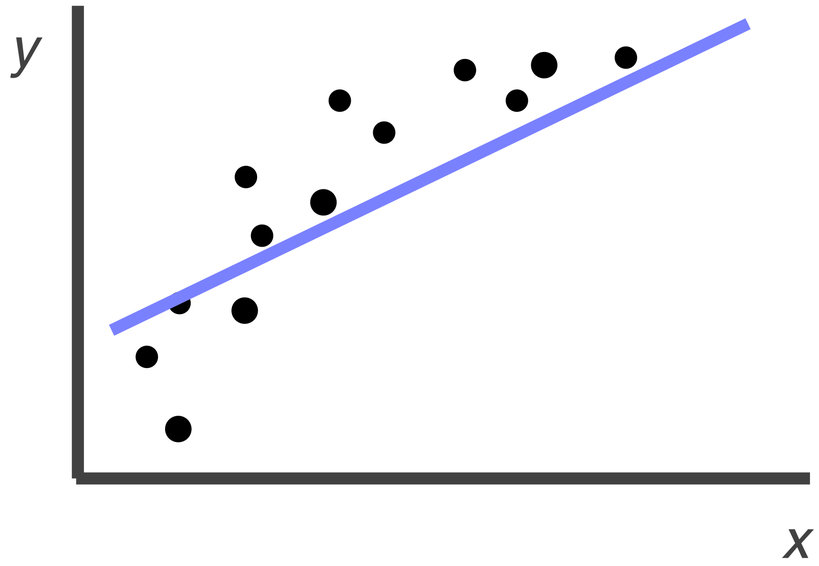
One feature

One point

$$f(x^*) \approx y^* = wx^* + b$$

alternative formulation

$$f(x^*) = x^*w = [1 \quad x^*] \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$



All points

$$f(\mathbf{X}) \approx \mathbf{y} = w\mathbf{X} + b$$

$$f(\mathbf{X}) = Xw = \begin{bmatrix} 1 & x^1 \\ \vdots & \vdots \\ 1 & x^m \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

```python
def linear_model(x, w, b):
    return np.dot(x, w) + b
```

Multiple linear regression

Multiple features

Single data point

$$f(\mathbf{X}^*) = \mathbf{X}^*\mathbf{w} = \begin{bmatrix} 1 & x_1^* & x_2^* & \dots & x_p^* \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{bmatrix}$$

All data points

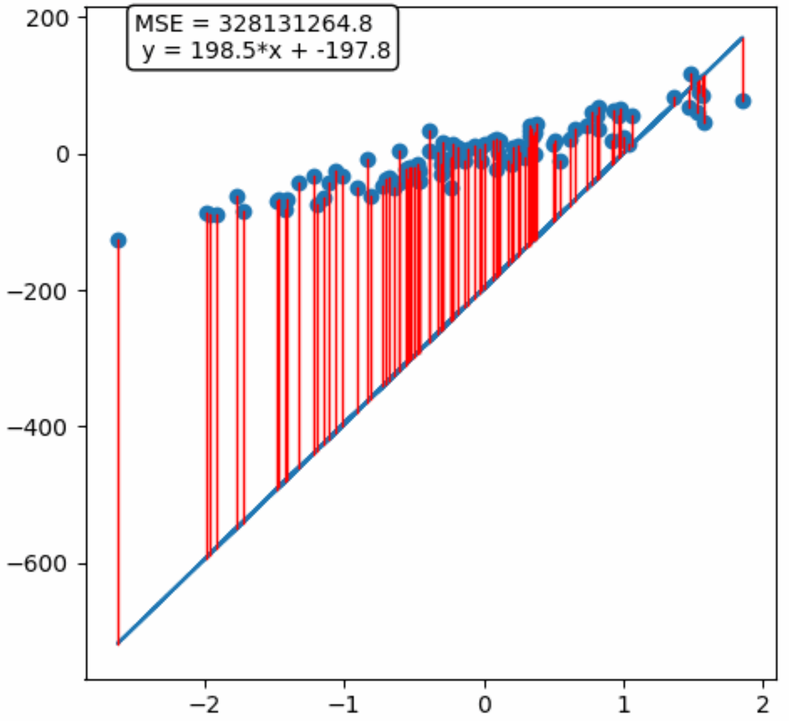
$$f(\mathbf{X}) = \mathbf{X}\mathbf{w} = \begin{bmatrix} 1 & x_1^1 & x_2^1 & \dots & x_p^1 \\ \vdots & \vdots & & \ddots & \\ 1 & x_1^n & x_2^n & \dots & x_p^n \end{bmatrix} \begin{bmatrix} w_0 \\ \vdots \\ w_p \end{bmatrix}$$

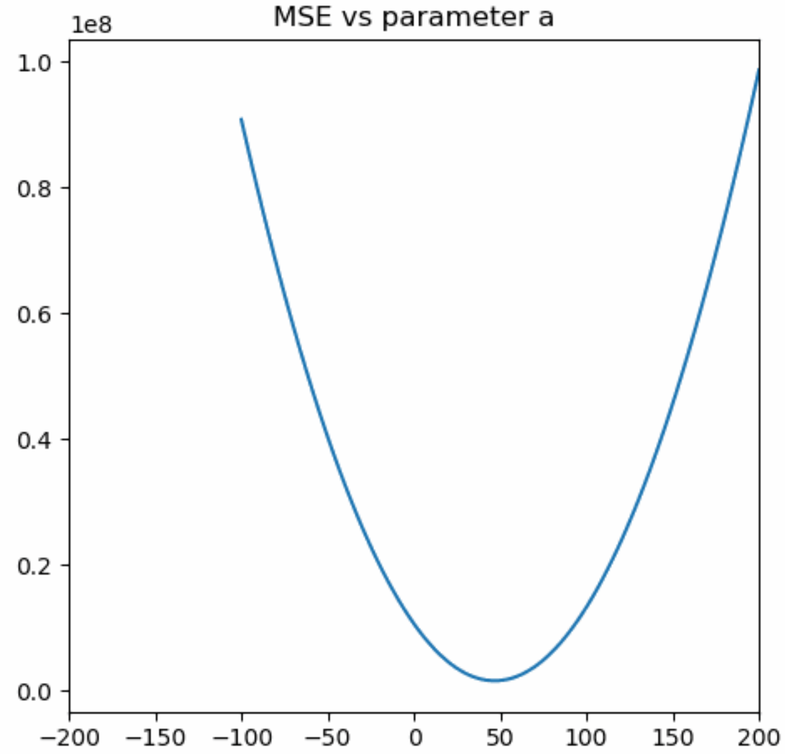
Coding example - Implementing Linear Regression

How do we find the adjustable parameters?

For a *all* models: Numerical optimization

Step 2: Go downhill





How do we find the adjustable parameters?

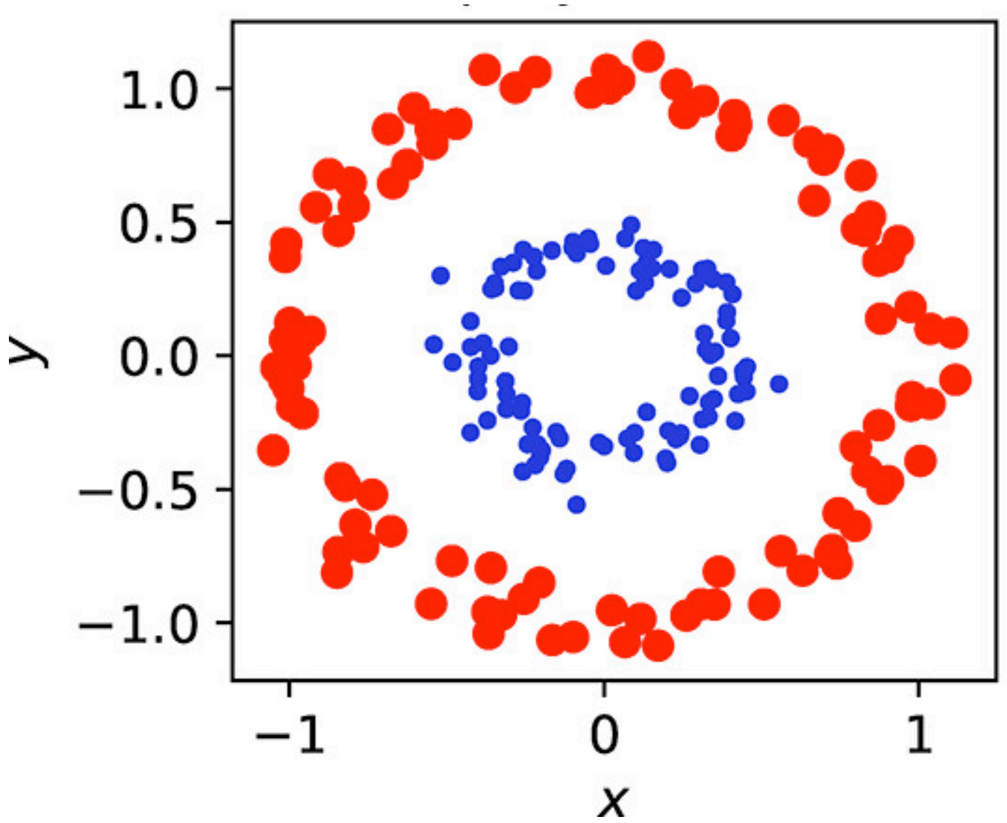
For a *all* models: Numerical optimization

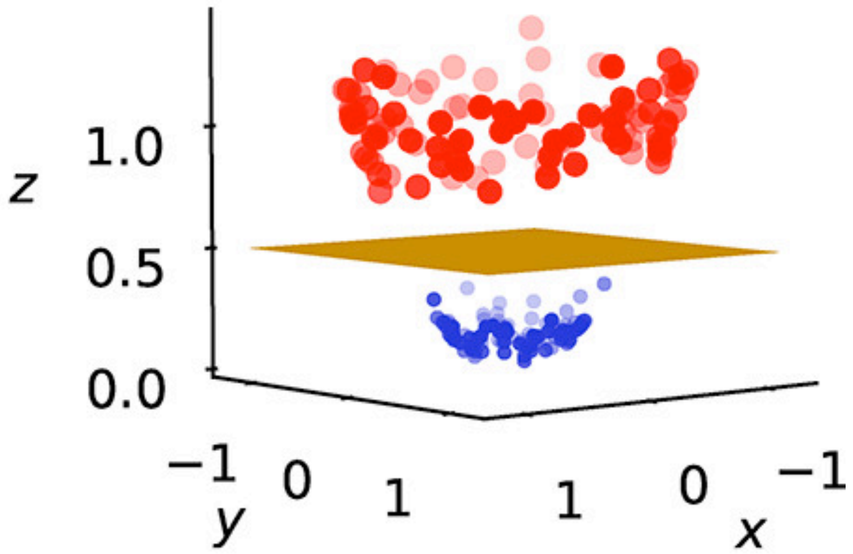
Step 2: Go downhill

Modern frameworks (e.g., jax) can automatically compute gradients!

```
def loss_wrapper(w, b, data):
    features = data[0]
    labels = data[1]
    y = linear_model(features, w, b)
    return loss(y, labels)


loss_grad = jax.grad(loss_wrapper, (0, 1))

data = (features, labels)

for i in range(10):
    grad = loss_grad(w, b, data)
    w -= eta * grad[0]
    b -= eta * grad[1]
```

How do we classify this data with a linear model?





By computing non-linear features!

$$[x_1^2 \quad x_2^2 \quad \sqrt{2}x_1x_2]$$

3. Regression

Learning Objectives:

- Matrix formulation of linear regression
- Recasting non-linear regression in matrix notation
- Solve non-linear problems by computing new features

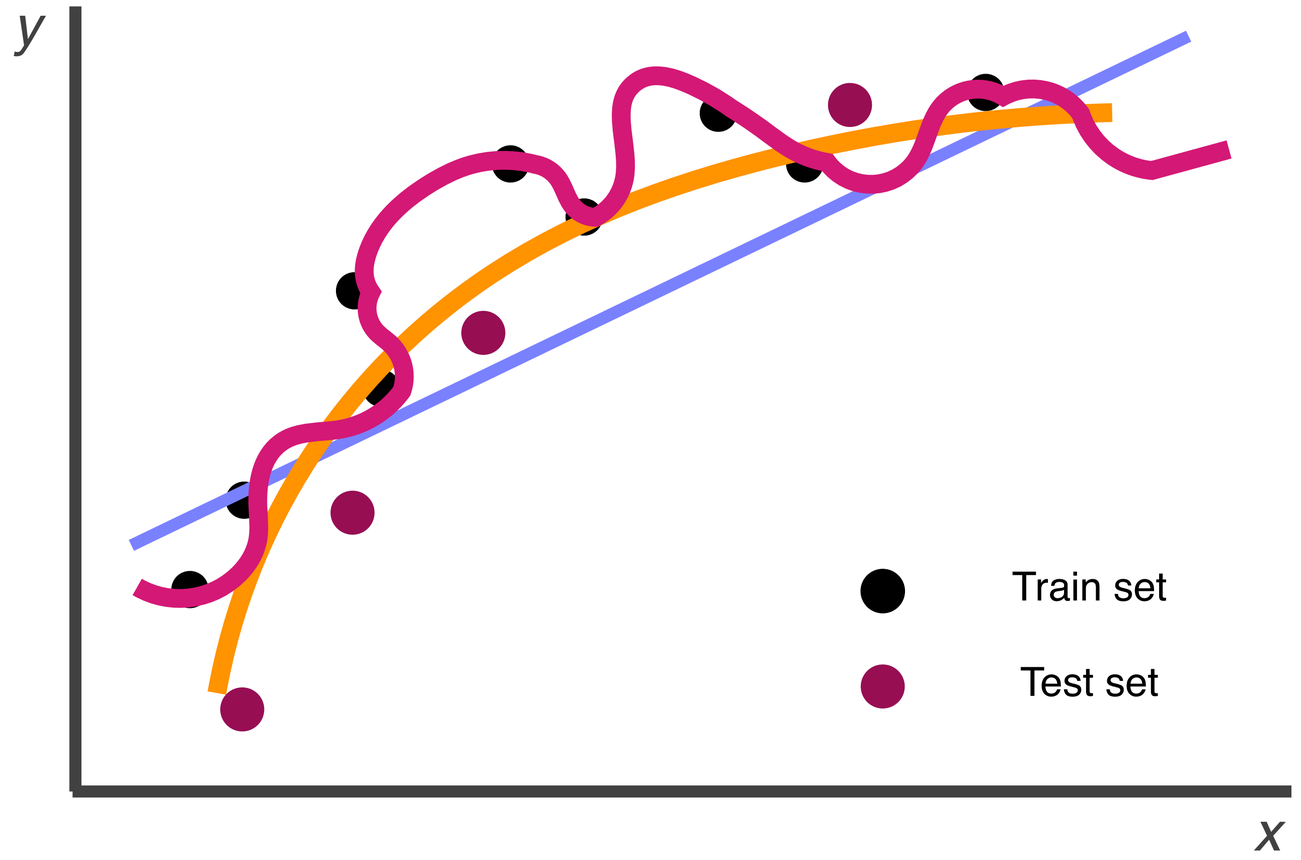
4. Bias-variance tradeoff

Learning Objectives:

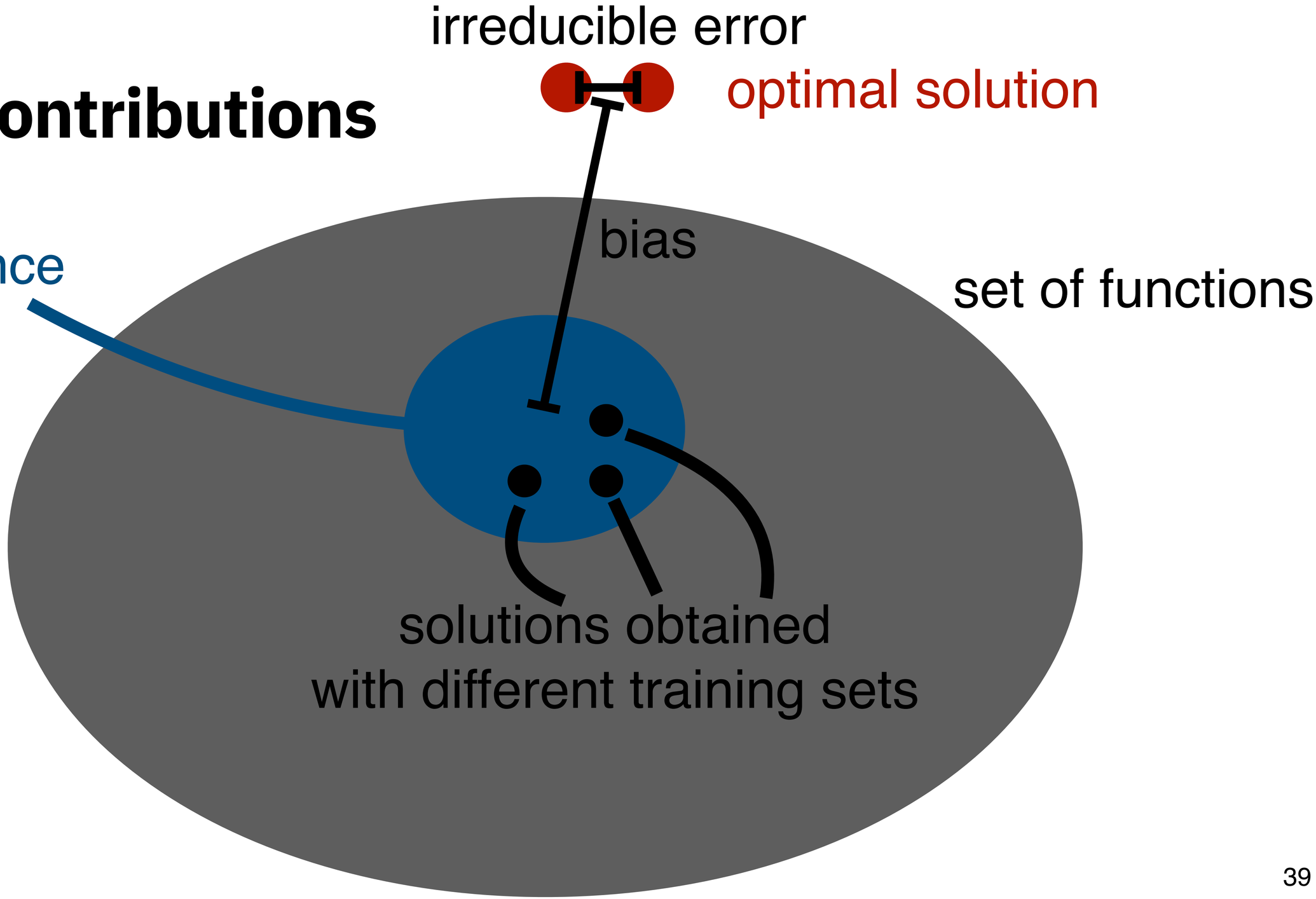
- We can split the error into different contributions: bias, variance, irreducible
- We cannot reduce bias and variance at the same time

Coding example - Fitting a LJ potential

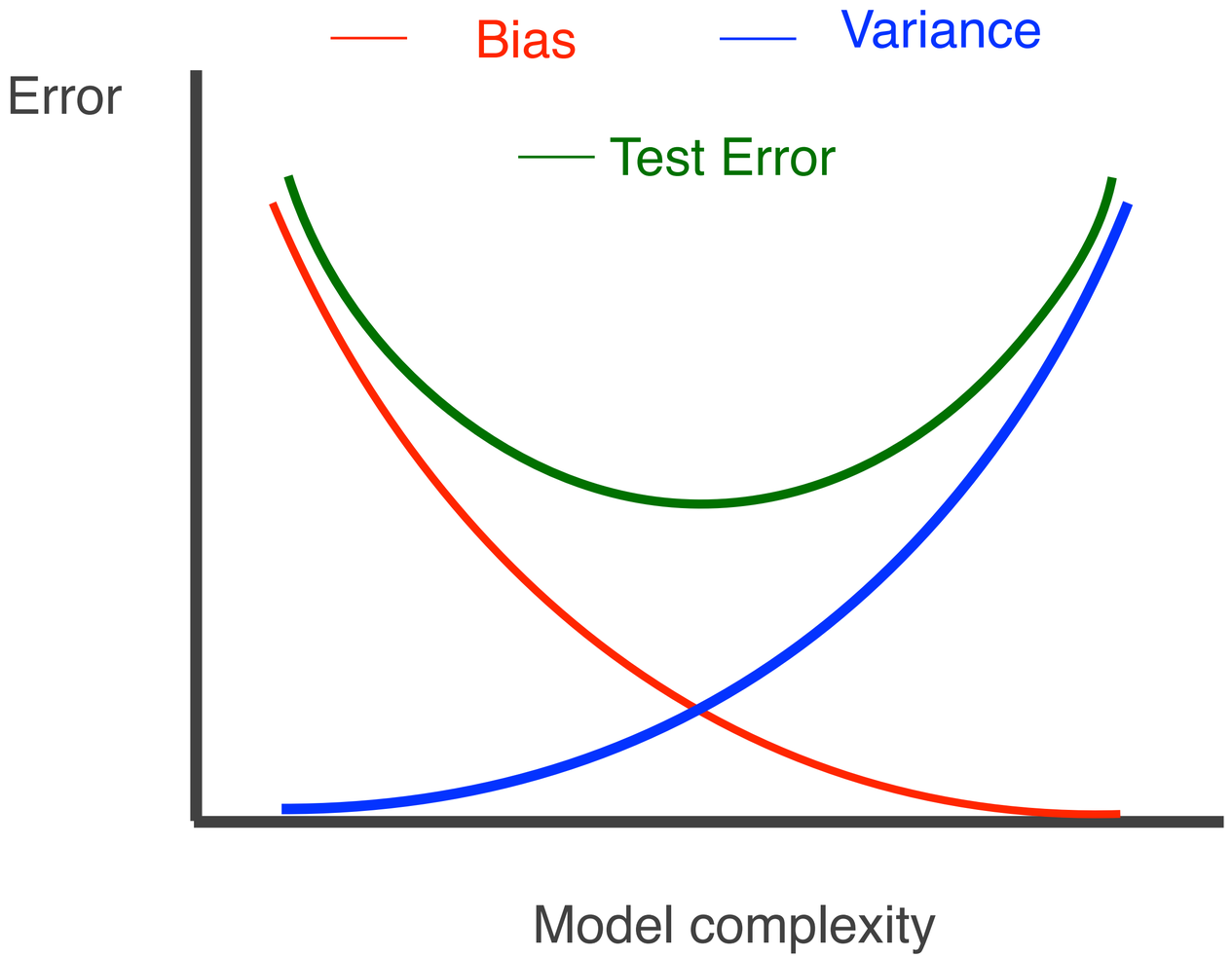
Overfitting vs. Underfitting



Error contributions

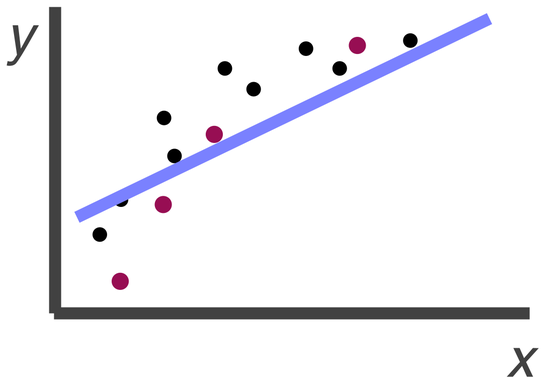


How do these error terms depend on the model complexity?



Regularization

Penalize functions
with large parameters

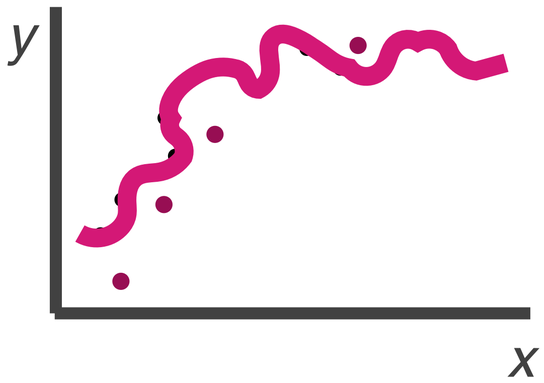


high bias

$w_0 + w_1 x$

“just right”

$w_0 + w_1 x + w_2 x^2$



high variance

$w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$

$$\mathscr{L} = \|\hat{y}_{ML} - y\|_2^2 + \lambda \|w\|_2^2$$

4. Bias-variance tradeoff

Learning Objectives:

- We can split the error into different contributions: bias, variance, irreducible
- We cannot reduce bias and variance at the same time

5. Hyperparameters

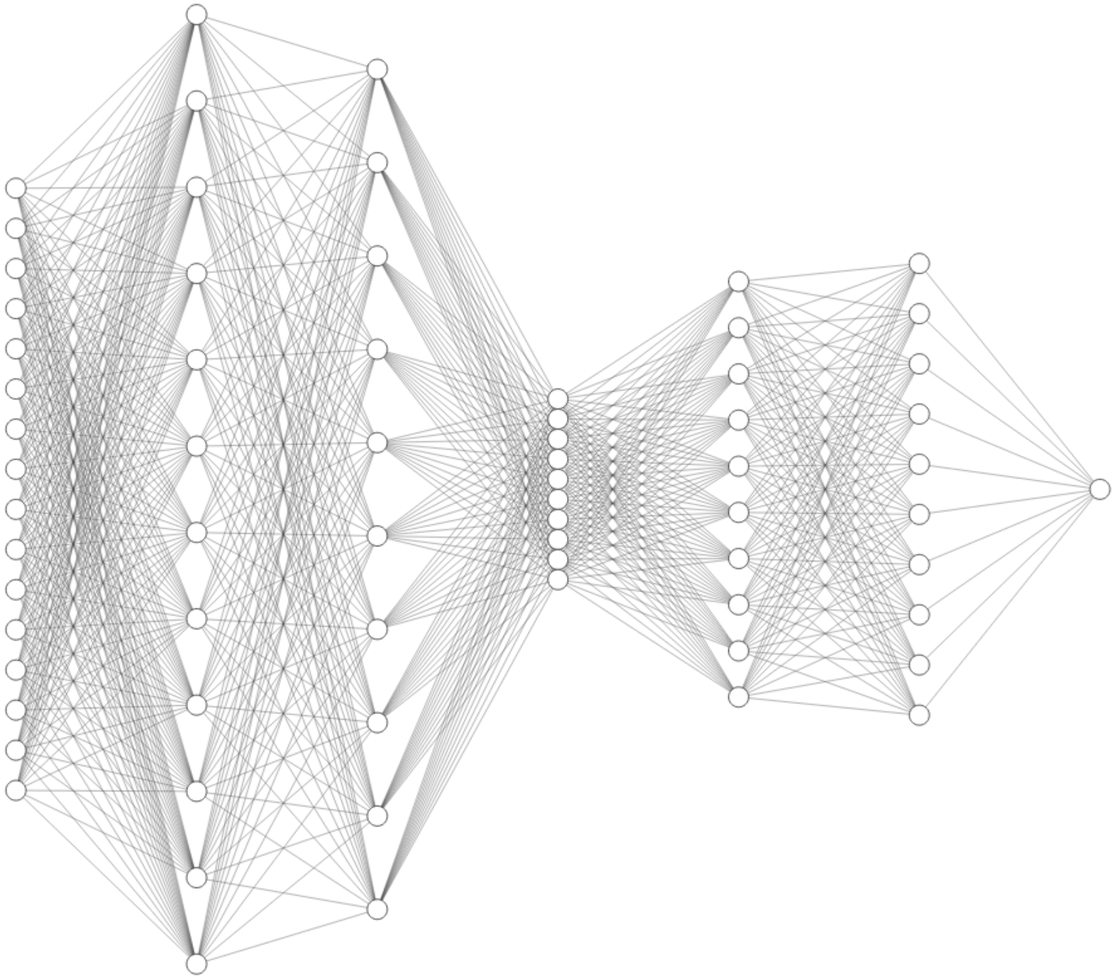
Learning Objectives:

- Hyperparameter are non-trainable parameters that need to be tuned
- (Nested) Cross-validation can be used to optimize them
- K-fold cross-validation

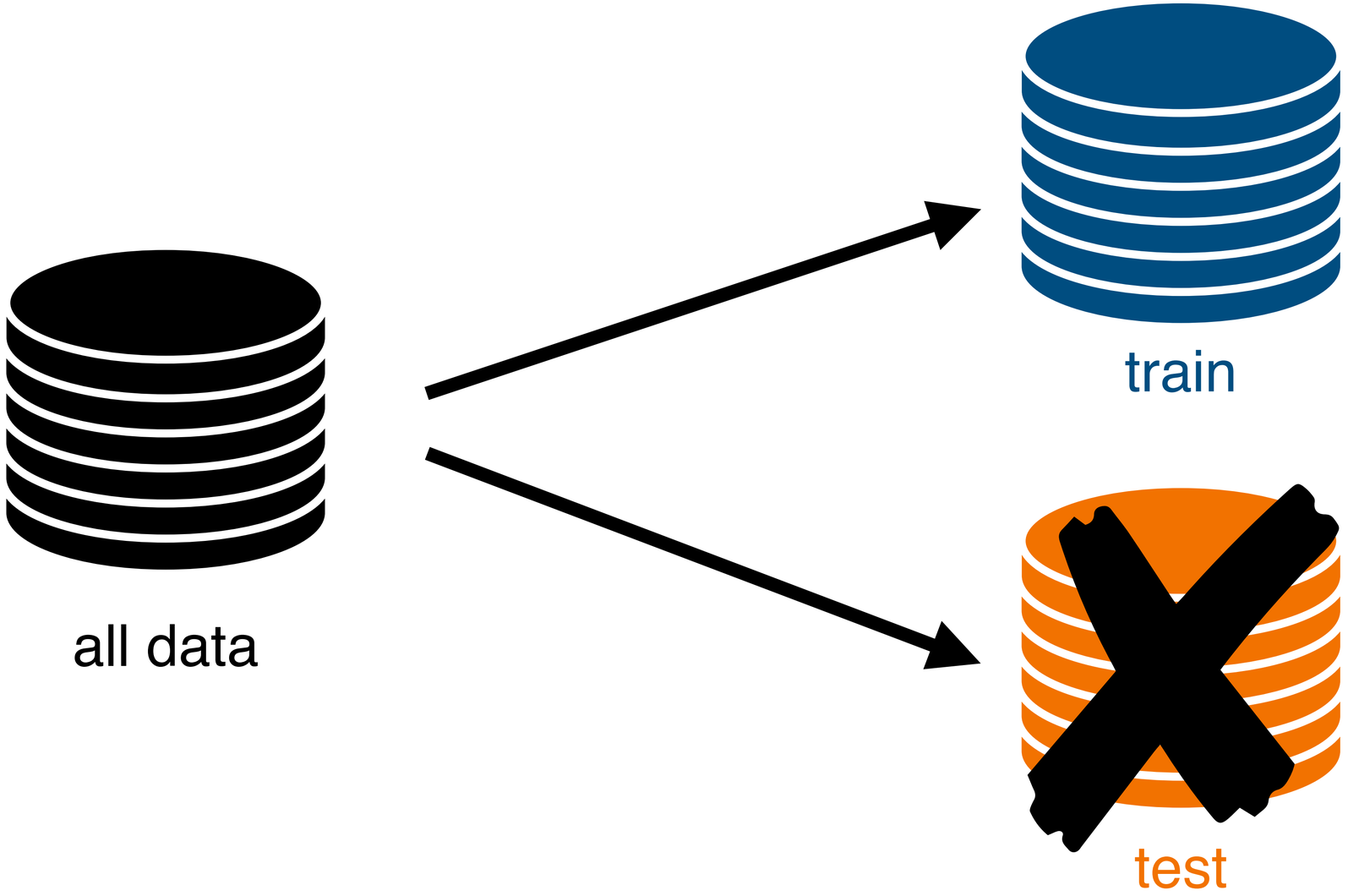
Hyperparameters

We cannot learn them,
but we need to optimize
them

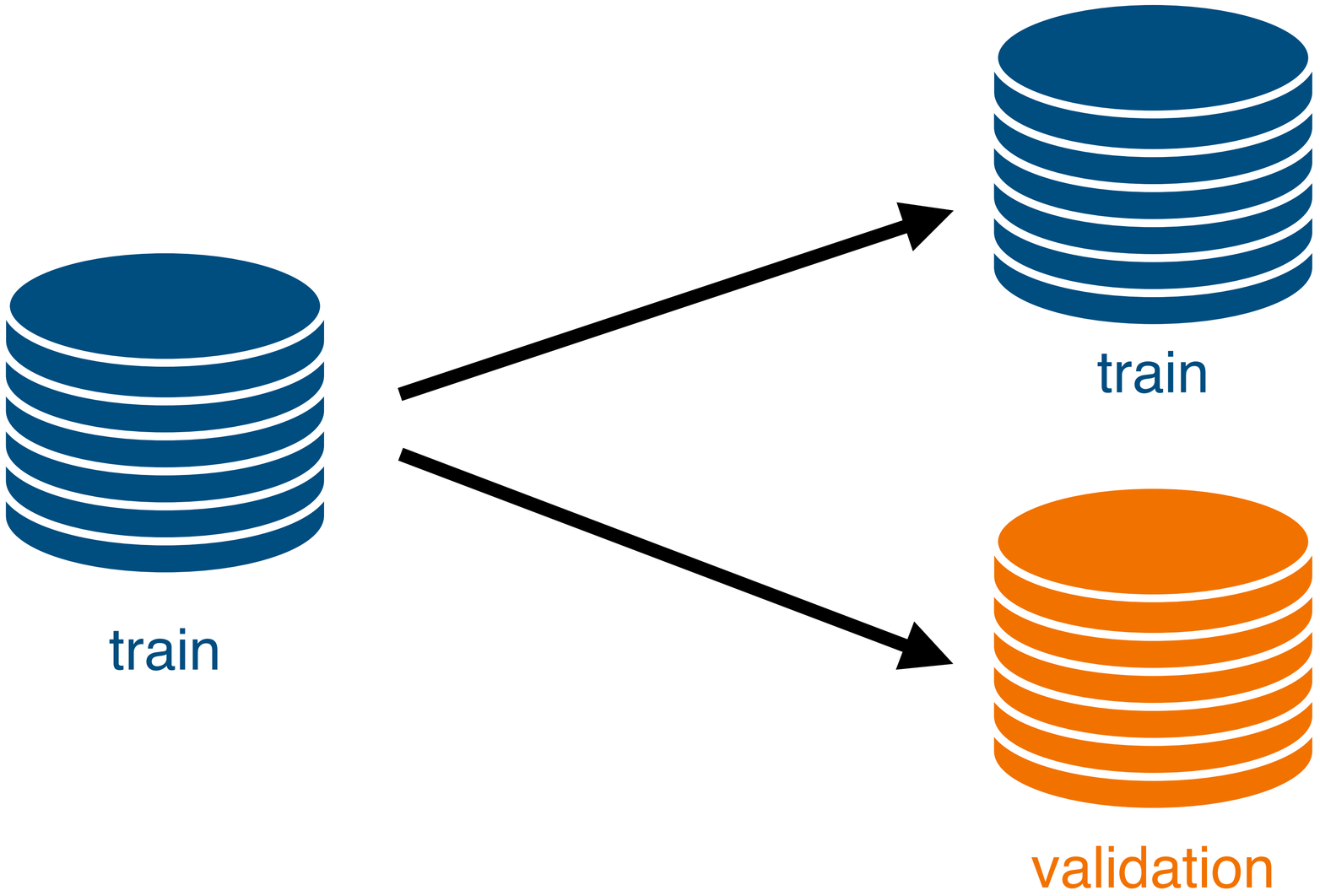
$$\mathscr{L} = \|\hat{y}_{ML} - y\|_2^2 + \lambda \|w\|_2^2$$



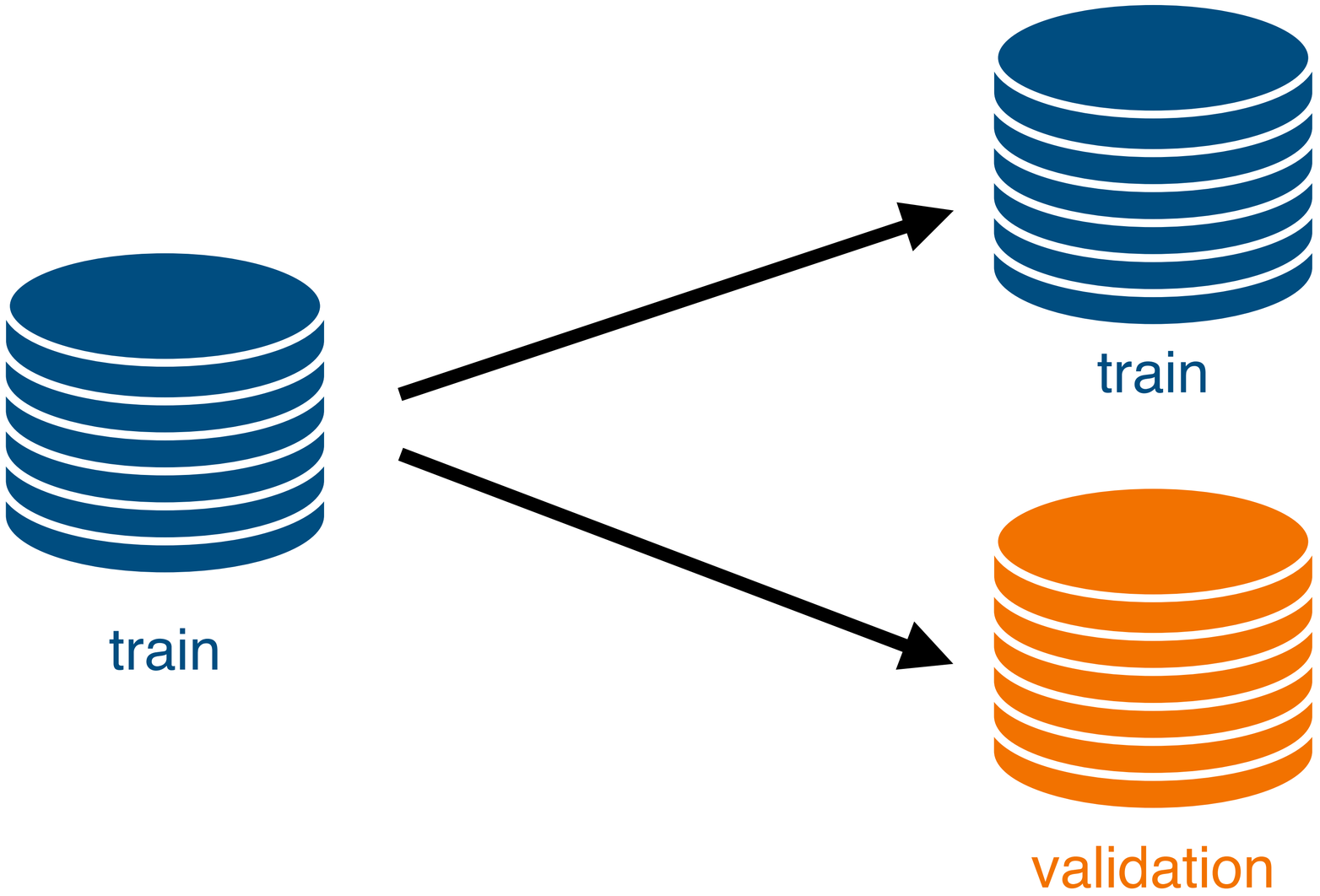
Recall: First step: Put aside some test data *you do not look at*



Hyperparameter tuning on *validation set*

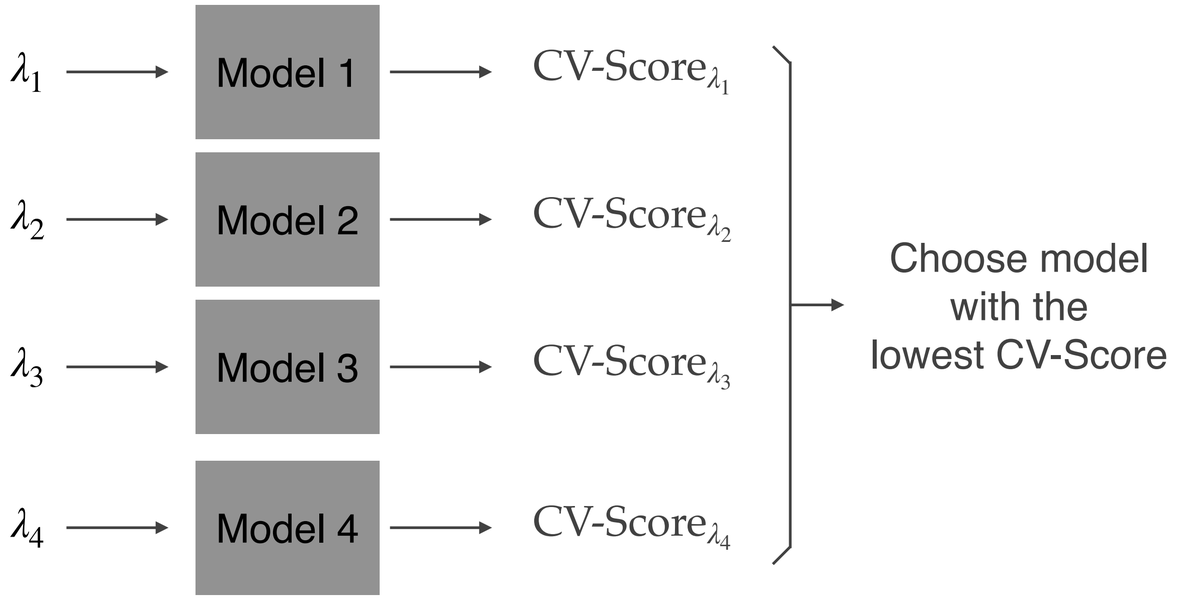


Hyperparameter tuning on *validation set*



Using cross-validation for hyperparameter tuning

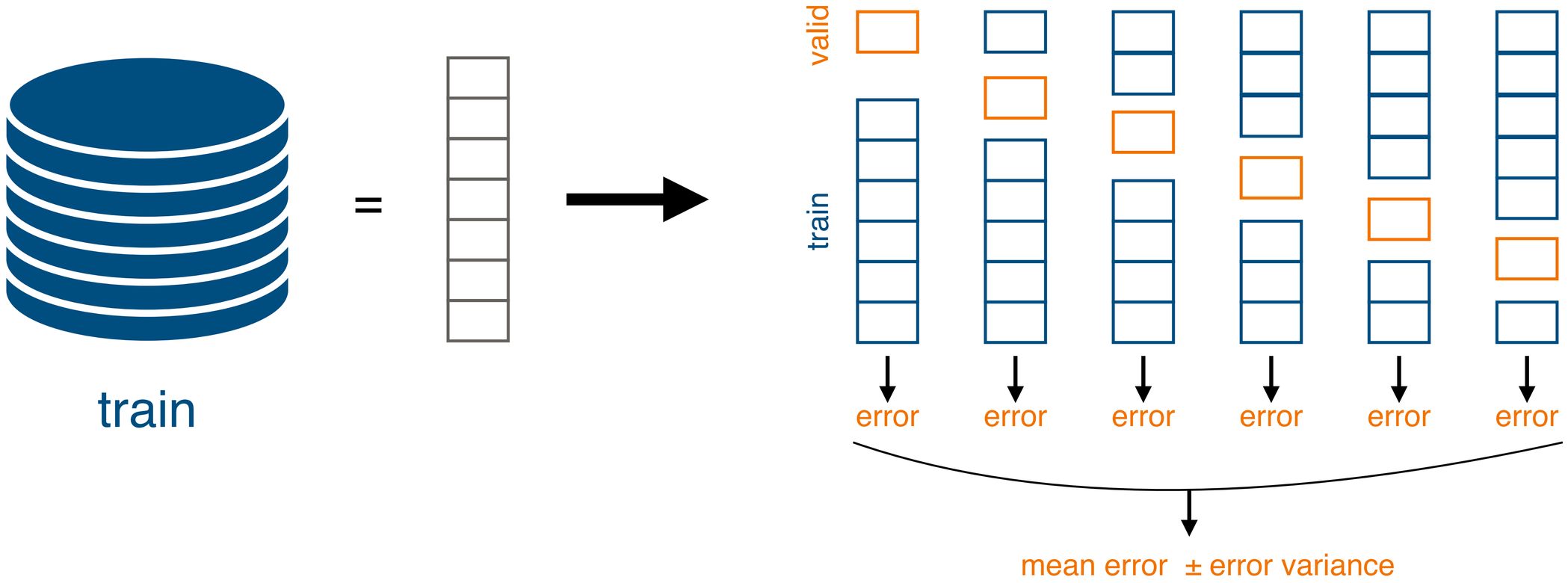
- We test different hyperparameter settings
- Estimate the error rate using cross validation
- Then pick the one with the lowest error
- And only then (when we are done) get the error rate on the test set

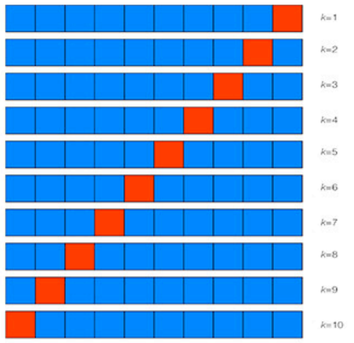


For small datasets: Build validation set using *k*-fold crossvalidation

Problem with simple split: Pessimistic bias and variance of the error estimate

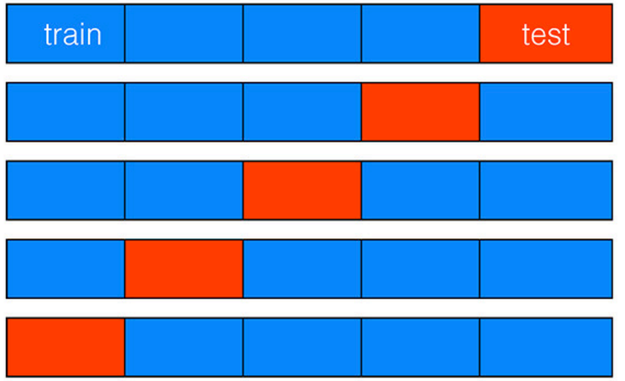
Doing split only once: High variance
Larger test set reduces the variance, but leads to underestimation (pessimistic bias)

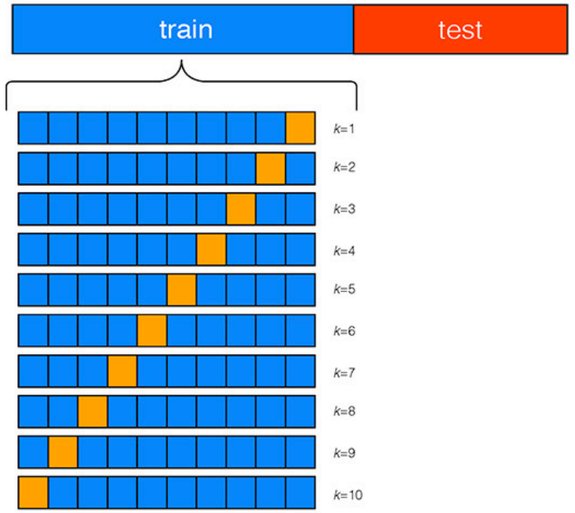




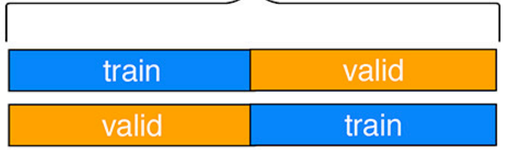
k-fold cross-validation and bootstrapped confidence interval

two-fold holdout split and normal approximation for confidence interval





inner k-fold crossvalidation for hyperparameter tuning



inner k-fold crossvalidation for hyperparameter tuning

little data — big data

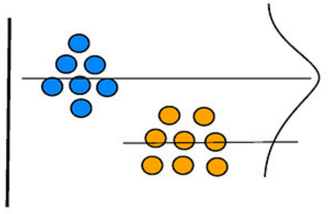
performance estimation

measuring and comparing model performance

model comparison

little data — nested cross validation

big data — effect size (Crochan's Q)



k-fold cross-validation with holdout set — little data

model selection (hyperparameter optimization)

three-fold holdout split — big data

5. Hyperparameters

Learning Objectives:

- Hyperparameter are non-trainable parameters that need to be tuned
- (Nested) Cross-validation can be used to optimize them
- K-fold cross-validation

6. Kernel learning

Learning Objectives:

- The kernel trick allows us to use the training data as basis
- We do not have to pre-compute non-linearities

Features as basis set

(Non) Linear regression
primal picture

$$f(x) = \sum_i^N w_i b_i(x)$$

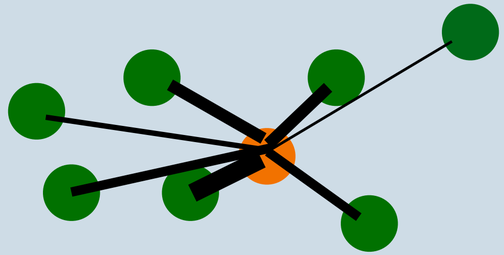
$b_i(x)$ basis functions

uptake $= w_1 \cdot$ density
$+ w_2 \cdot$ pore diameter
$- w_3$ maximum charge^2

Training data as basis set

Kernel regression
dual picture

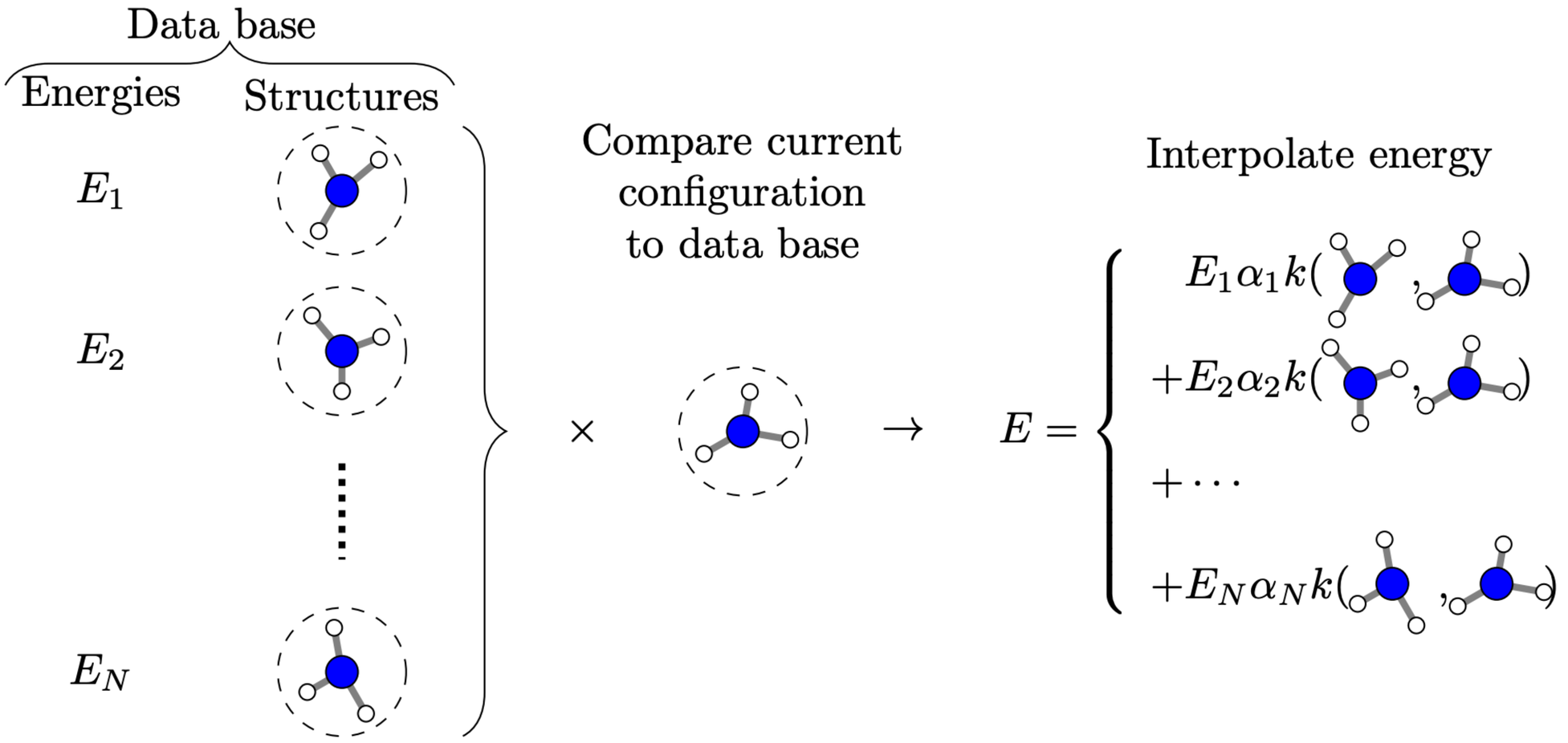
$$f(x) = \sum_i^N w_i K(x, x_i)$$



train example

query

Consider similarity to *all* training instances



M. A. Caro, *Arkhimedes* **2018**, 3, 21.

How do we get there?

Regularized linear regression

$$\mathbf{w} = \arg\min_{\mathbf{w}} \mathscr{L} = \arg\min_{\mathbf{w}} \left(\|\hat{\mathbf{y}}_{ML} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 \right)$$

Recall normal equation

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

Rewrite

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

Define

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

Using this definition

$$w = X^T a$$

Rewrite inference equation

$$\hat{y}_{ML}(x^*) = x^* w = \begin{bmatrix} x_1^* & \cdots & x_p^* \end{bmatrix} X^T a$$

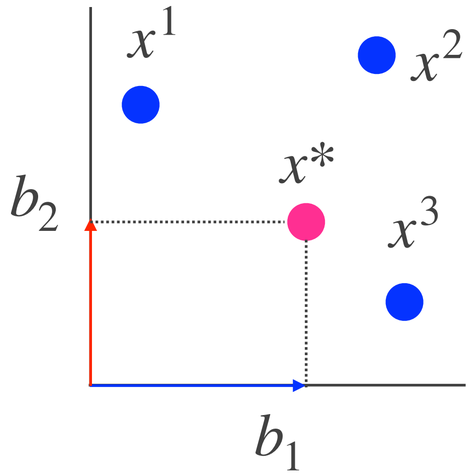
$$= \begin{bmatrix} x_1^* & \cdots & x_p^* \end{bmatrix} \begin{bmatrix} x_1^1 & \cdots & x_n^1 \\ \vdots & \ddots & \vdots \\ x_1^p & \cdots & x_n^p \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}$$

$$= \sum_{i=1}^{n} x^* x_i^T a_i = \sum_{i=1}^{n} K(x^*, x_i)\, a_i$$

Contrasting Feature and Data Basis

$$\hat{y}_{ML}(x^*) = x^* w = \begin{bmatrix} x_1^* & x_2^* \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = x_1^* w_1 + x_1^* w_2$$

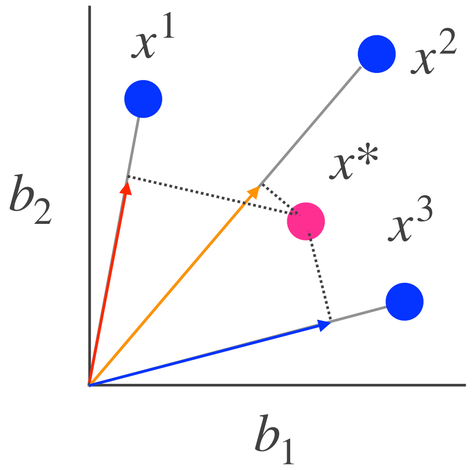
$$= \begin{bmatrix} x_1^* & x_2^* \end{bmatrix} \begin{bmatrix} b_1 & 0 \\ 0 & b_2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$



We describe the query point using the features

$$\hat{y}_{ML}(x^*) = \begin{bmatrix} 1 & x_1^* & \dots & x_d^* \end{bmatrix} X^T a$$

$$= \begin{bmatrix} x_1^* & x_2^* \end{bmatrix} \begin{bmatrix} x_1^1 & x_1^2 & x_1^3 \\ x_2^1 & x_2^2 & x_1^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_3 \end{bmatrix}$$



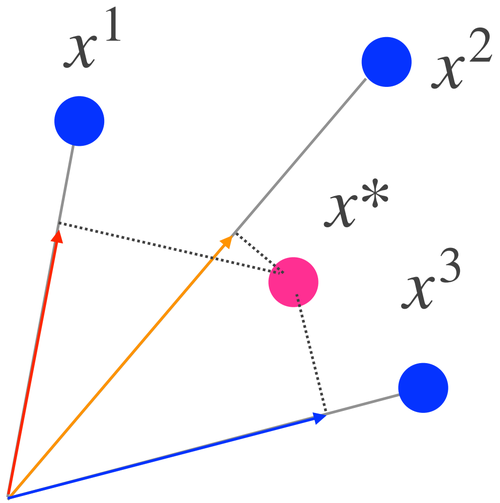
We describe the query point using the other points

Examples

Kernel is a notion of similarity, i.e. distance in feature space: $\hat{y}_{ML}(x^*) = \sum_{i=1}^{n} a_i K(x^*, x^i)$

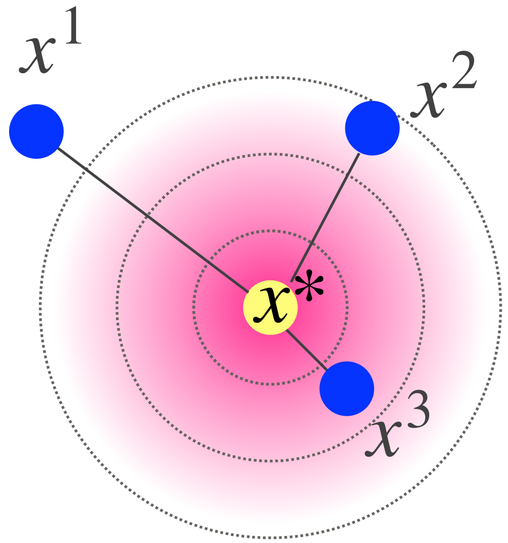
Linear basis

$K(x^*, x_i) = \langle x^*, x_i \rangle$

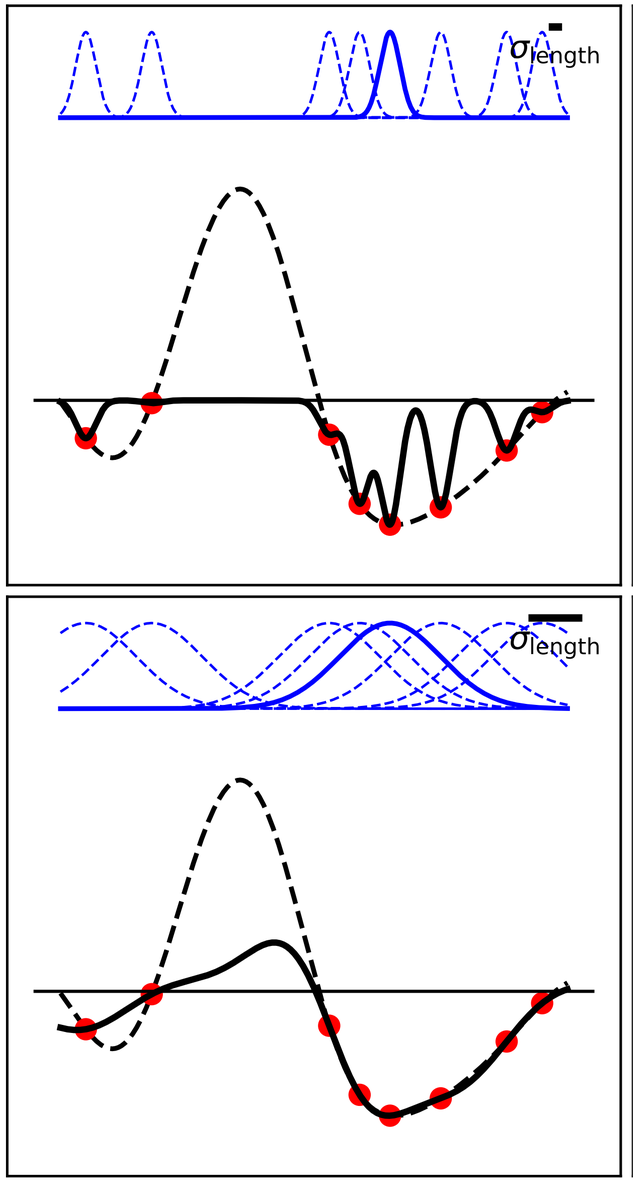


Radial basis

$K(x^*, x_i) = \exp\left(-\gamma \|x^* - x_i\|_2^2\right)$



Learning from function observations



6. Kernel learning

Learning Objectives:

- The kernel trick allows us to use the training data as basis
- We do not have to pre-compute non-linearities

7. Feature importance

Learning Objectives:
- Motivation for explainable ML
- Permutation feature importance

Motivation

Debugging models

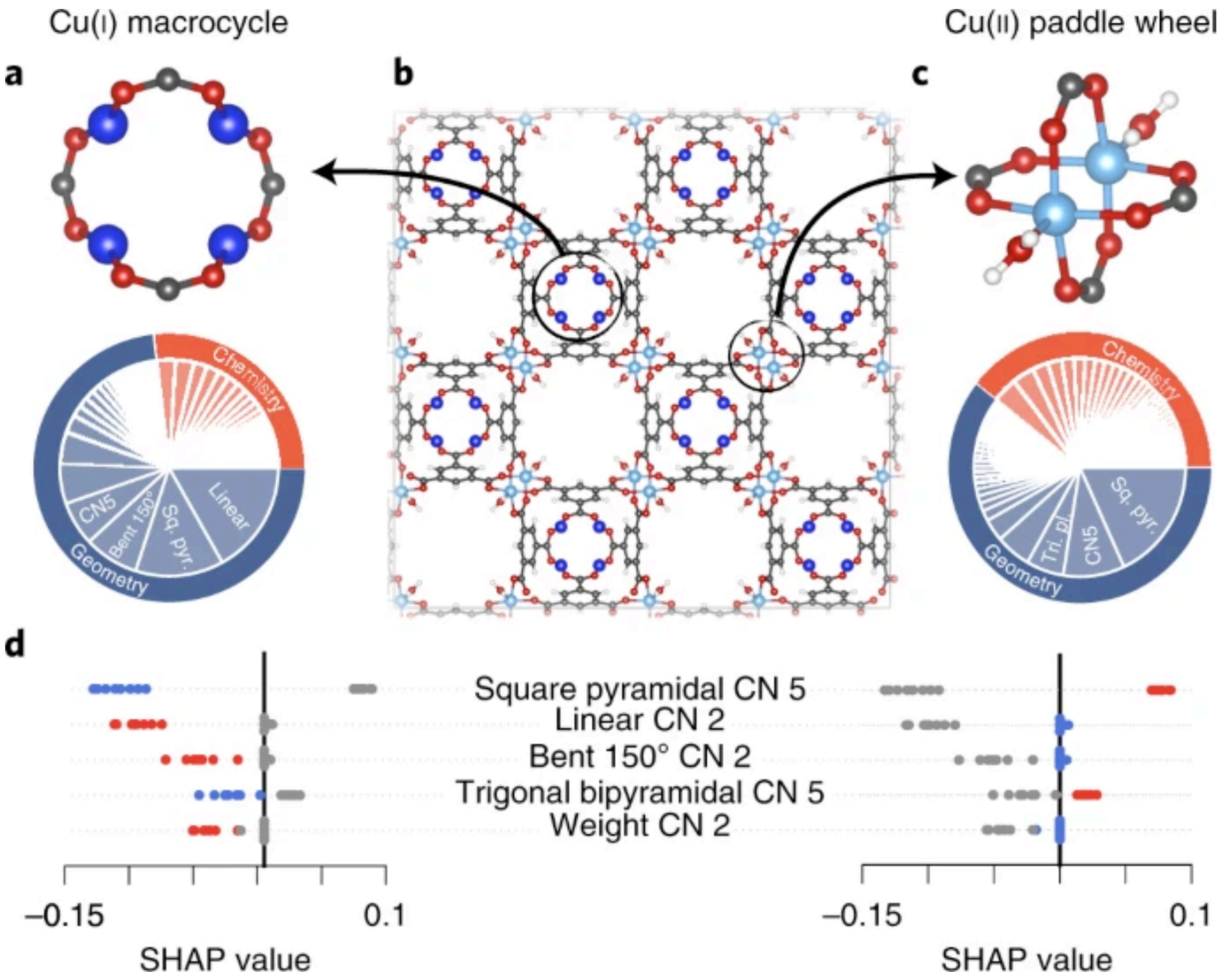
(a) Sheep - 26%, Cow - 17%

(d) Bird - 100%, Person - 39%

https://bdtechtalks.com/rise-explainable-ai-example-saliency-map/

Motivation

Chemical insights into system



Permutation feature importance

Look at performance drop when trained on shuffled features

0. Build the model

$$X = \begin{pmatrix} x^{1,1} & x^{1,1} & \dots & x^{1,p} \\ x^{2,1} & x^{2,1} & \dots & x^{2,p} \\ \vdots & \ddots & & \vdots \\ x^{n,1} & x^{n,1} & \dots & x^{n,p} \end{pmatrix} \quad y = \begin{pmatrix} y^1 \\ y^2 \\ \vdots \\ y^n \end{pmatrix} \quad \hat{y}_{ML} = f(X), \mathscr{L}(y, \hat{y}_{ML})$$

1. Estimate the model error

$$e^{\text{orig}} = \mathscr{L}(y, \hat{y}_{ML})$$

2. For each feature *j=1,..,p*:

   - Generate permuted feature matrix for feature (*j*)

$$X_j^{\text{perm}} = \begin{pmatrix} 1 & x^{1,1} & \dots & x^{50,j} & \dots & x^{1,p} \\ 1 & x^{2,1} & \dots & x^{n,j} & \dots & x^{2,p} \\ 1 & \vdots & \dots & \vdots & \ddots & \vdots \\ 1 & x^{n,1} & \dots & x^{2,j} & \dots & x^{n,p} \end{pmatrix}$$

   - Estimate the error with permuted features

$$e_j^{\text{perm}} = \mathscr{L}(y, f(x_j^{\text{perm}}))$$

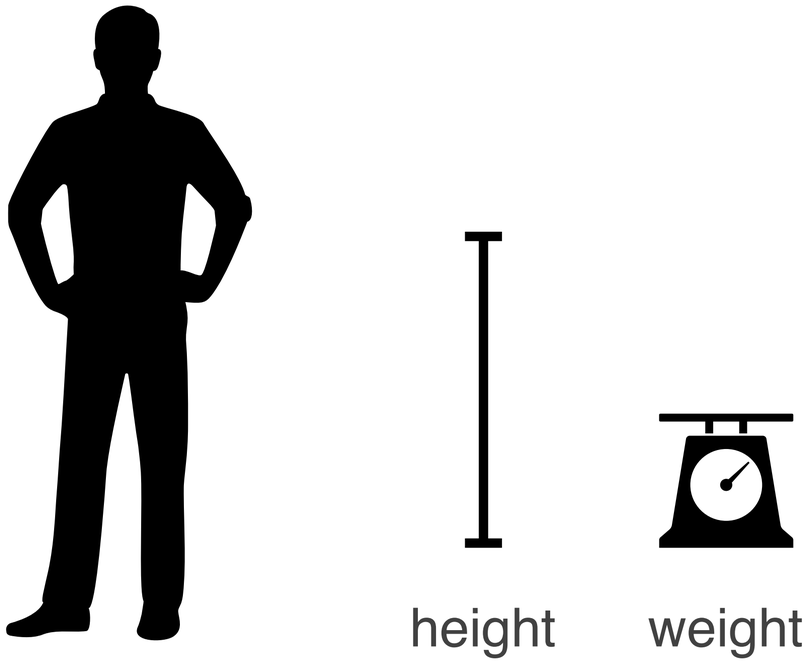
   - Get the the performance difference

$$e_j^{\text{perm}} - e_j^{\text{orig}}$$

Permutation feature importance

Caveats with correlated features

- Importance of correlated features might be split
- Might generate unreasonable data points



7. Feature importance

Learning Objectives:
- Motivation for explainable ML
- Permutation feature importance

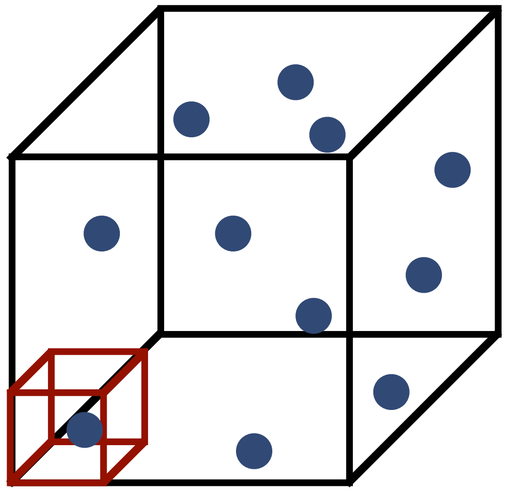
8. Feature Selection

Learning Objectives:

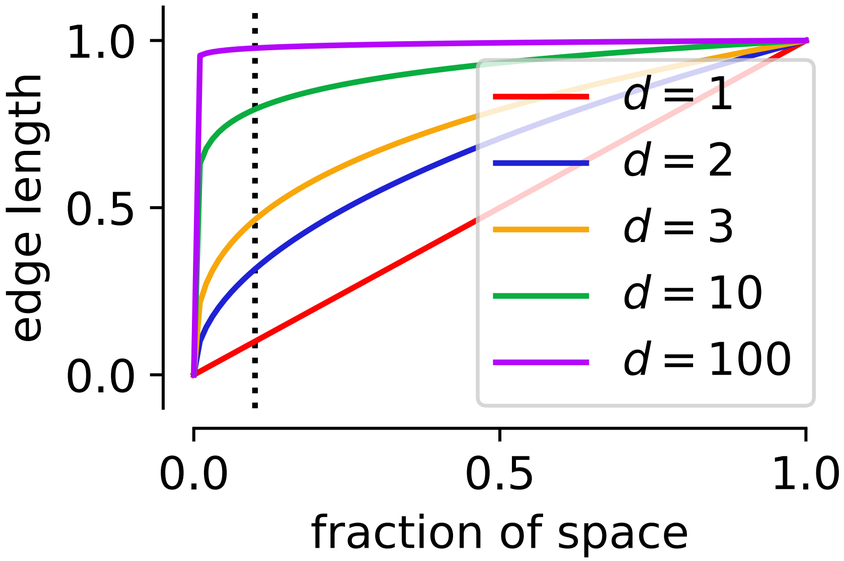
- Curse of dimensionality: No locality in high-dimensional spaces
- Filter heuristics
- LASSO

No locality in high dimensions

$$\text{edgelength} = \text{fraction of space}^{1/\text{dimensionality}}$$



unit cube



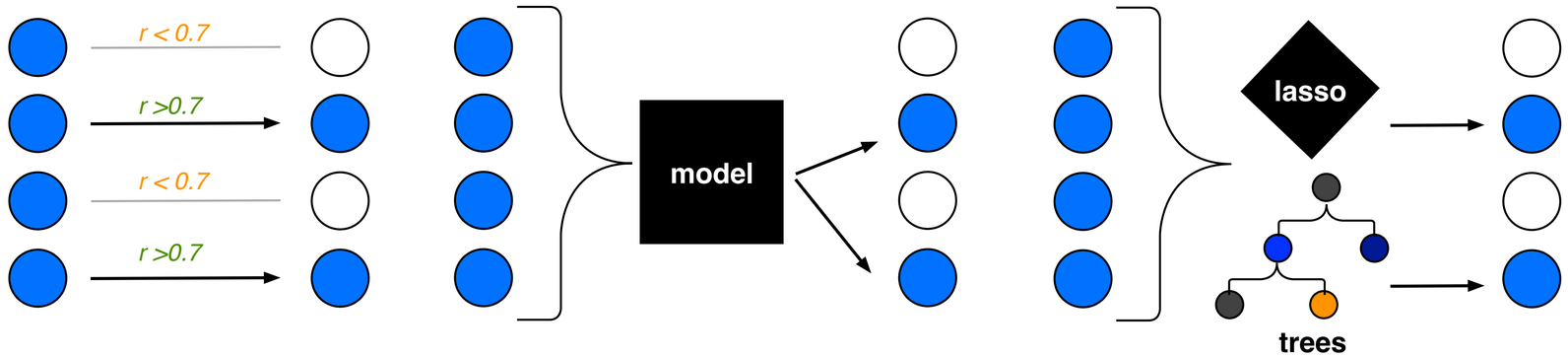
Methods that are based on similarity (*k*NN/KRR) might fail in high dimensional spaces!

We want to **reduce the dimensionality** of our feature matrix!

Domingos, P. A Few Useful Things to Know about Machine Learning. *Communications of the ACM* **2012**, *55* (10), 78..

Feature selection

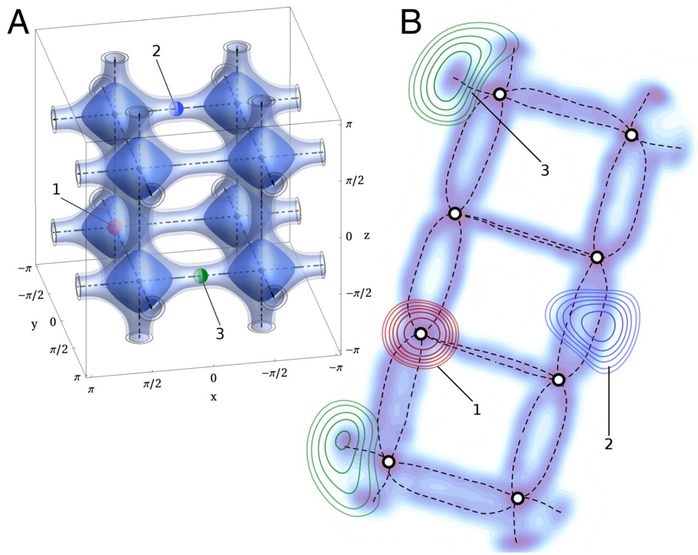
Reduce dimensionality of feature space by feature selection (compression)



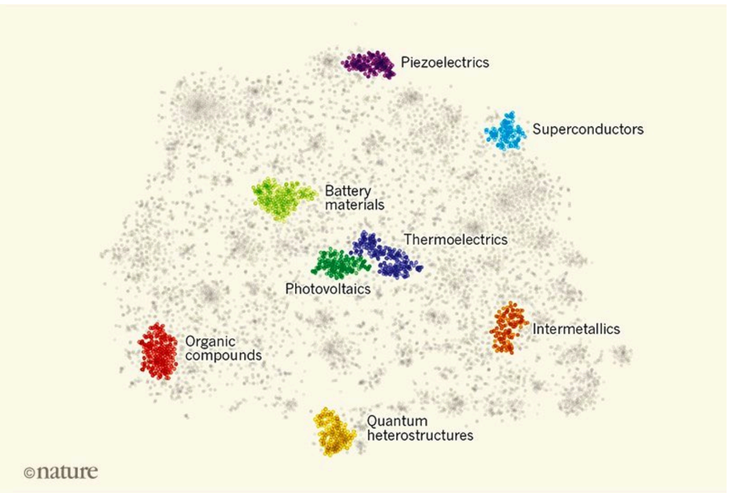
a Univariate filters. **b** Wrapper methods. **c** Shrinkage or direct.

Feature projection

Reduce size of feature space by dimensionality reduction (feature projection)

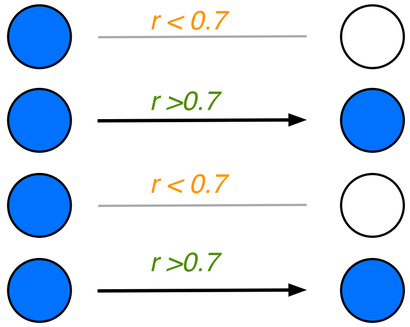


Visualize data and Materials Cartography



Janet, J. P.; Kulik, H. J. *J. Phys. Chem. A* **2017**, *121* (46), 8939–8954.
Tribello, G. A.; Ceriotti, M.; Parrinello, M. *PNAS* **2012**, *109* (14), 5196–5201.

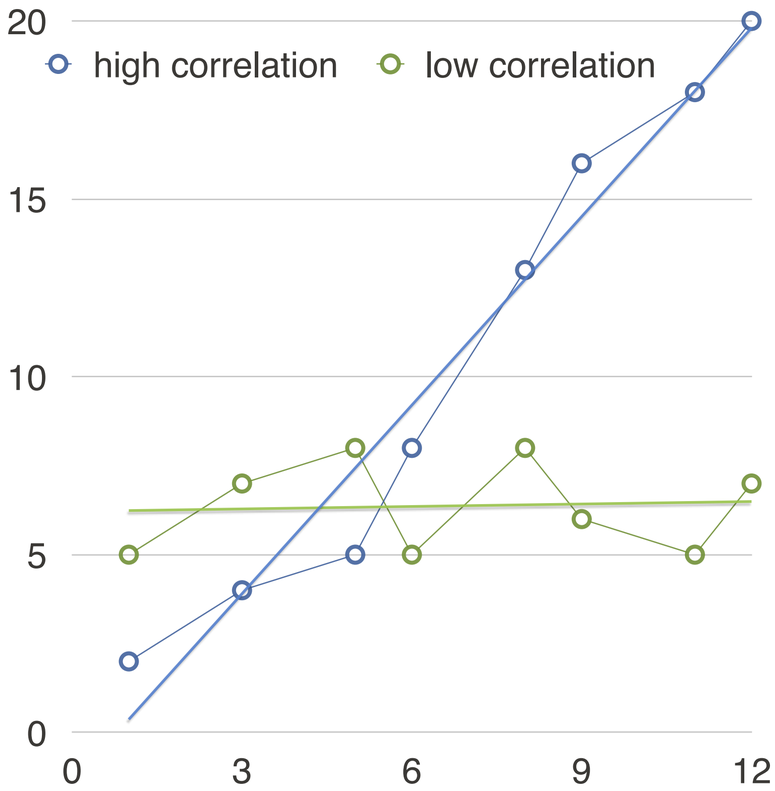
Feature Selection: Filter Methods



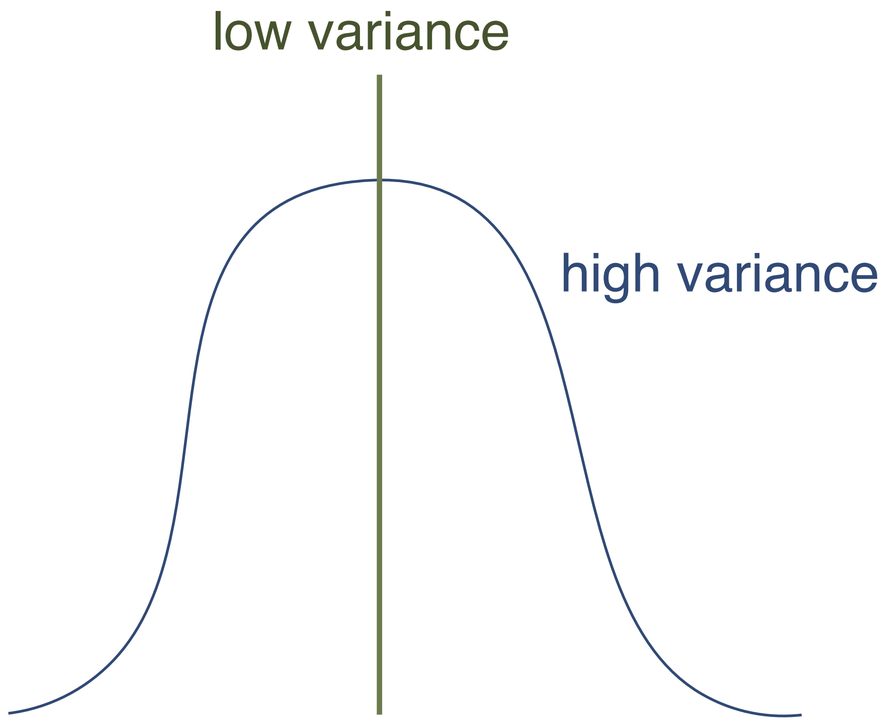
Easy and cheap

Interaction effects are not considered

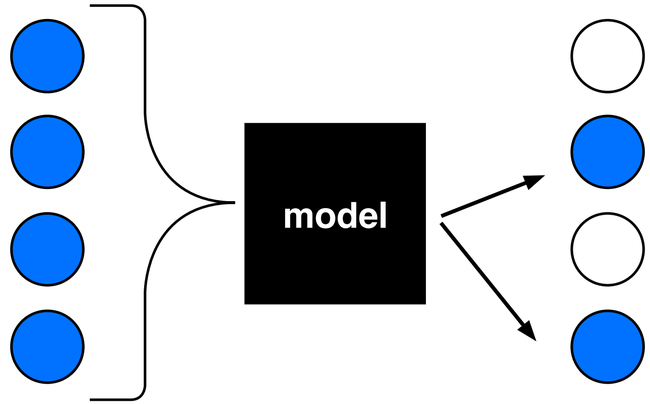
correlation threshold



variance threshold



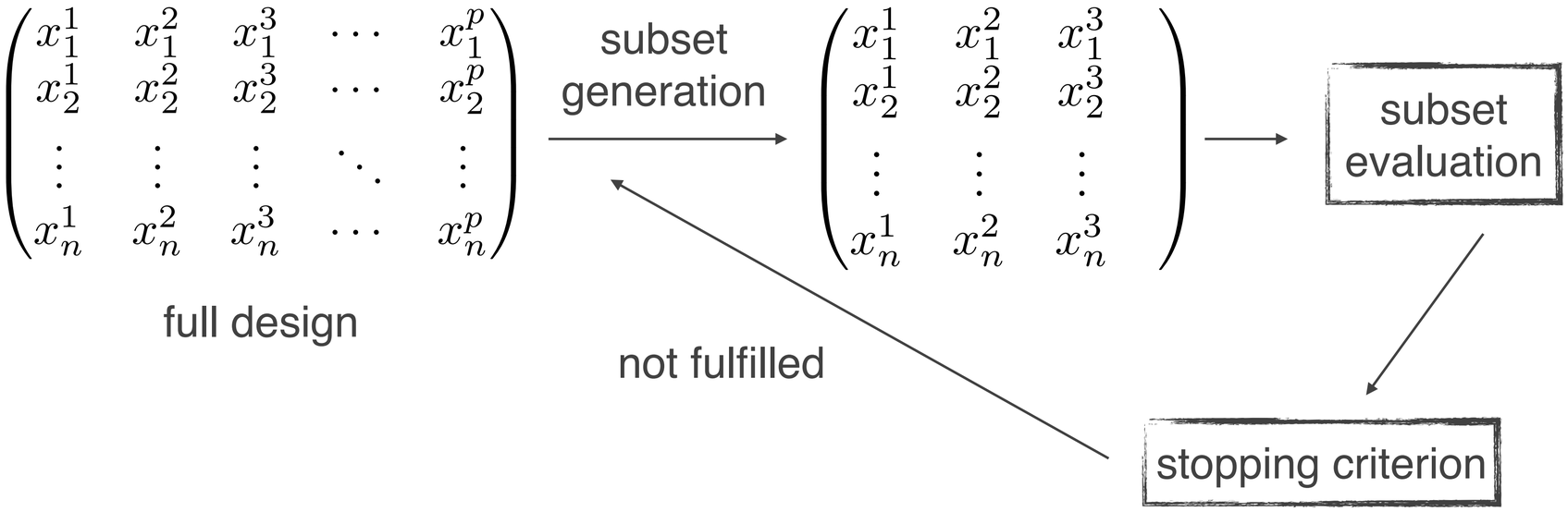
Feature Selection: Wrapper Methods



Uses a good surrogate of the real objective

Expensive

For example recursive feature addition or elimination

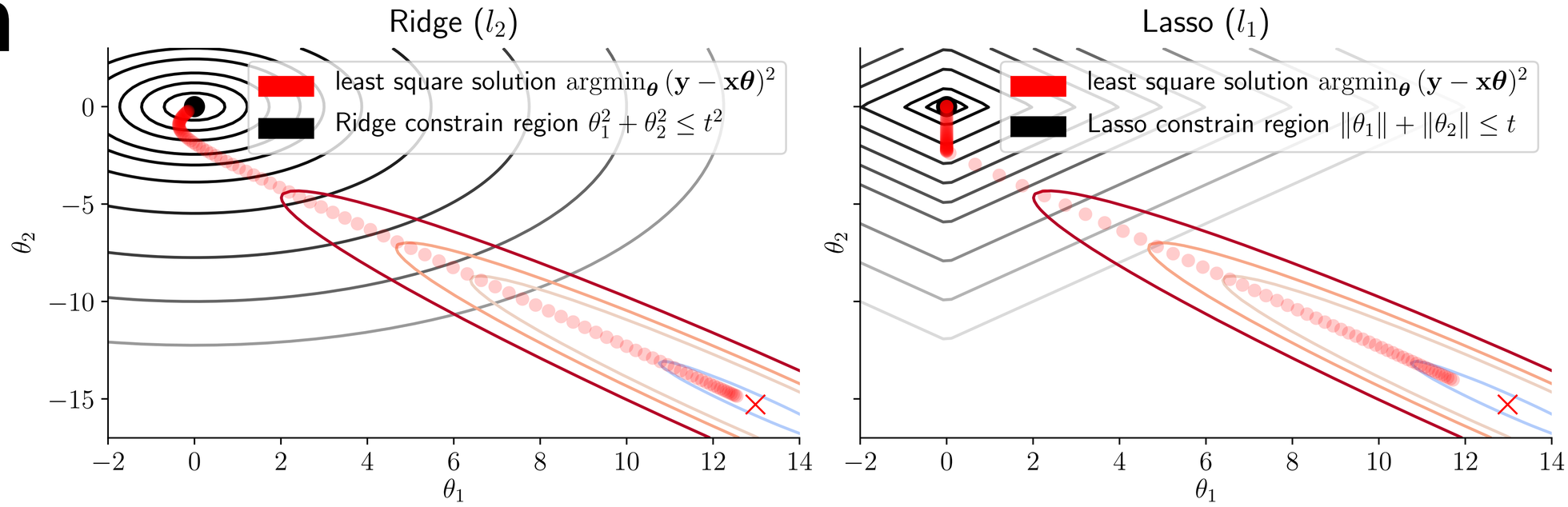


Feature Selection: Relaxing best subset selection

The basic problem: Best subset selection

$$\min_{\beta \in \mathbb{R}^p} \|y - X\beta\|^2 \text{ subject to } \quad \|\beta\|_0 \leq k \qquad \|\beta\|_0 = \sum_{j=1}^{p} 1\{\beta_j \neq 0\}$$

But this is our hard problem (NP hard) …
… hence we relax the constraint to have a problem that is convex
… the Lasso gives use sparsity as the most feasible approximation to l_0



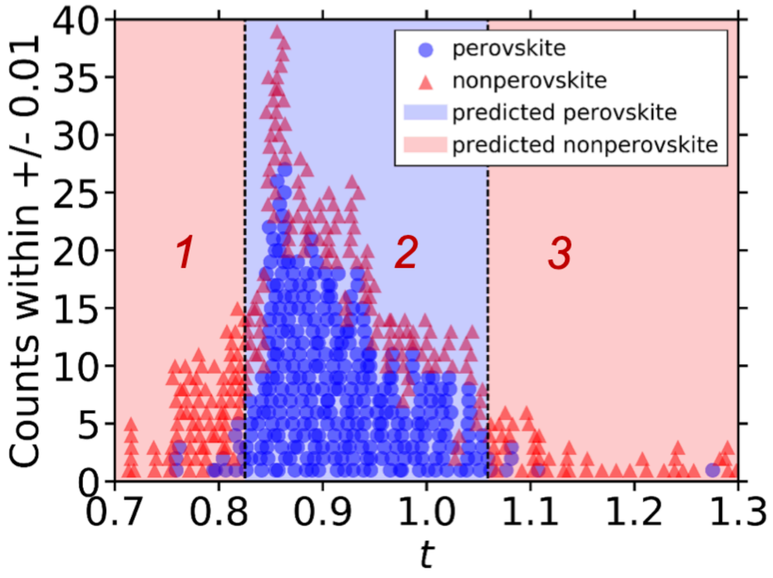
Hastie, T.; Tibshirani, R.; Wainwright, M. *Statistical Learning with Sparsity: The Lasso and Generalizations*; Monographs on statistics and applied probability; CRC Press, Taylor & Francis Group: Boca Raton, 2015.

Feature Selection: LASSO in practice

primary features ⟶ **billions** of feature candidates $\xrightarrow{\text{(SI) +Lasso}}$ subset of features

$$\text{primary features} = \{r_A, r_B, n_a, n_b, \ldots\} \qquad \hat{R} = \{+, -, \cdot, \exp, \lg, ^{-1}, ^{2}, ^{3}, \sqrt{}, \|\cdot\|\}$$

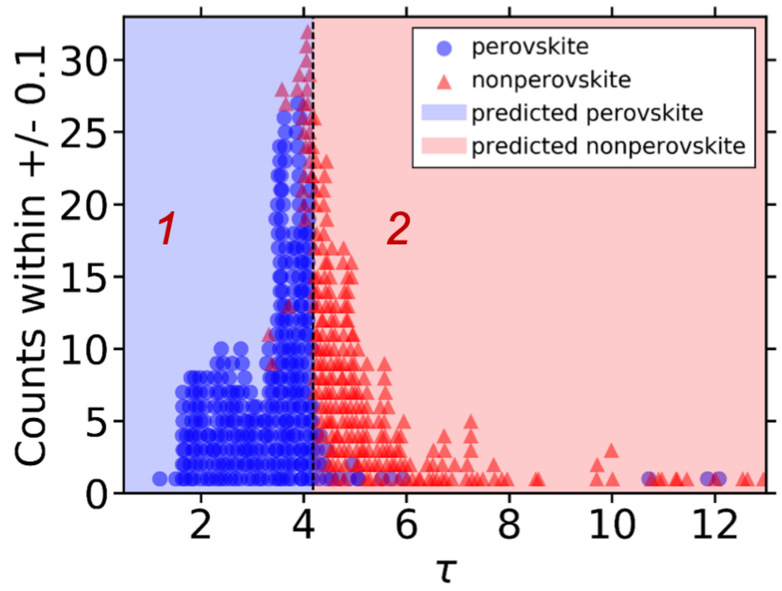
$$t = \frac{r_A + r_X}{\sqrt{2}(r_B + r_X)}$$



$0.825 < t < 1.059 \rightarrow$ perovskite

74% accuracy

$$\tau = \frac{r_X}{r_B} - n_A\left(n_A - \frac{r_A/r_B}{ln\, r_A/r_B}\right)$$



$\tau < 4.18 \rightarrow$ perovskite

92% accuracy

Ghiringhelli, *et al. Phys. Rev. Lett.* **2015**, *114* (10), 105503.
Ouyang, R.; *et al. Phys. Rev. Materials* **2018**, *2* (8), 083802.
Bartel, C.; *et al.*, M. *Sci. Adv.* **2019**, *5* (2), eaav0693.

8. Feature Selection

Learning Objectives:

- Curse of dimensionality: No locality in high-dimensional spaces
- Filter heuristics
- LASSO

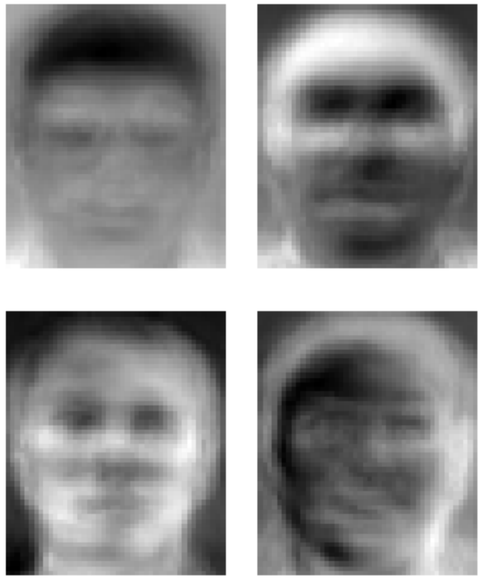
9. Feature Projection

Learning Objectives:

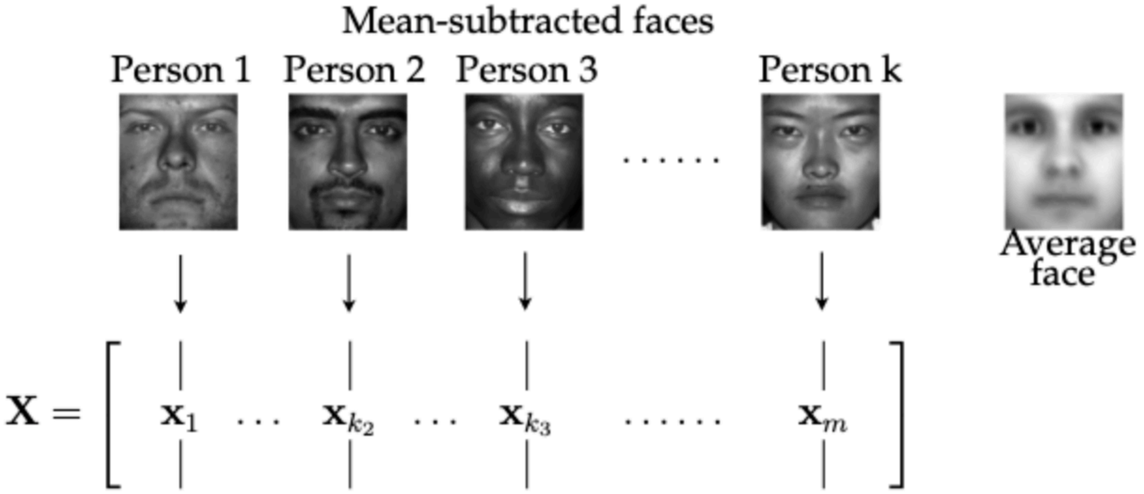
- Principal component analysis, t-SNE: Principles and pitfalls
- Projecting high dimensional data into a lower dimensional space

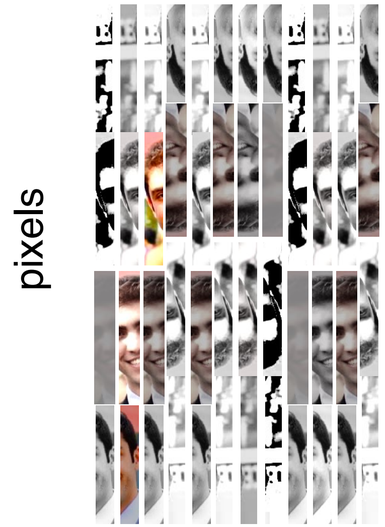
Feature Projection: Projecting High-Dimensional Data

Eigenfaces: Principal Component Analysis on face images to get "basis vectors" of face image space



Linear combination of basis vectors





PCA

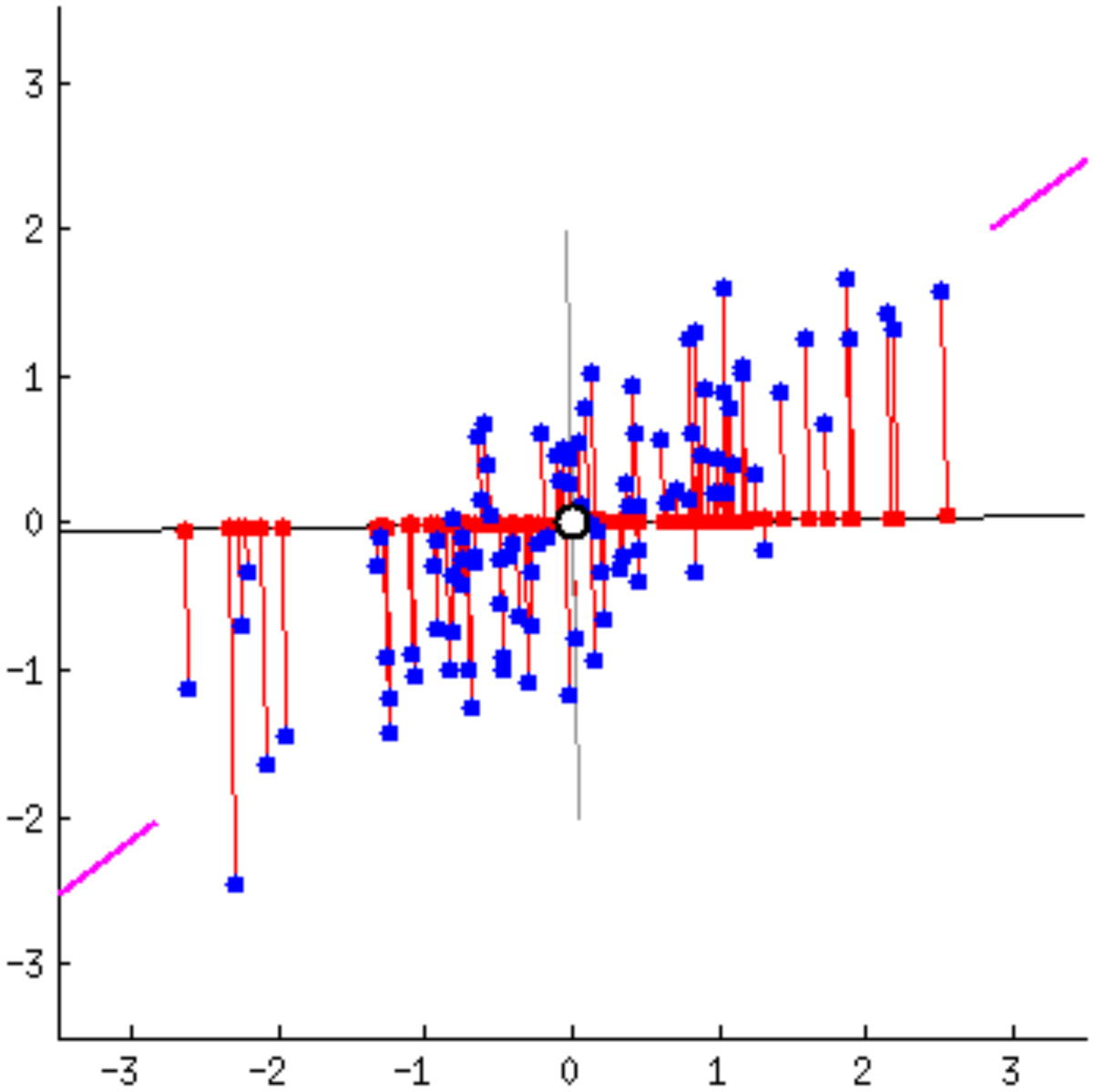
Find basis vectors

many faces

Navarrete, P.; Ruiz-Del-Solar, J. *Int. J. Patt. Recogn. Artif. Intell.* **2002**, *16* (07), 817–830.

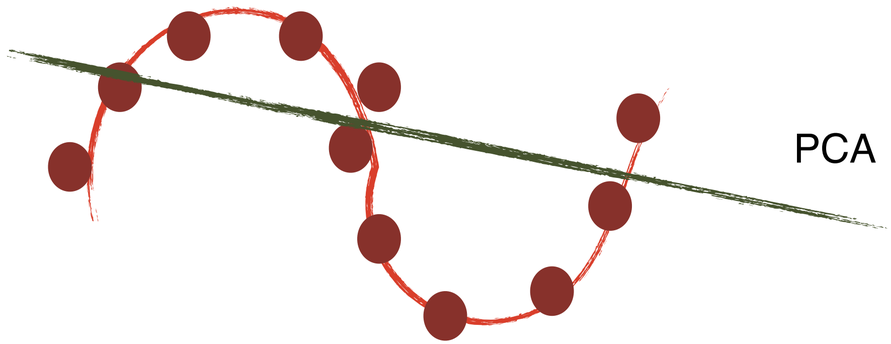
PCA Intuition

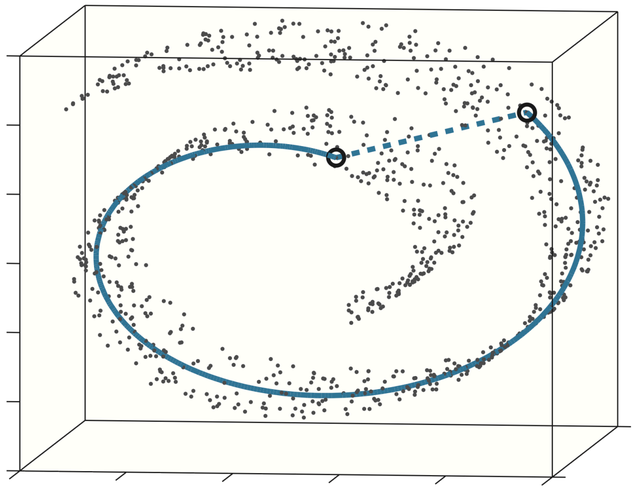
Finding the line that captures most of the variance, i.e., maximum decorrelation



Caveats with PCA

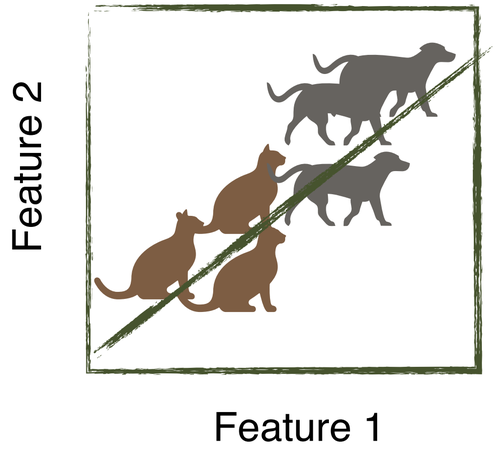
Non-linearity

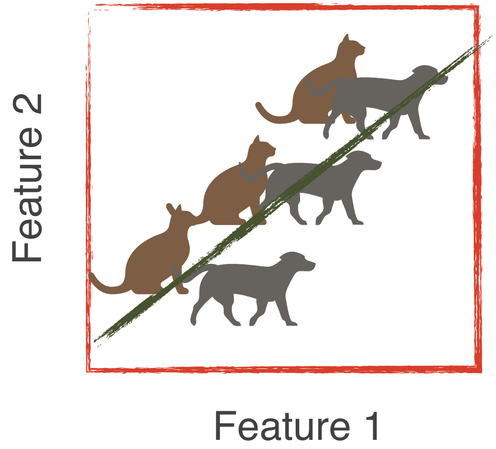




- Data is linearly uncorrelated
- But there is still a non-linear dependence

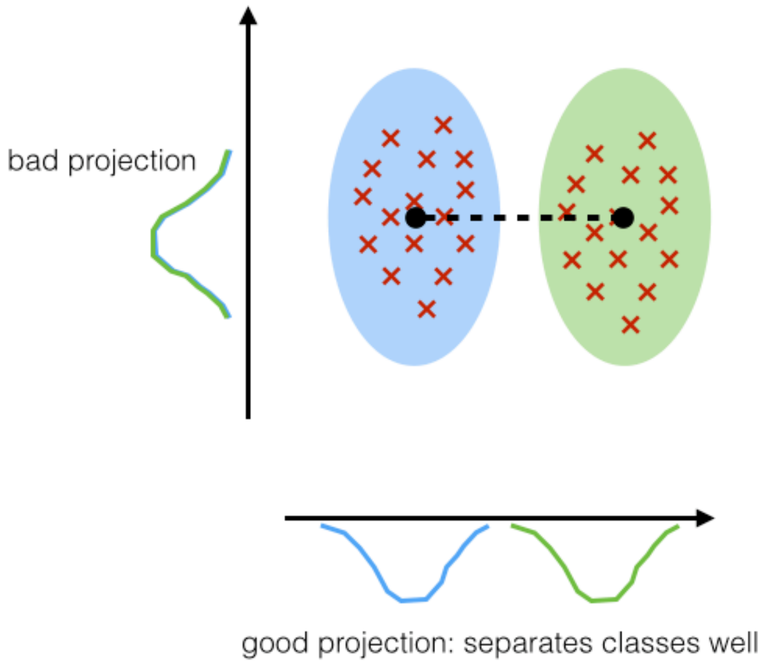
Higher variance feature is not more discriminative





Other member of dimensionality reduction zoo

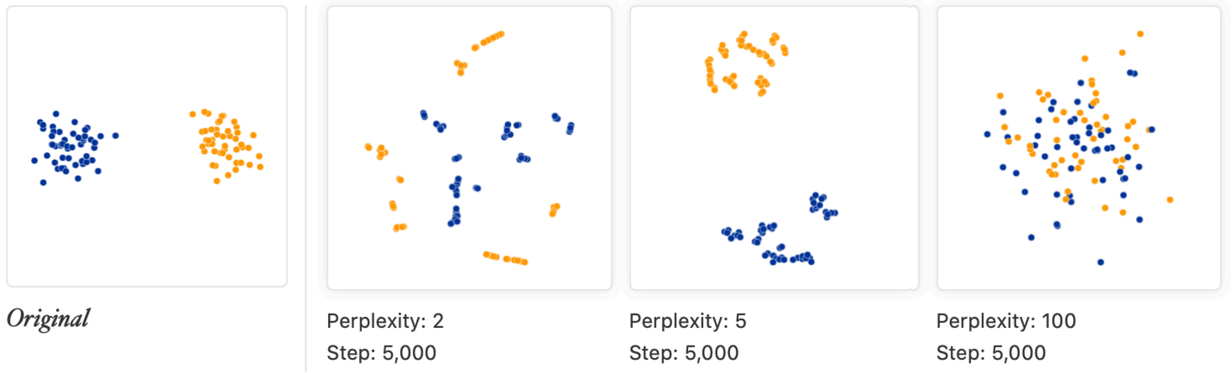
Linear Discriminant Analysis (LDA)



- math like for PCA
- maximizing component axes for class

t-distributed stochastic neighbor embedding (*t*-SNE)

- Non-linear
- Conditional probabilities that represent similarities
- Sensitive to perplexity (number of neighbors)



Wattenberg, et al., "How to Use t-SNE Effectively", *Distill*, **2016**. Talk by Laurens van der Maaten: https://bit.ly/2RaFJIw

9. Feature Projection

Learning Objectives:

- Principal component analysis, t-SNE: Principles and pitfalls
- Projecting high dimensional data into a lower dimensional space

10. Feature Learning

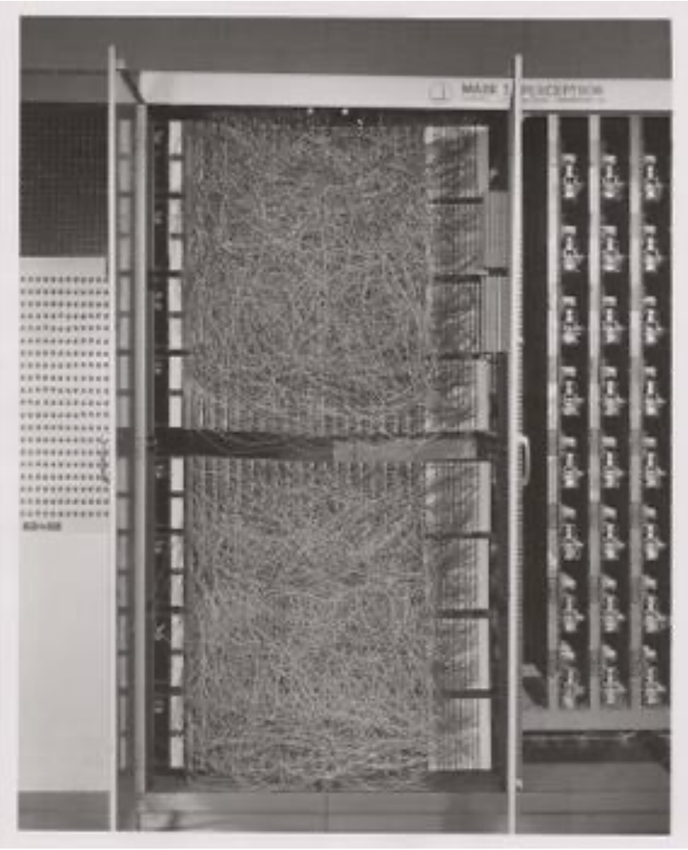
Learning Objectives:

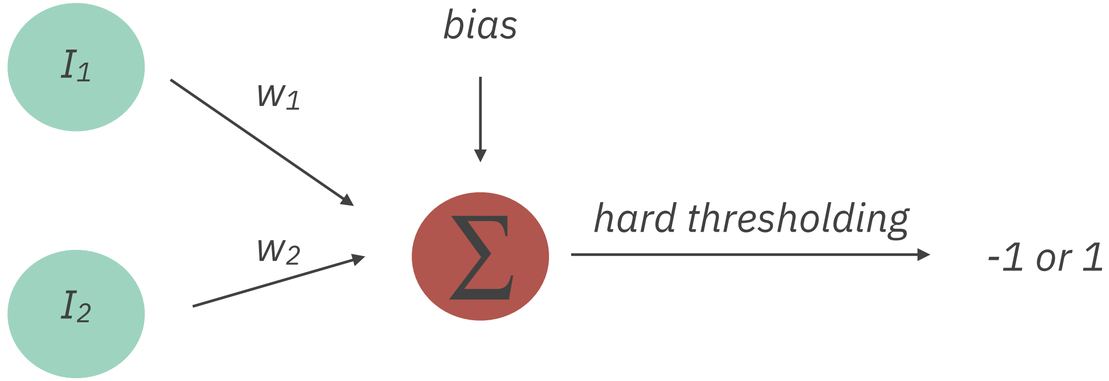
- Multiple layers with non-linearities can act as universal function approximations
- Intermediate layers learn predictive representations
- Symmetry of problem guides architecture choice

Neural Networks: Perceptron (*"may eventually be able to learn, make decisions, and translate languages"*)

The Perceptron (Rosenblatt (1957))

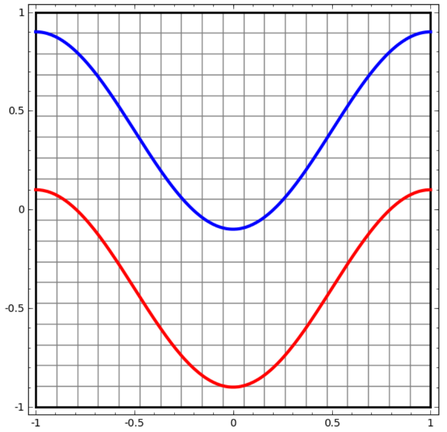
Mark 1 built for image recognition

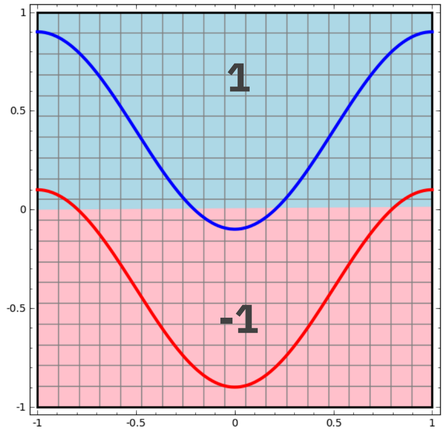




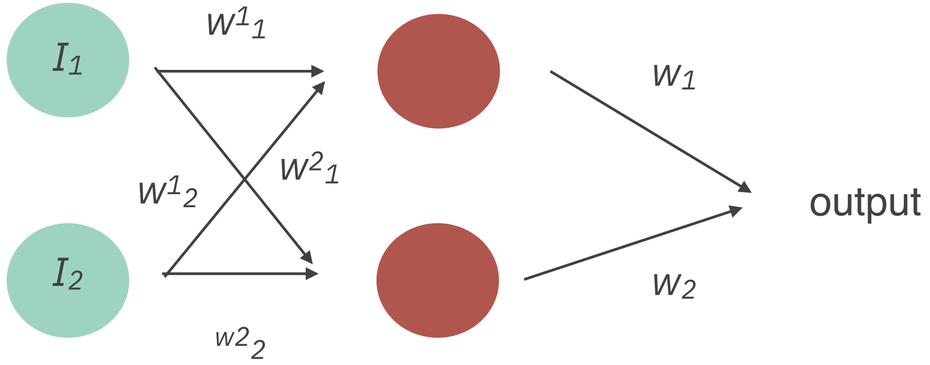
Try to classify which points belong to which curve.

Decision boundary: $w_1x_1 + w_2x_2 + b = 0$

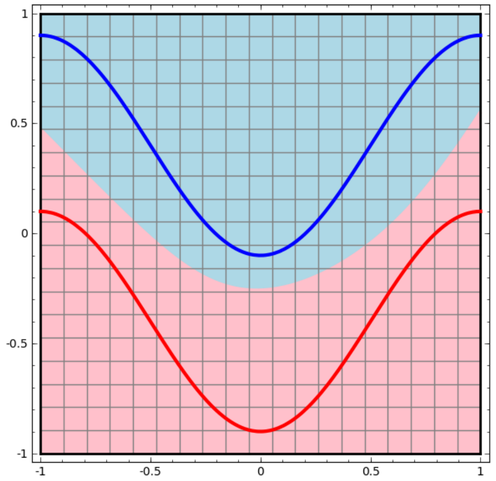




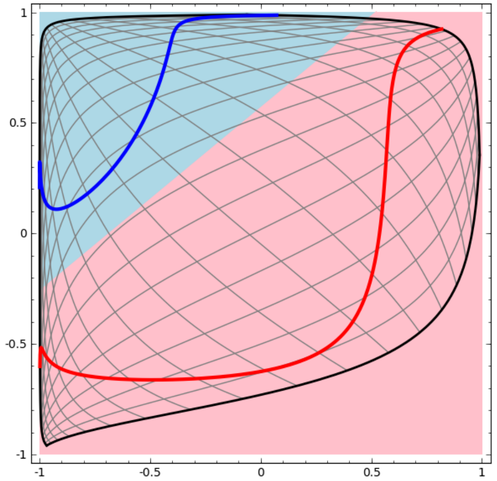
Neural Networks: Representation learning



transform representation such that it is linearly separable

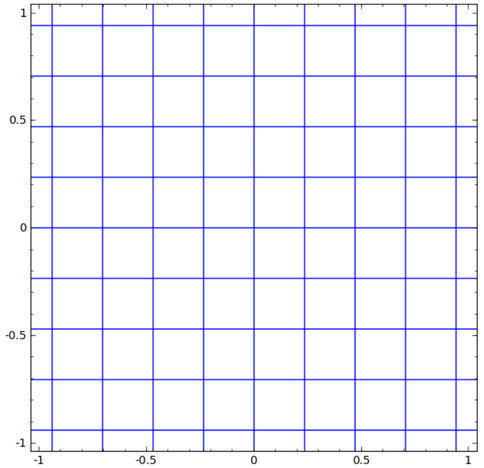


hidden layer, e.g. tanh



in tanh(**Wx**+b):

- rotate (**W**)
- Translate (b)
- point-wise application of tanh

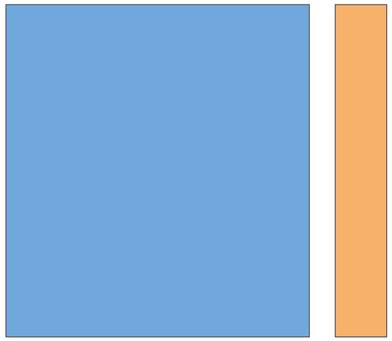


Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning.* The MIT Press: Cambridge, Massachusetts, 2016.
LeCun, Y.; Bengio, Y.; Hinton, G. *Nature* **2015**, *521* (7553), 436–444.
http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Explore NNs at https://playground.tensorflow.org!

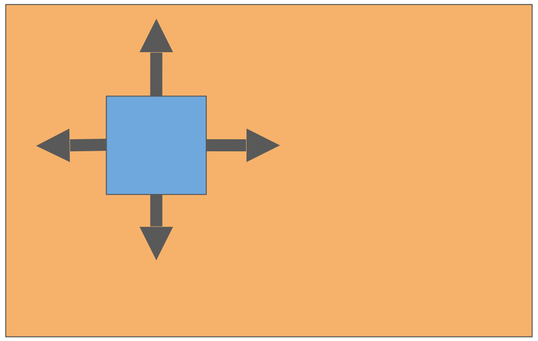
The right architecture for your symmetry

Arrays
⇨ Dense NN



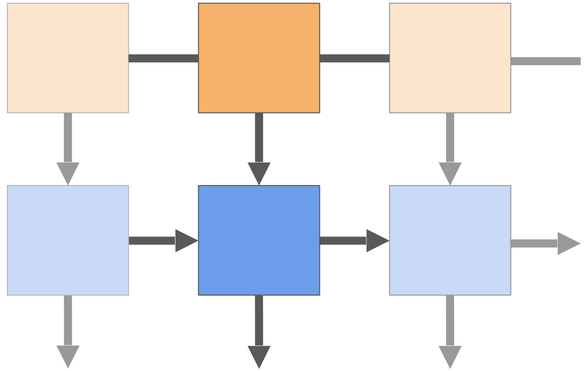
Components are independent.

2D images
⇨ Convolutional NN



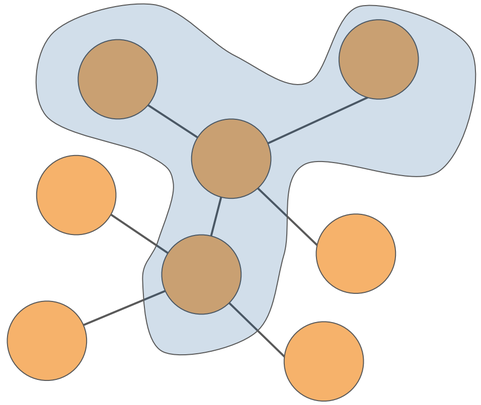
The same features can be found anywhere in an image. Locality.

Text
⇨ Recurrent NN



Sequential data. Next input/ output depends on input/ output that has come before.

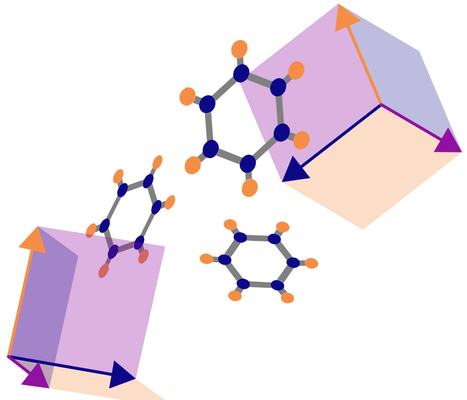
Graph
⇨ Graph (Conv.) NN



Non-Euclidean data. Nodes have features and network passes messages between nodes connected via edges.

3D physical data
⇨ Euclidean NN

Data in 3D Euclidean space.



10. Feature Learning

Learning Objectives:

- Multiple layers with non-linearities can act as universal function approximations
- Intermediate layers learn predictive representations
- Symmetry of problem guides architecture choice

Applications

Learning Objectives:

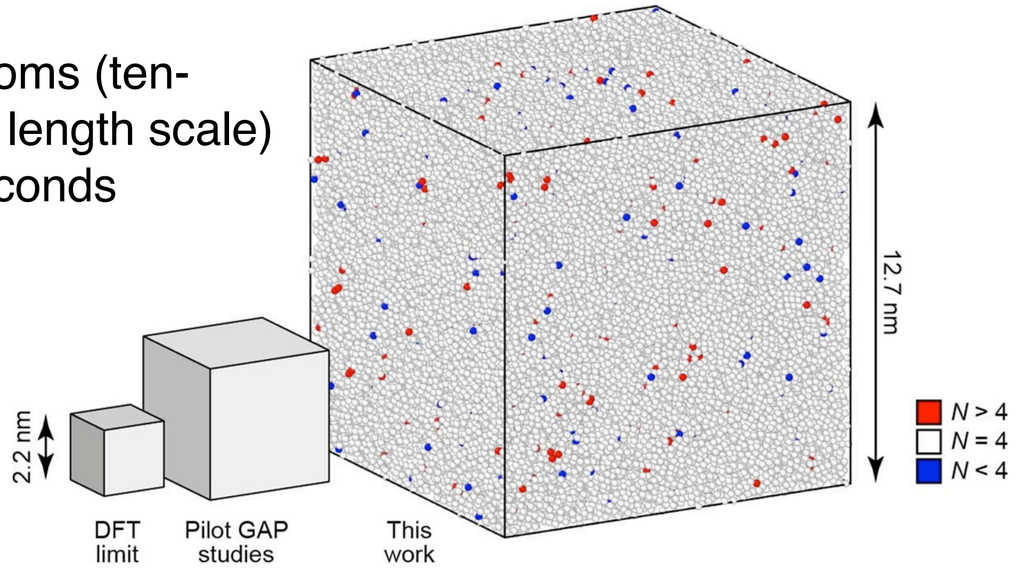
- ML “force fields”
- ML for efficient sampling
- Using natural language processing for materials discovery

Machine learned force fields

- High accuracy for long time scales and/or large systems
- ML potentials are close to being a “commodity” technique

Nature **589**, 59–64 2021.

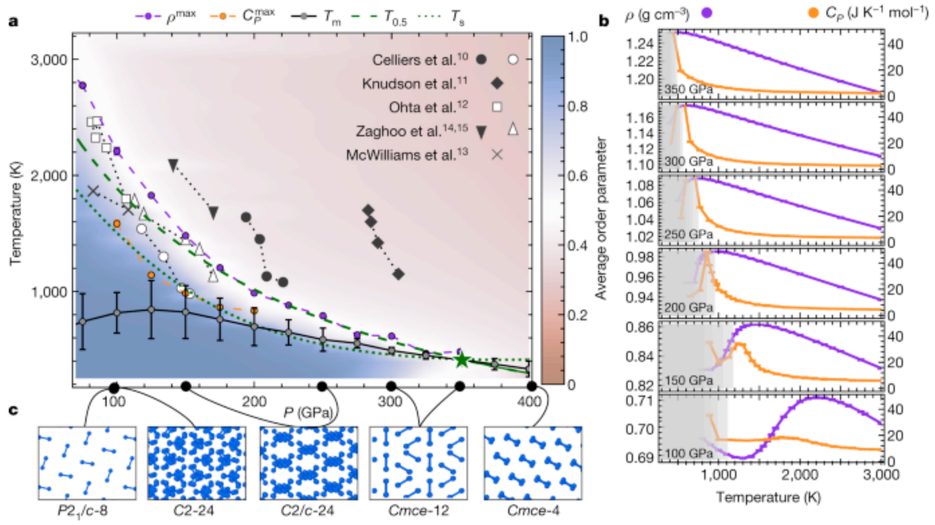
100,000 atoms (ten-nanometre length scale) for nanoseconds



Hydrogen in planets (under high pressure):

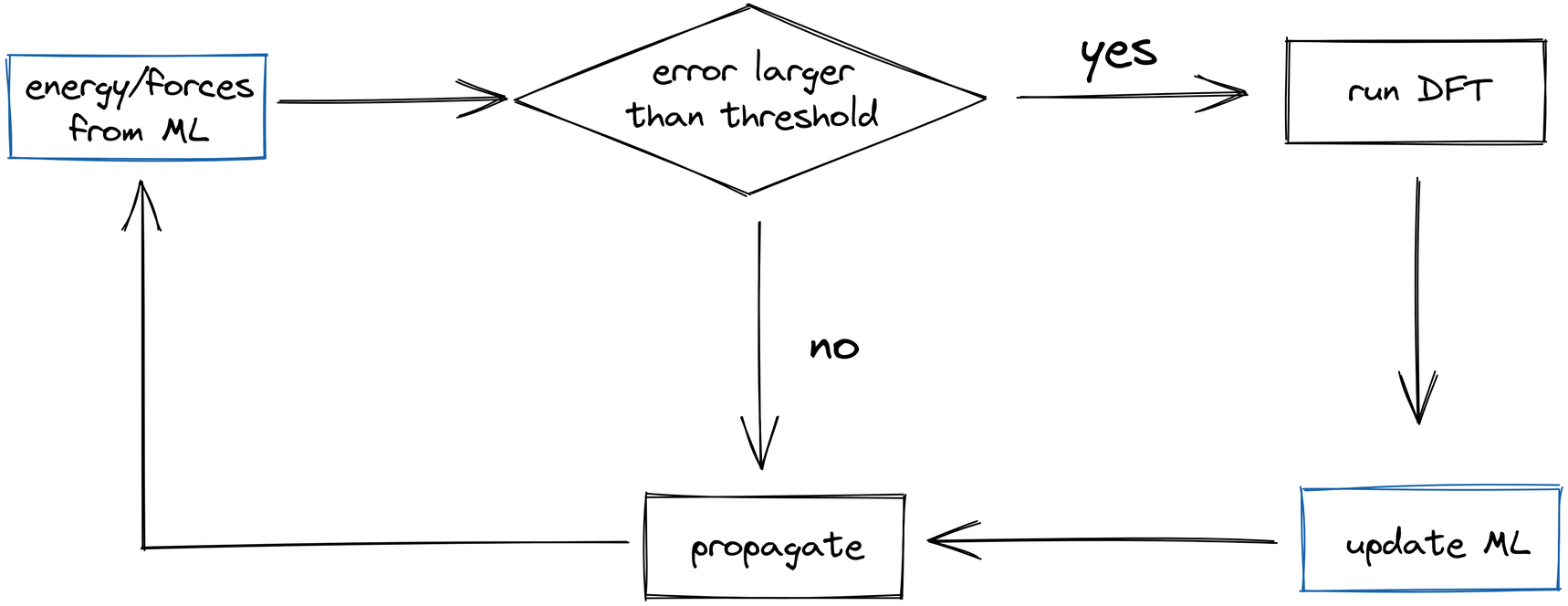
Thousands of atoms for nanoseconds and milli-electronvolt energy differences

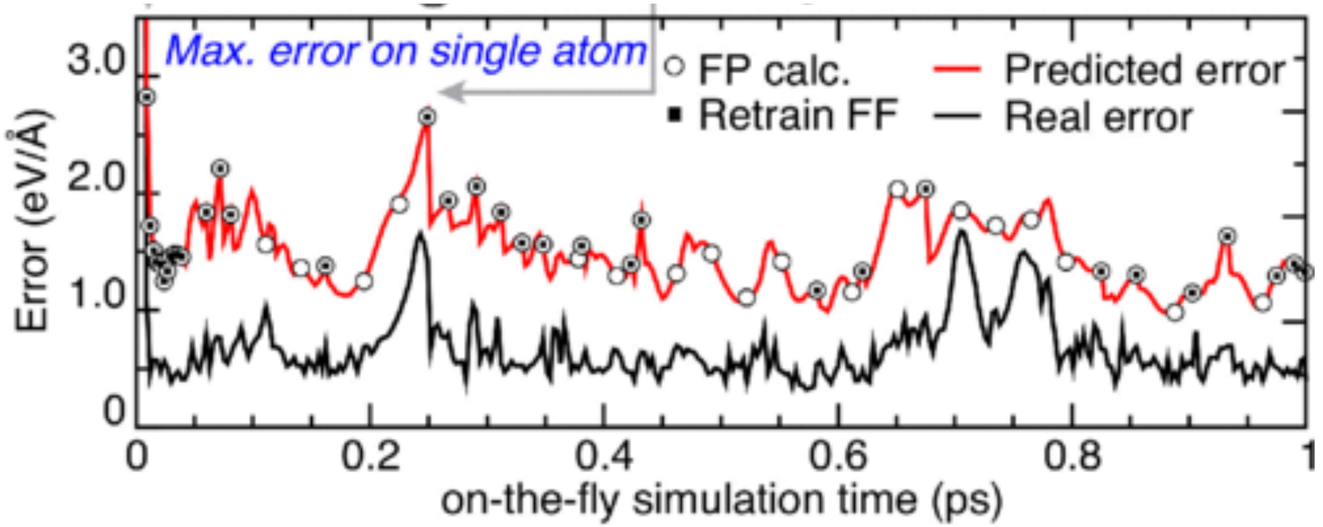
Nature **585**, 217-220 2020.



Machine learned force fields

DFT accuracy with only 1% DFT!

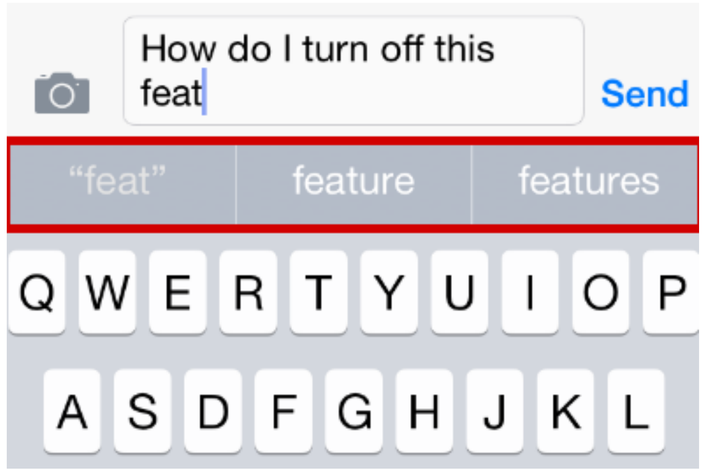




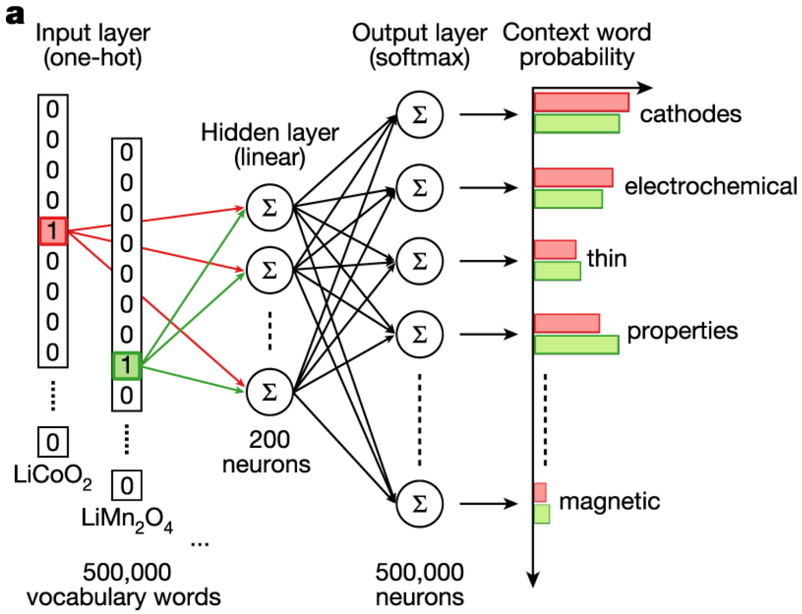
Phys. Rev. Lett. **2019**, *122* (22), 225701.

Unsupervised word embeddings

ferromagnetic–NiFe + IrMn ≈ antiferromagnetic

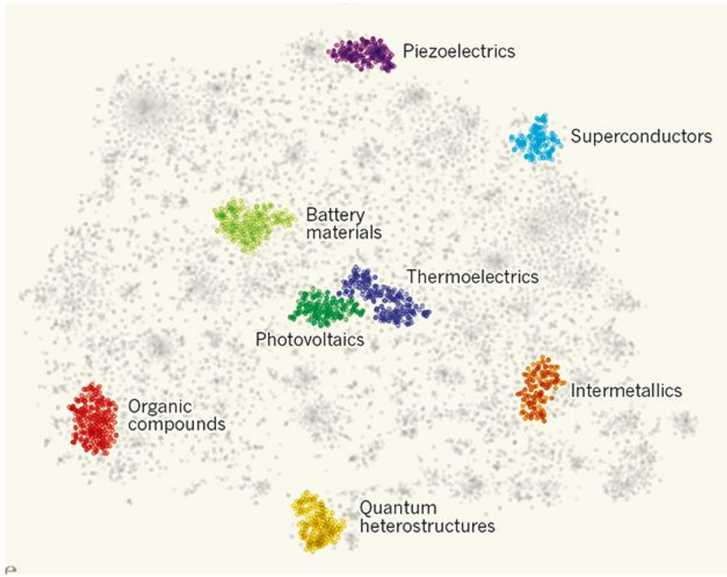


single layer NN is trained to predict all context words for the given target word.



for similar words the context words are the same

Projection of embeddings onto two dimensions (t-SNE).

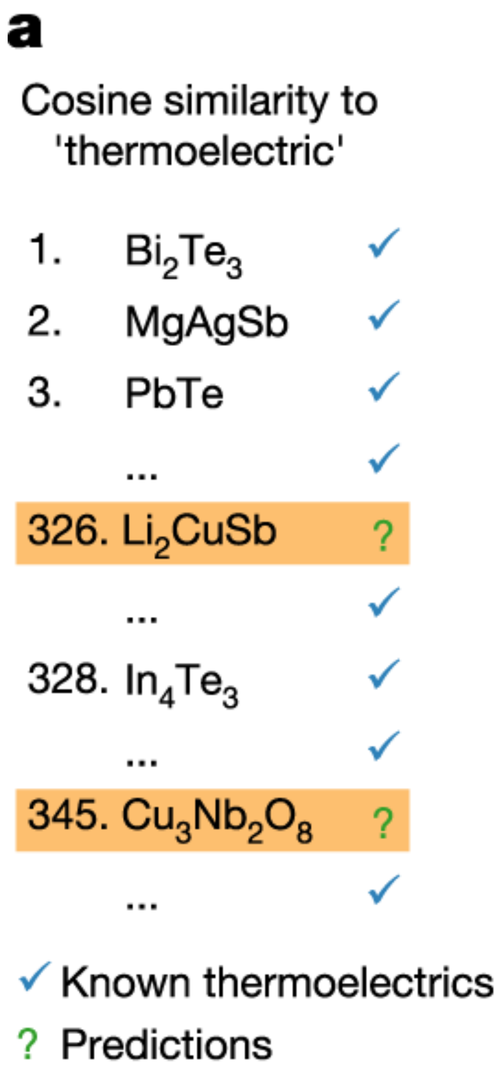


Tshitoyan, V.; *et al.*. *Nature* **2019**, *571* (7763), 95.
Perspective: Isayev, O. *Nature* **2019**, *571* (7763), 42–43.

Unsupervised word embeddings

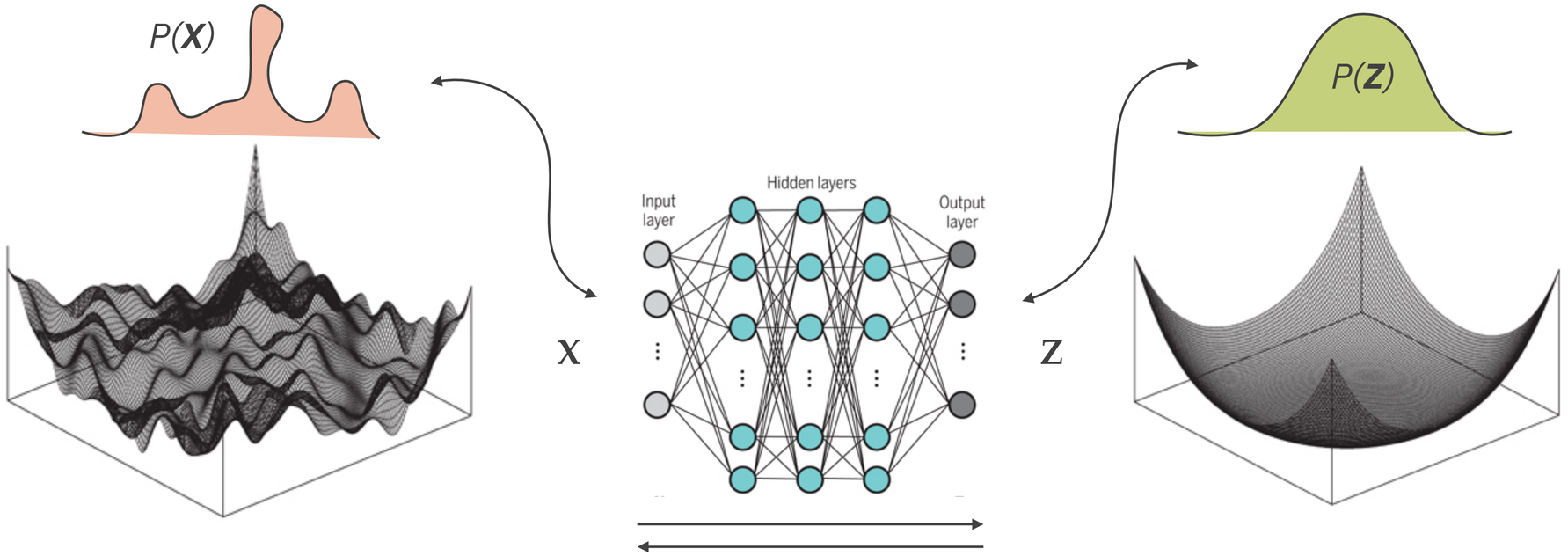
Embeddings can also be used for predictions.

- Top ten predictions even slightly higher than known average
- Better rank correlation with experiments than DFT



Boltzmann samplers

Molecular Simulations: *U(**X**)* given, need approach to sample *P(**X**)*



$-kT\lg(P(\mathbf{X}))$

Landscapes are often rugged, hard to sample

Invertible neural network does the invertible mapping

$-kT\lg(P(\mathbf{Z}))$

Noé, F.; Olsson, S.; Köhler, J.; Wu, H. *Science* **2019**, *365* (6457)
Perspective: Tuckerman, M. E. *Science* **2019**, *365* (6457), 982–983.,

Applications

Learning Objectives:

- ML “force fields”
- ML for efficient sampling
- Using natural language processing for materials discovery