Learning Representations of Data:

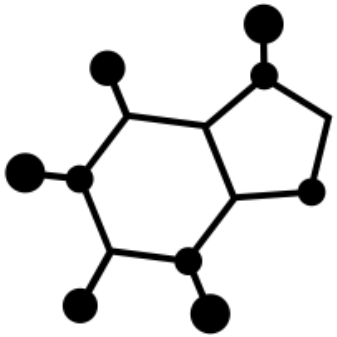
An introduction to the Machine/Deep Learning Toolkit (with molecules)

Benjamin Sanchez-Lengeling

CrossTalk #3

Panorama: Yesterday

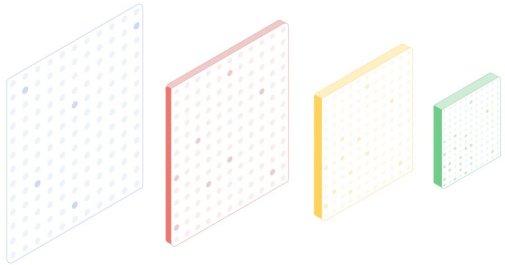
ML/EDA for molecular data



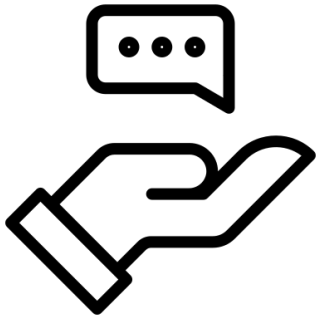
DEL data

Panorama

2. The (neural) modelling choices



1. Learning representations of data (with DL)



3. Miscellaneous practical advice

Activity: Talk to someone spatially distant

1. How do we convert molecules into numbers?

2. If someone sends you a DEL dataset, what the first thing you might want to with it?

3. What are some of the challenges of DEL data?

Panorama

- What is a representation?
- How do we learn a representation?
- Encoder/Decoder frameworks

1. Learning representations of data (with DL)

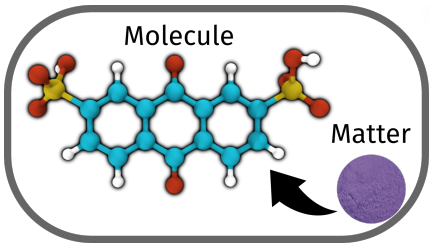
2. The (neural) modelling choices

3. Miscellaneous practical advice

The idea of a representation (for molecules)

Very related to embeddings, feature vectors, latent codes, intermediate activations, etc.

Our data as "observed in the real world"



Approximation

Our numerical **representation**

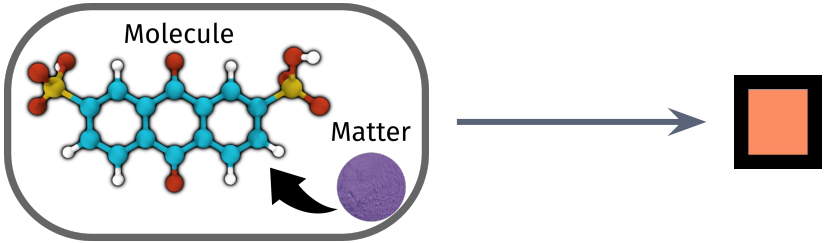
A "good" representation

- Generalizable
- Meaningful distances
- "Local Smoothness" wrt to Activity Cliffs
- Interpretable
- Respects: Symmetries

Operations:

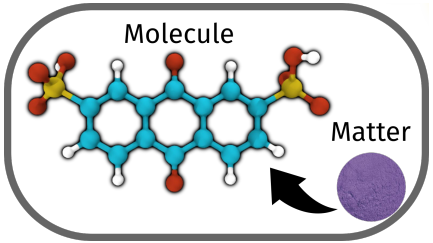
- Search
- Optimize
- Rank
- Generate (with constraints)

Building a representation can be simple: Molecular Weight.



- Multiple molecules have the same molecular weight (not unique!).
- Not directly interpretable.

Building a representation can be simple: Molecular Weight.

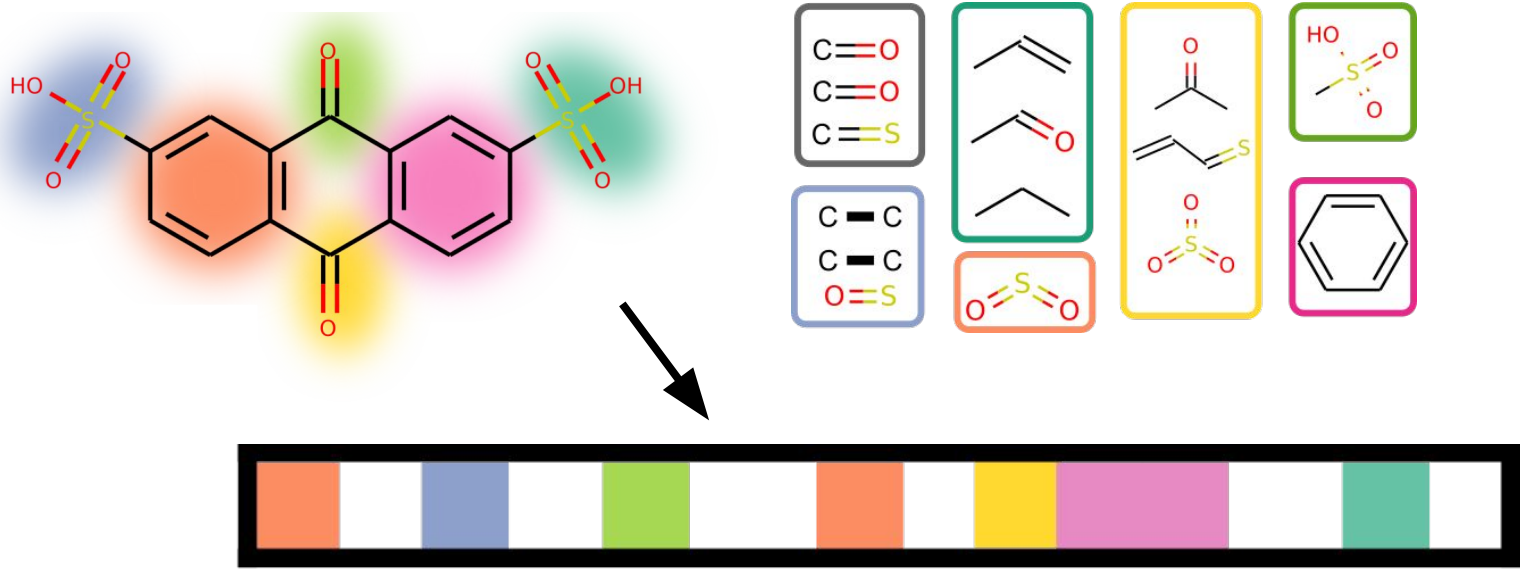


Mordred representation:
1.6k chemoinformatic kitchen sink features

Molecular Fingerprints.

An effective bag-of-fragments representation.

SMILES C1=CC2=C(C=C1S(=O)(=O)O)C(=O)C3=C(C2=O)C=CC(=C3)S(=O)(=O)O



Defining learning algorithms

Tom Mitchell in 1997 provides a concrete definition.

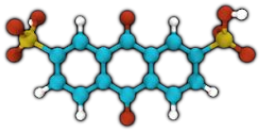
"A computer program M is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E"

- *experience E* ~ *Data*
- *performance measure P* ~ *Loss function*
- *tasks T* ~ *"Prediction problem"*
- *computer program M* ~ *Model*

Machine learning (ML) and deep learning (DL).

Learning representations of our data, optimized to a task.

Input: Molecular representation

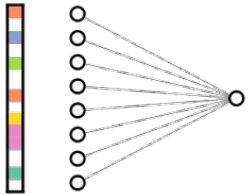


$x \longmapsto$

Traditional machine learning approach

Hand-crafted features

x'



Predictive model: e.g. Generalized linear model

$$y = f(w \cdot x' + b)$$

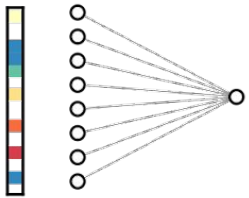
Requires more data to generalize

$x \longmapsto$

Parametrized, optimizable transformations

Deep learning approach

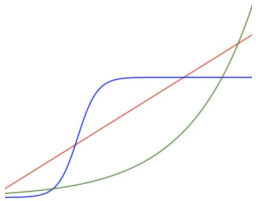
z



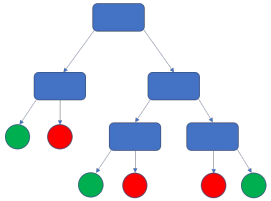
Learned representation

$$y = f(w \cdot z + b)$$

Some broad but non-exhaustive classes of "ML" learning algorithms



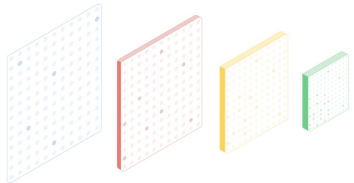
Generalized Linear Models



Decision Trees

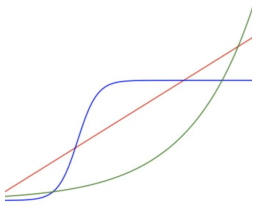
$$\|x_i - x_j\|^2$$

Kernel Machines
(GPs, Kernel regression)

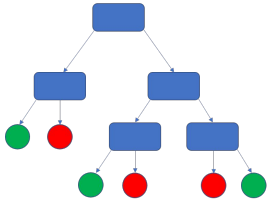


Neural Nets

Some broad but non-exhaustive classes of "ML" learning algorithms



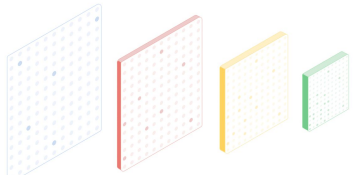
Generalized Linear Models



Decision Trees

$$\|x_i - x_j\|^2$$

Kernel Machines
(GPs, Kernel regression)



Neural Nets

Intuition behind Linear transformations

They transform data from one vector space to another

$$W \cdot x$$

x Vector

W Linear Transformation

- If x has dim 50 and W projects to dimension 100, what is the shape of W?

Generalized Linear models in equations

Linear transformations warped to a prediction target

$$Link(W \cdot x) = y$$

x Input features

y Output / target

W Linear Transformation or Weight Matrix

- Who is Link(x)?
- What is a linear transformation?
- If x has dim 50 and y dim 10, how many parameters does W have?

Linear regression in context

$$W \cdot x = y$$

- *experience E ?*
- *performance measure P ?*
- *tasks T ?*
- *computer program M ?*

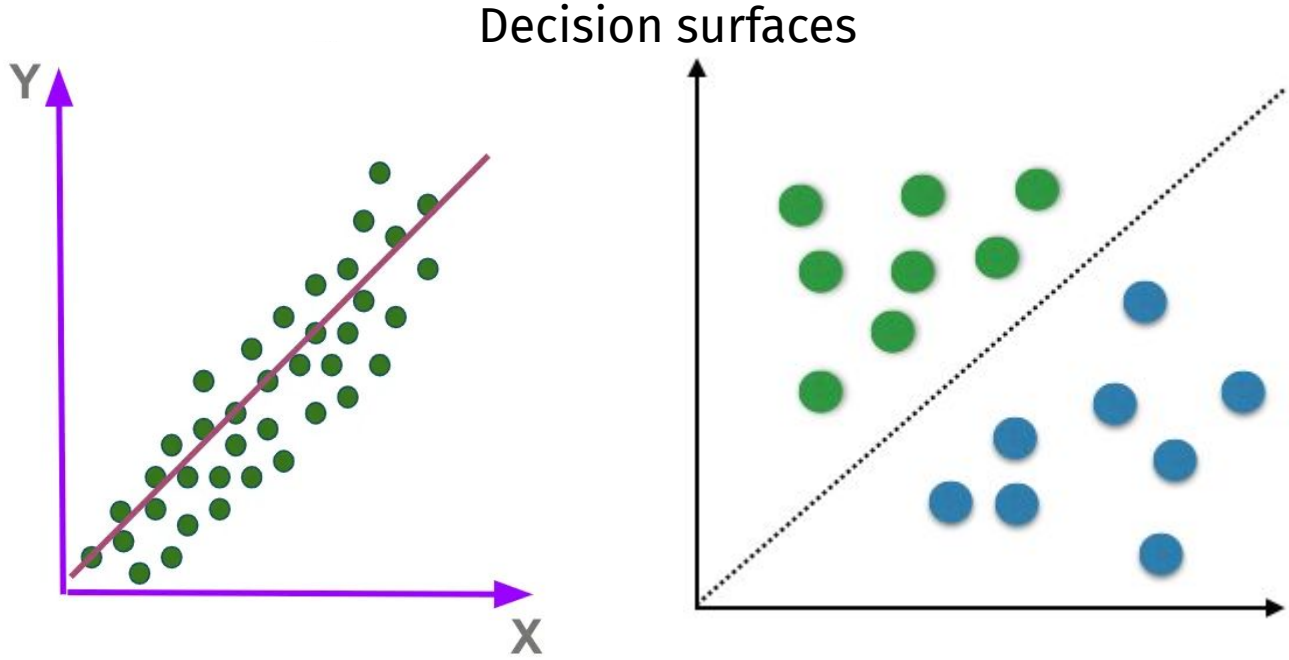
Linear regression in context

$$W \cdot x = y$$

- *E (x and y)*
- *P ? mean squared error*
- *T ? Predict y from x*
- *M ? Linear model (W)*

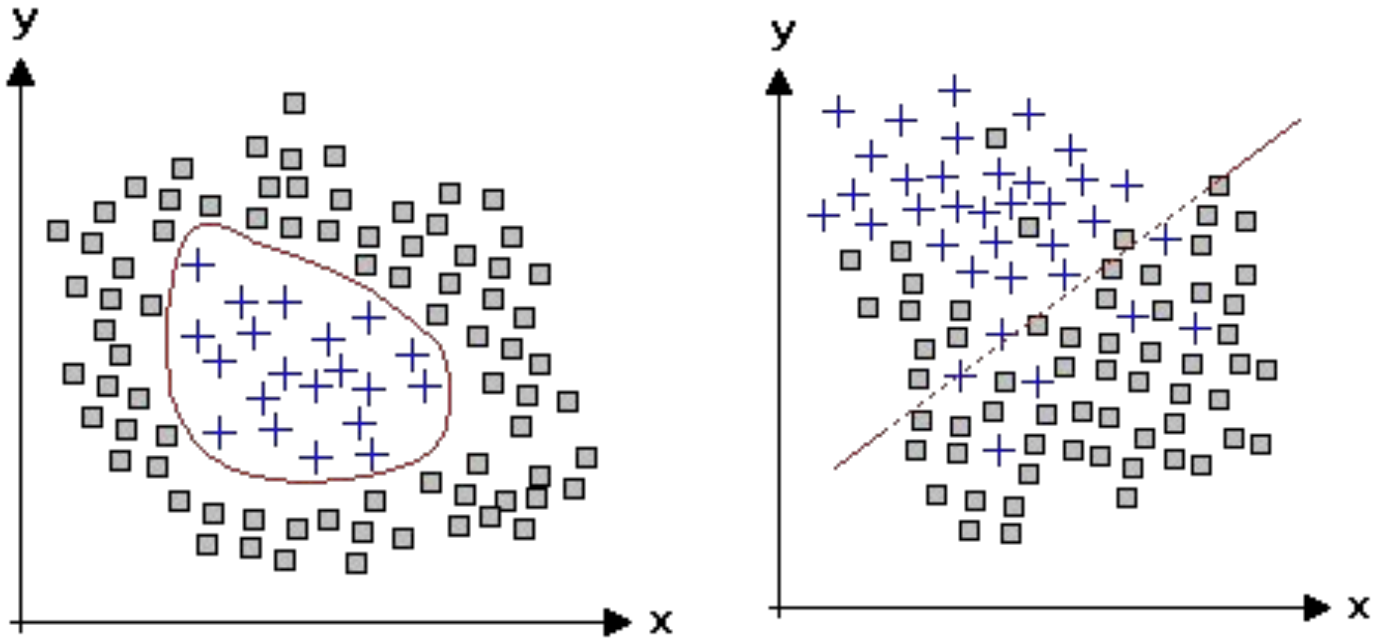
How does a linear model make a prediction?

By either mapping to a line or separating data by a line



What can we do when the data cannot be separated by a line?

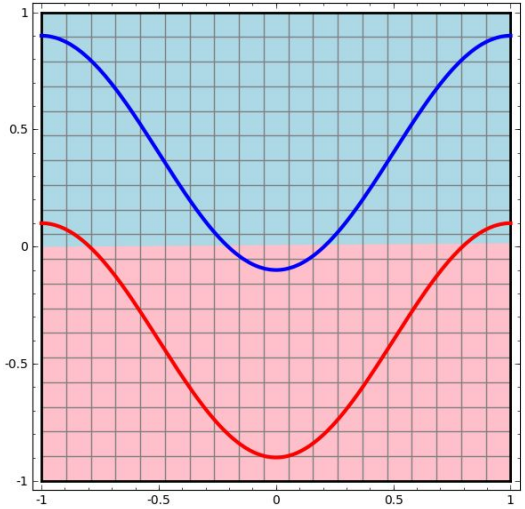
We must resort to different decision surfaces

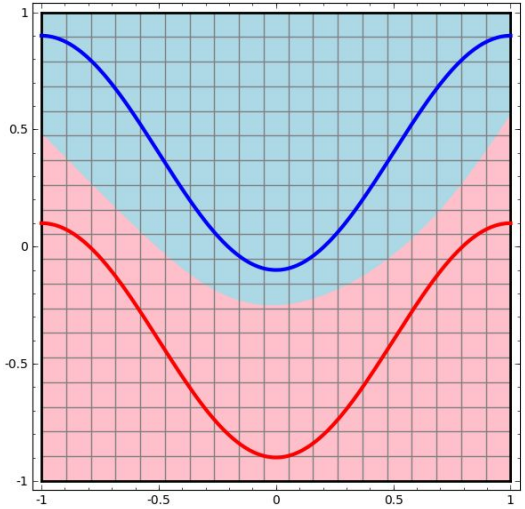


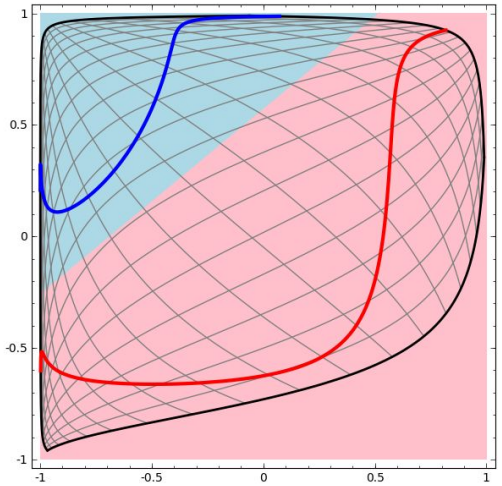
http://www.statistics4u.com/fundstat_eng/cc_data_structure.html

Intuition behind neural nets

Neural nets learn to warp space to make better predictions



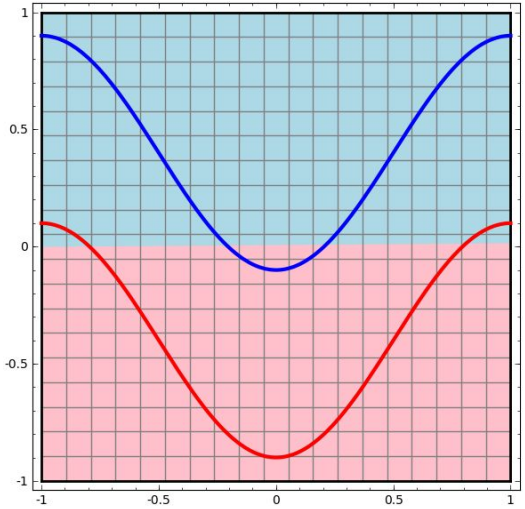


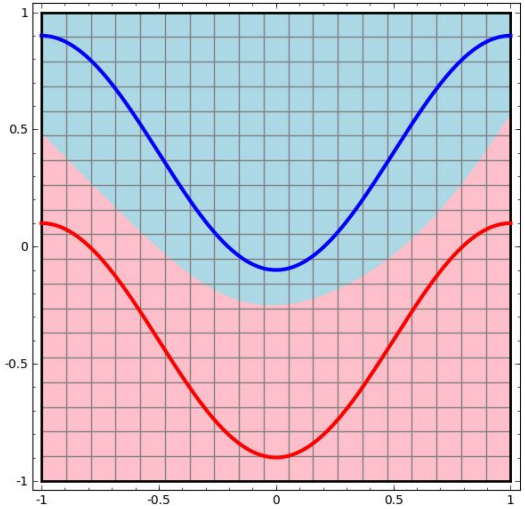


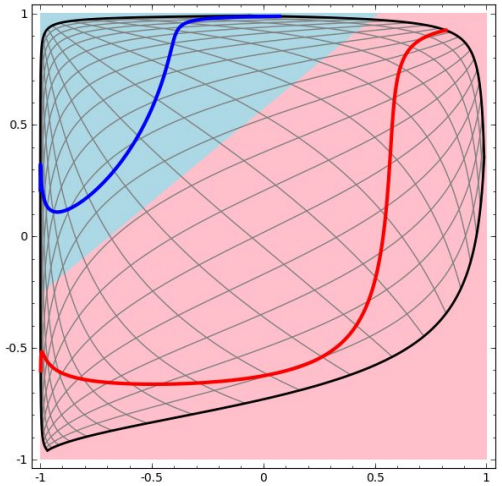
https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Intuition behind neural nets

Neural nets learn to warp space to make better predictions







https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

2-Layer Neural Net in equations

By stacking linear transforms with activation functions

$$Link(W_2 \cdot relu(W_1 \cdot x)) = y$$

x Input features

W_2, W_1 Linear Transformations or Weight Matrices

y Output / target

$relu(x) = max(0, x)$ Activation function

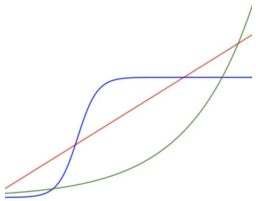
- What purpose does relu serve?
- If x has dim 50, y dim 10, and we have layer size of 50, how many parameters do we have?

Neural Network in context

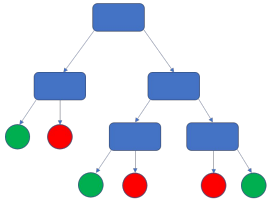
$$Link(W_2 \cdot relu(W_1 \cdot x)) = y$$

- *E (x and y)*
- *P ? mean squared error*
- *T ? Predict y from x*
- *M ? Neural net (W1, W2)*

Some broad but non-exhaustive classes of "ML" learning algorithms



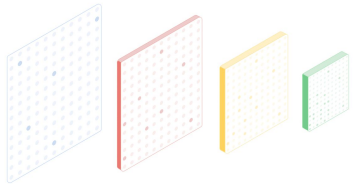
Generalized Linear Models



Decision Trees

$$\|x_i - x_j\|^2$$

Kernel Machines
(GPs, Kernel regression)

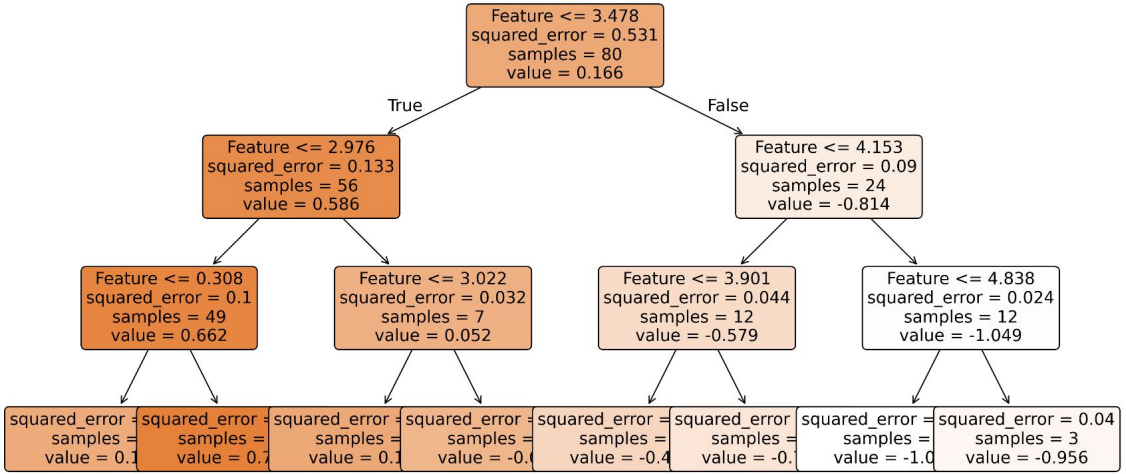


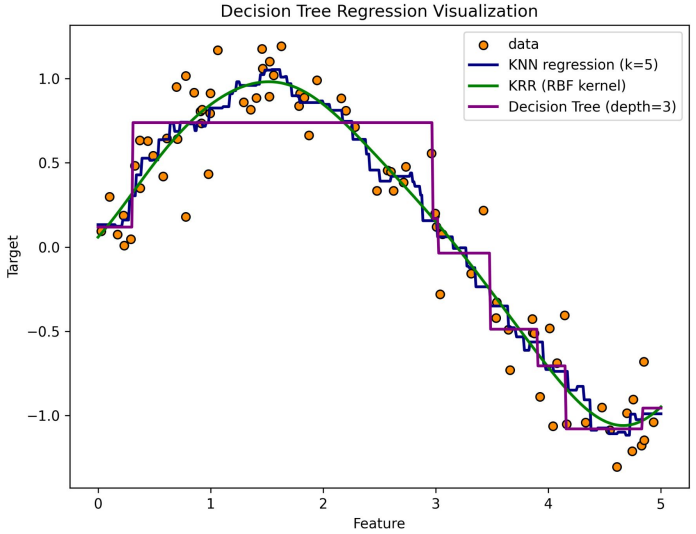
Neural Nets

Tree Based Models

Idea: recursively split feature space

- At each node: choose feature & threshold minimizing error
- Prediction = average (regression) or majority vote (classification)
- Highly interpretable!



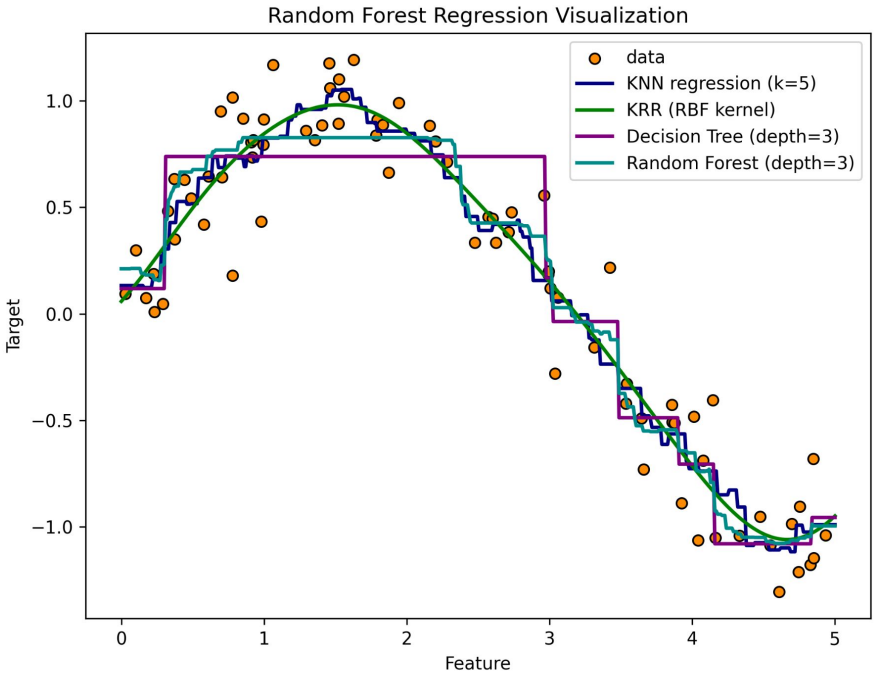


Ensemble with bagging: Random Forest

Idea: reducing variance by better sampling training set

E.g., ensemble of decision trees

- trained on random subsets of data & features
  - Bootstrap samples (random subset with replacement)
- Prediction is the average prediction of all trees in the forest
  - Averaging reduces variance

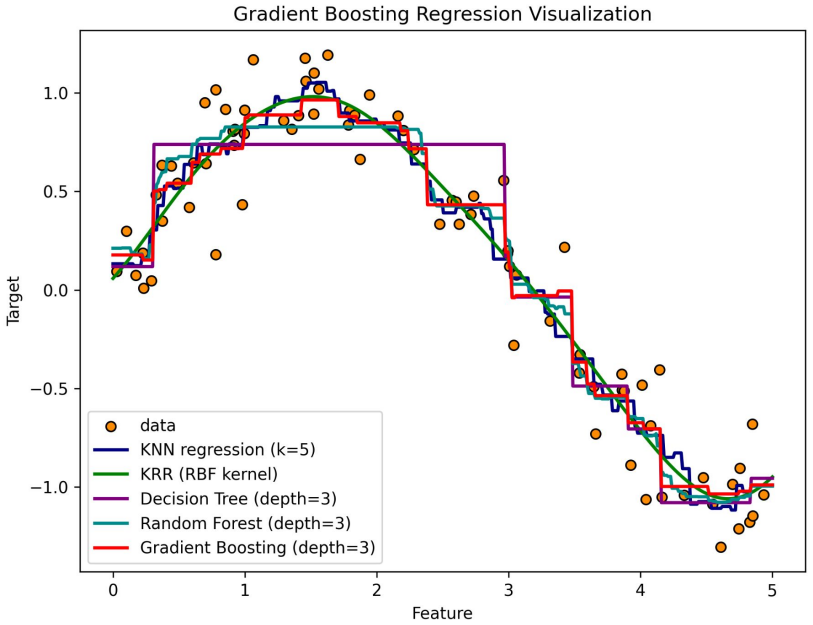


Ensemble with boosting: Gradient Boosting

Idea: Building a strong model by sequentially adding weak learners $h_t(x)$

- Each new learner is trained to reduce the errors (residuals) of the current ensemble

$$\hat{y}^{(t+1)} = y^{(t)} + \eta h_t(x)$$



Gradient boosting is model-agnostic

The weak learner can be any differentiable function approximator.

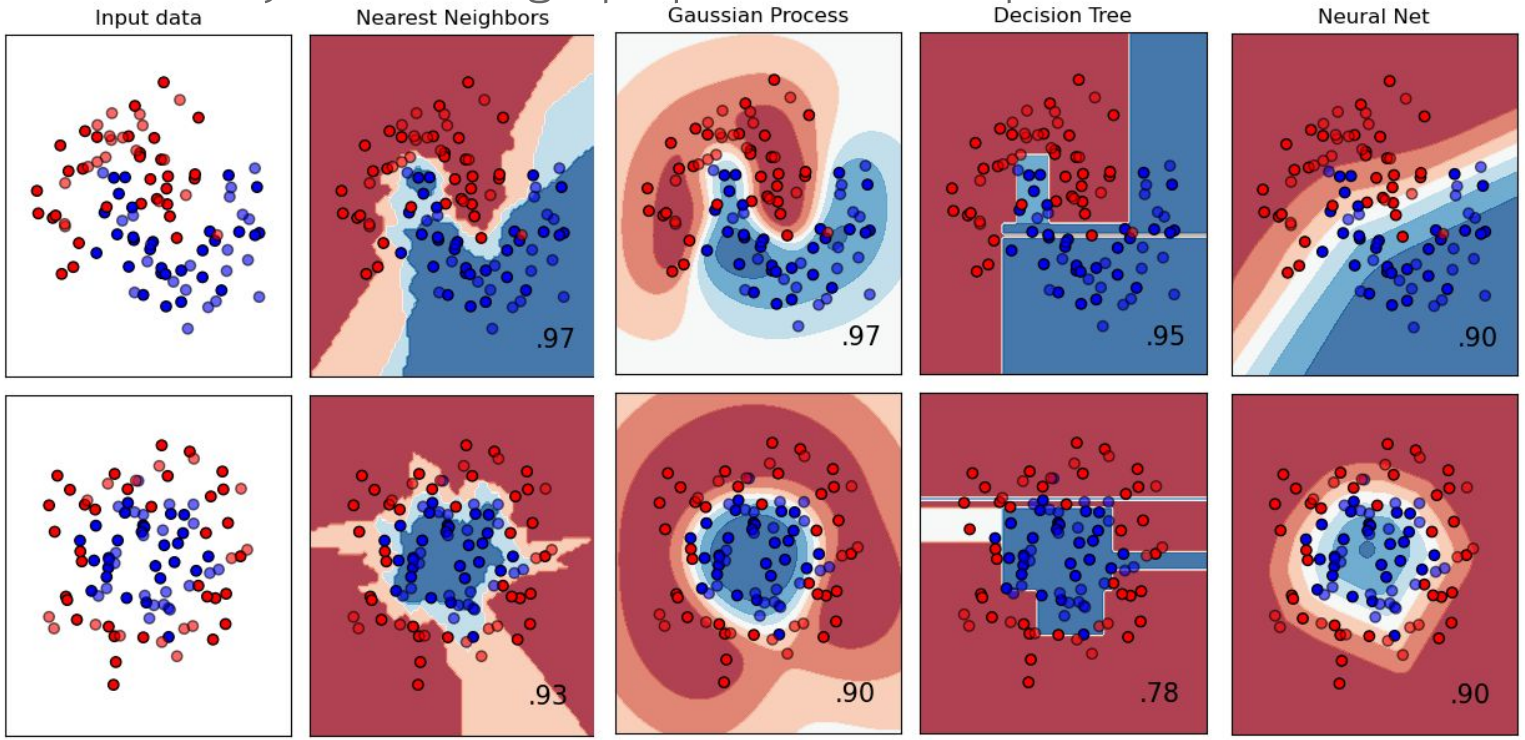
The most common learner is decision trees. Example algorithms include:

- GBDT (Gradient Boosted Decision Trees): core idea of boosting
- XGBoost (Extreme Gradient Boosting): Optimized and Regularized GBDT
- LightGBM (Light Gradient Boosting Machine): GBDT optimized for speed and memory

Gradient Boosting models are the strongest models on tabular data!

Intuition behind learning models: Decision surfaces

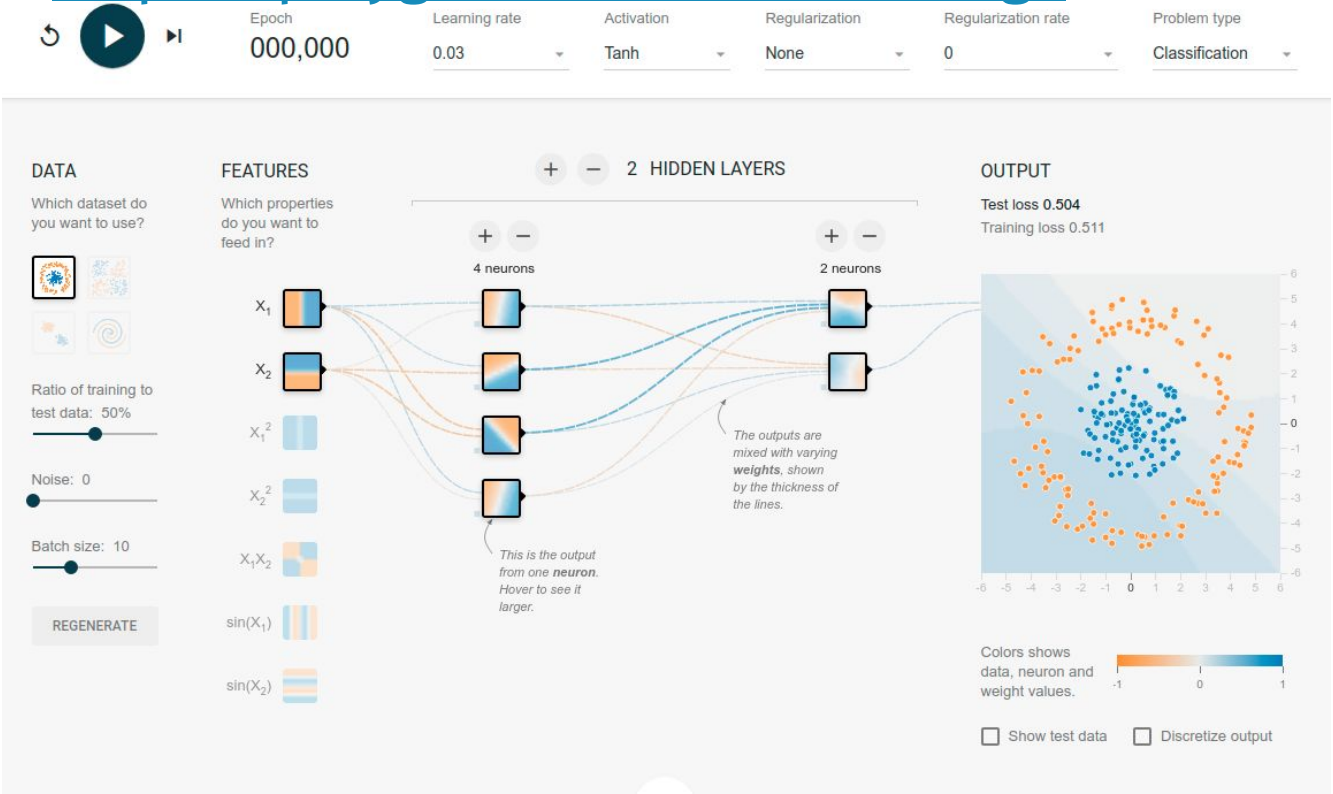
Different ways of cutting up space to make predictions



https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

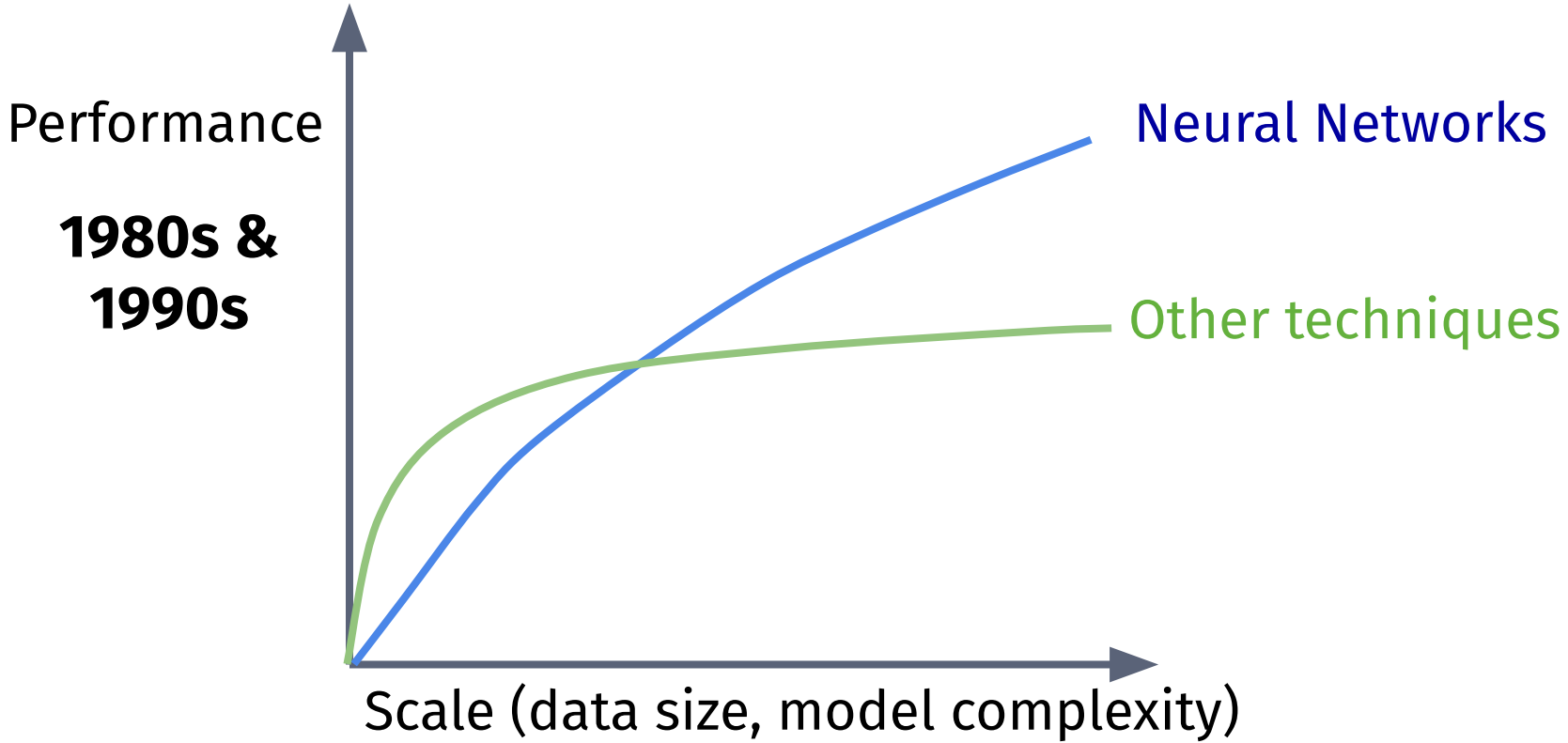
Tensorflow playground

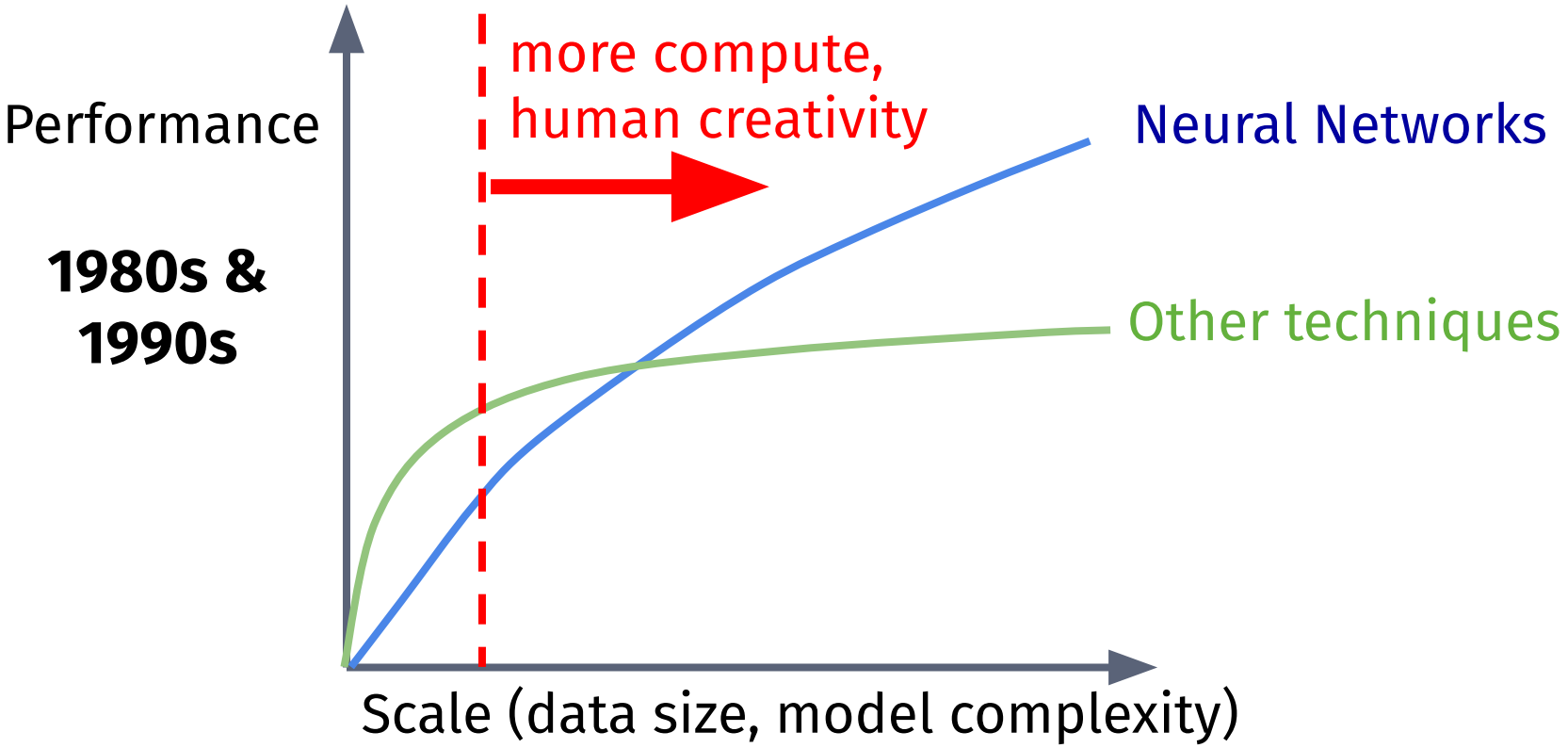
https://playground.tensorflow.org/

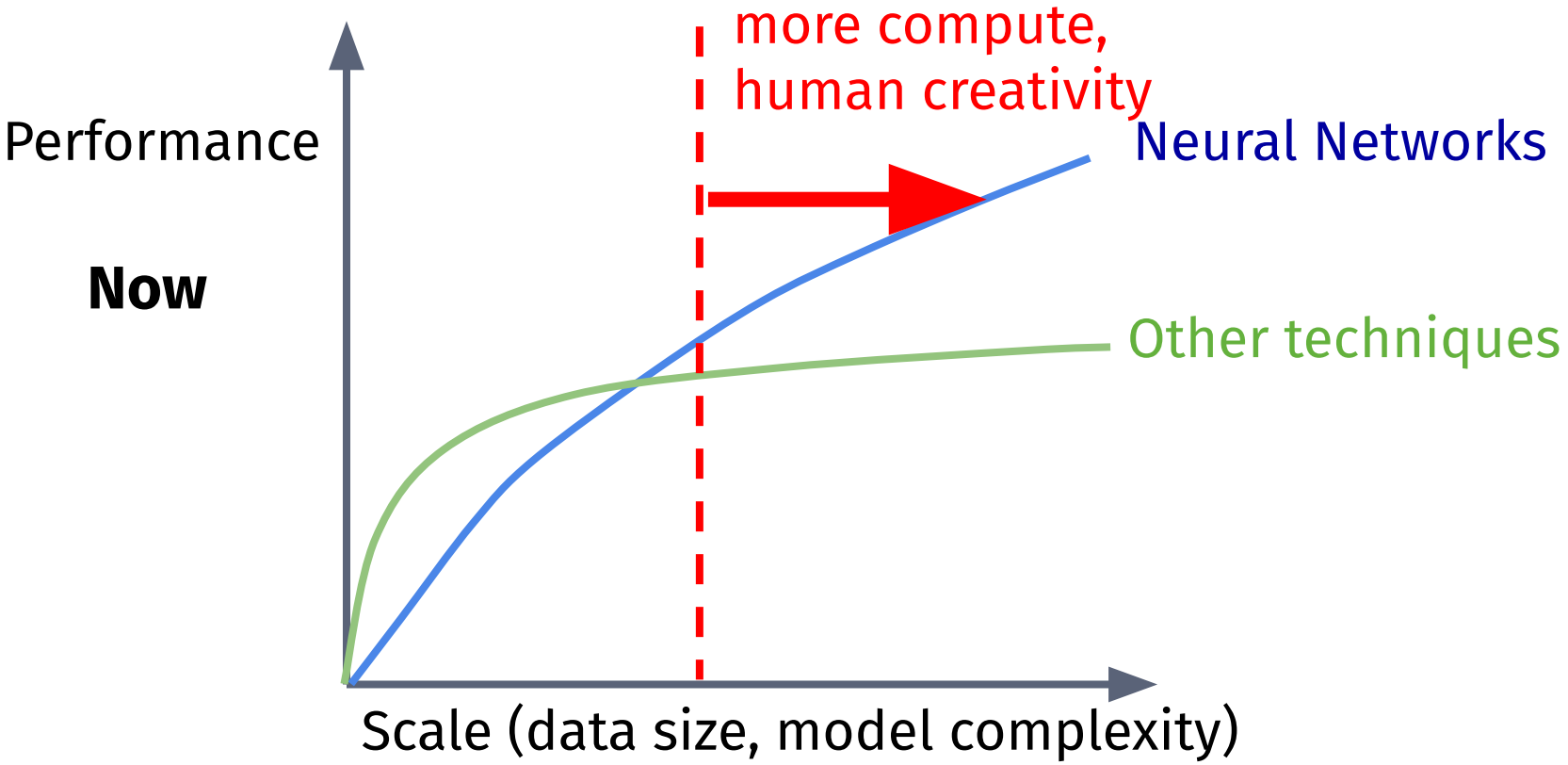


And why now?

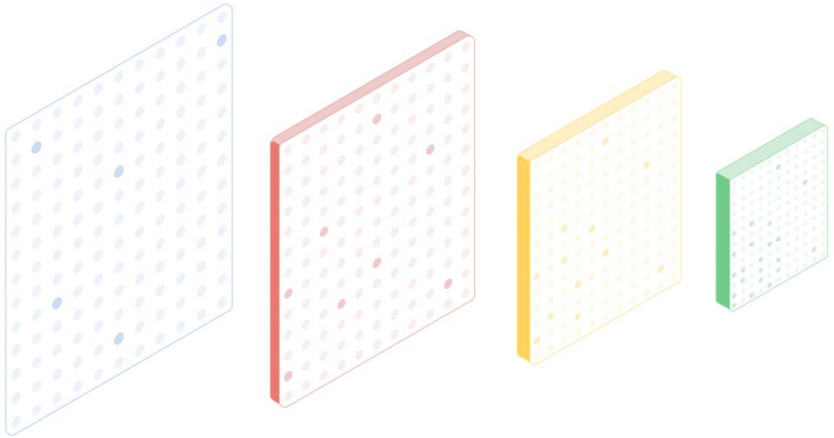
Learning algorithms across time



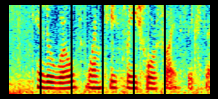

Performance
1980s &
1990s
more compute,
human creativity
Neural Networks
Other techniques
Scale (data size, model complexity)


more compute,
human creativity
Performance
Now
Neural Networks
Other techniques
Scale (data size, model complexity)

To repeat, a neural network is ...



"learnable (**optimizable)**
transformations of data"

Input		Output
	PIXELS	“lion”
	AUDIO	“Is it cold outside?”
“How are you?”	TEXT	“你好，你好吗？”
	PIXELS	“A blue and yellow train moving on rails”

Encoding and Decoding data / representations

A useful framework to think about data transformations

$$x \xrightarrow{Encode} z \xrightarrow{Decode} x'$$

x

z

x'

- Are Tensors, typically
  - rank-0 (scalars)
  - rank-1 (vectors)
  - rank-2 (matrices)
  - rank-3
- NestedTensors!

Nested tensor for a graph:

```
{
    nodes: tensor,
    edges: tensor,
    adjacency: tensor
}
```

A basic example of encode and decode with PCA

We encode and decode data into a reduced dimensionality

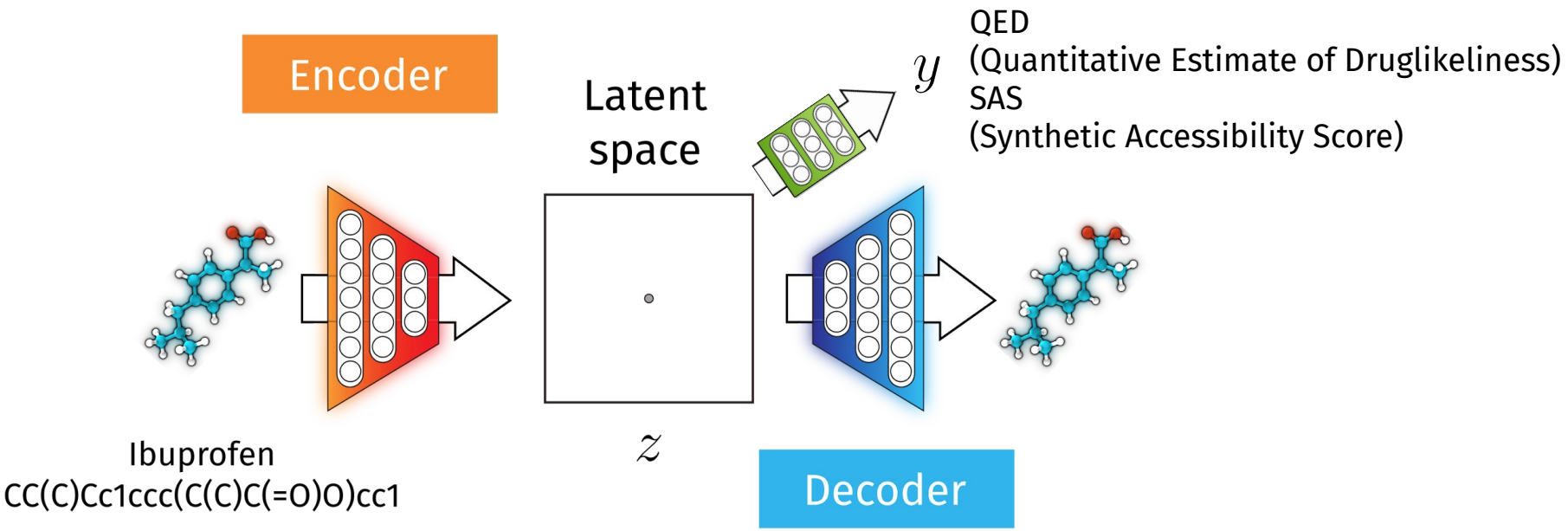
$$x \underset{Encode}{\mapsto} z \underset{Decode}{\mapsto} x\prime$$

$$Encoder(x) = W \cdot x$$

$$Decoder(z) = W^{-1} z$$

A concrete example for molecules

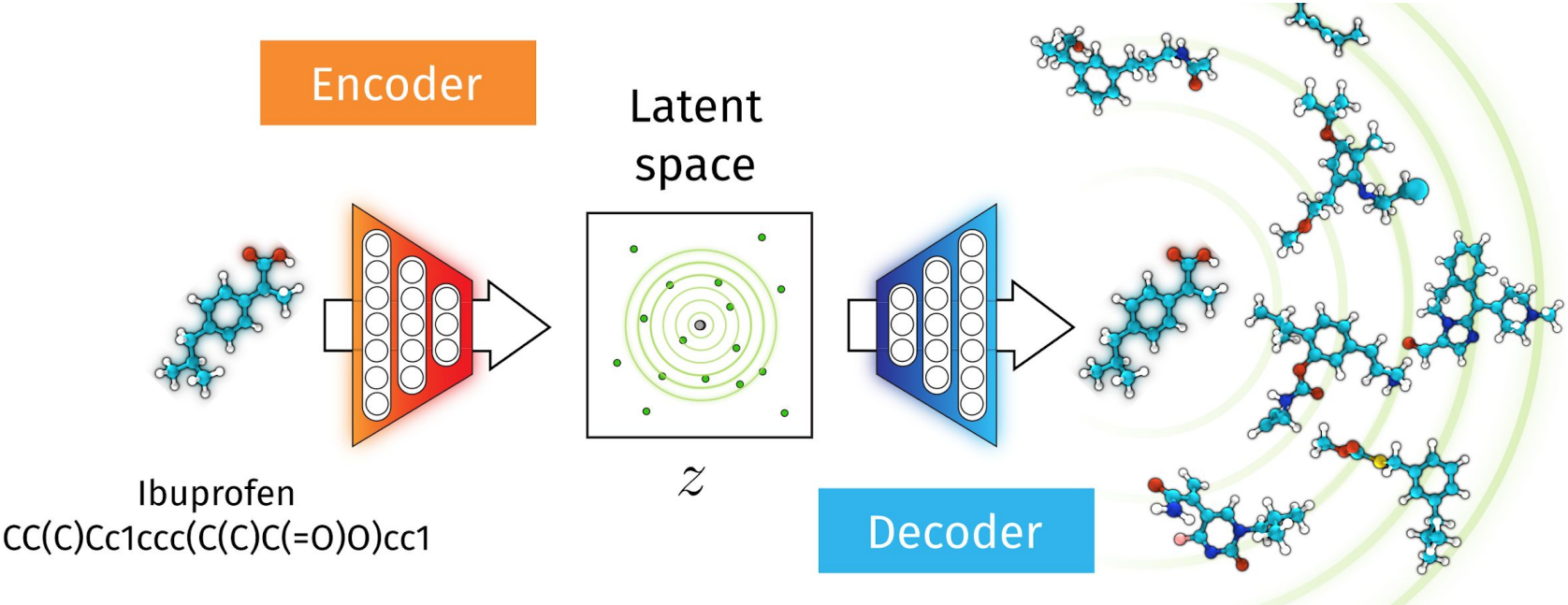
Learning a continuous and reversible representation for molecules



"Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules" ACS Cent. Sci., 2018
Rafa Bombarelli*, Jennifer N. Wei*, David Duvenaud*, José Miguel Hernandez-Lobato*, **Benjamín Sanchez-Lengeling**, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alan Aspuru-Guzik

Sampling in Latent Space

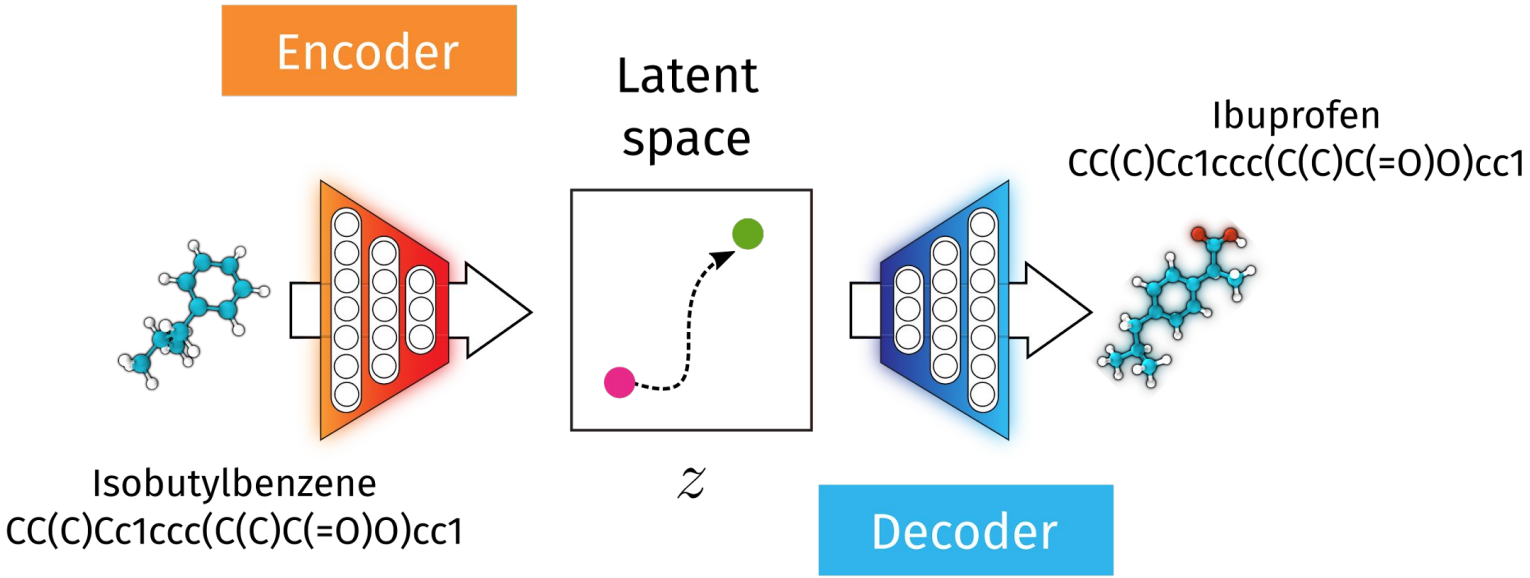
Decoded latent vectors become molecules



"Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules" ACS Cent. Sci., 2018
Rafa Bombarelli*, Jennifer N. Wei*, David Duvenaud*, José Miguel Hernandez-Lobato*, **Benjamín Sanchez-Lengeling**, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alan Aspuru-Guzik

Optimizing in Latent space

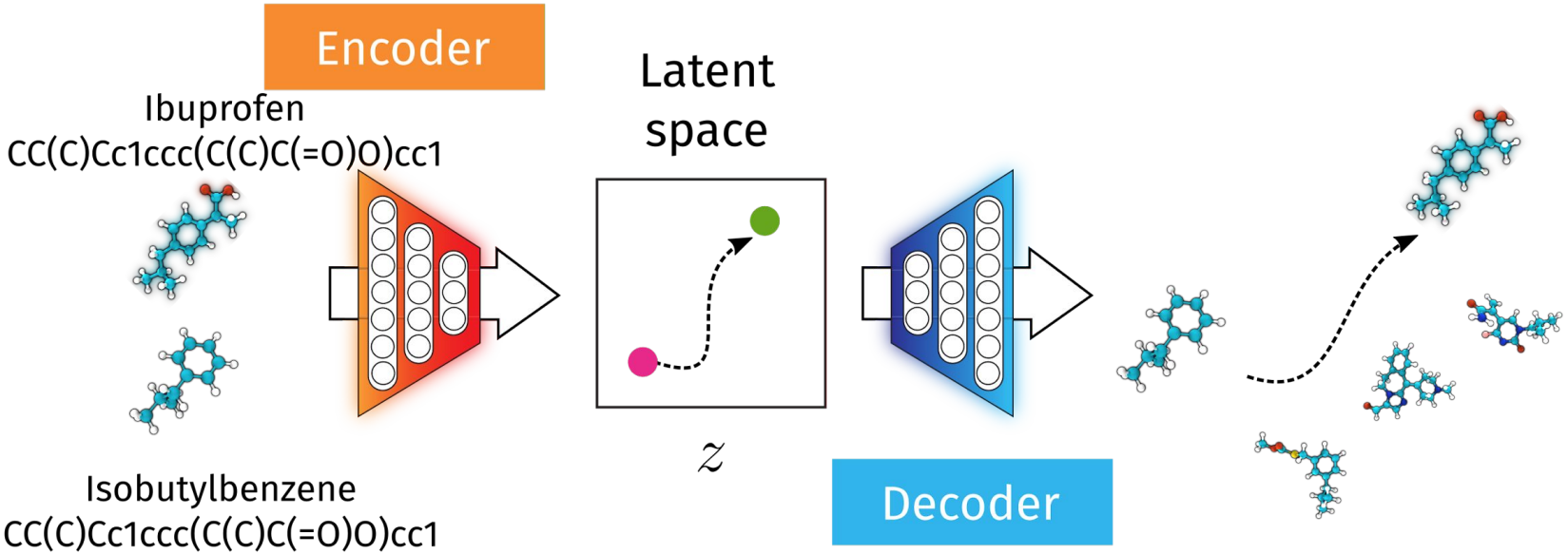
We can perform any optimization in this new latent space



"Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules" ACS Cent. Sci., 2018
Rafa Bombarelli*, Jennifer N. Wei*, David Duvenaud*, José Miguel Hernandez-Lobato*, **Benjamín Sanchez-Lengeling**, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alan Aspuru-Guzik

Interpolating in latent space

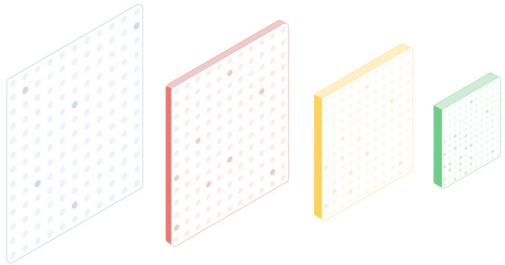
Connecting two latent vectors by smooth paths



"Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules" ACS Cent. Sci., 2018
Rafa Bombarelli*, Jennifer N. Wei*, David Duvenaud*, José Miguel Hernandez-Lobato*, **Benjamín Sanchez-Lengeling**, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alan Aspuru-Guzik

Panorama

2. The (neural) modelling choices



1. Learning representations of data (with DL)

- Predictive heads
- Inductive priors
- MLP, CNN, RNN, GNNs
- Transformers

3. Miscellaneous practical advice

Inputs and Outputs, transforming data

Inputs

Outputs

$$x \overset{Model}{\longmapsto} y$$

One example:

$$Model(x) = Decoder(Encoder(x))$$

Inputs and Outputs, transforming data

Inputs

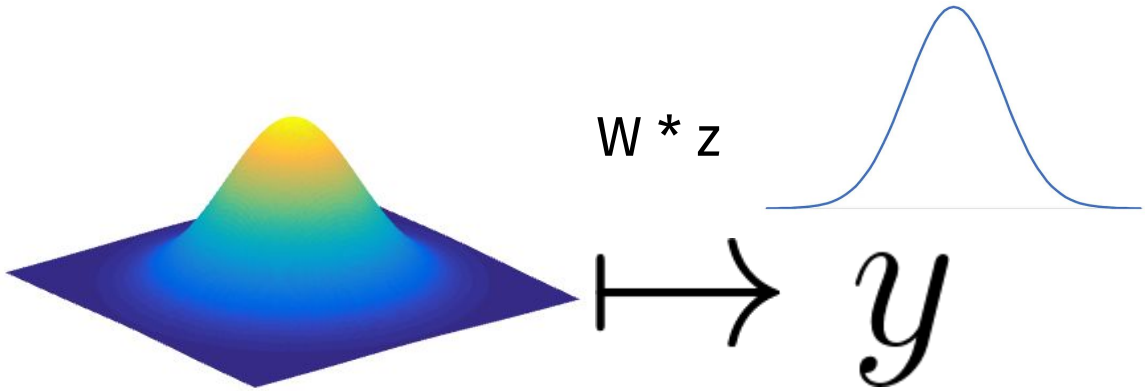
Outputs

$$x \quad \overset{Model}{\longmapsto} \quad y$$

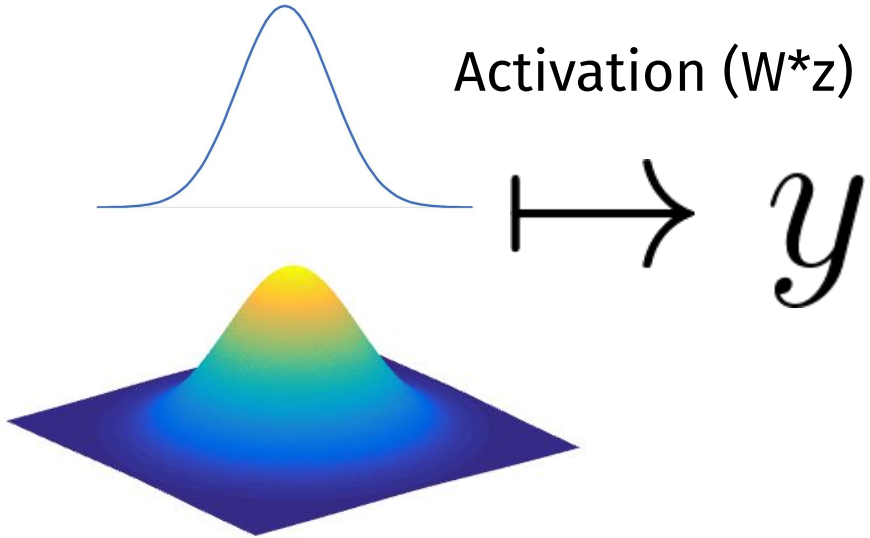
$$Model(x) = Decoder(Encoder(x))$$

$$Decoder(z) = Pred(z)$$

One useful “trick” in ML:
Transform **smoothly** Gaussian-like data to Non-Gaussian data via **Linear Transforms** + **Activations**



One useful "trick" in ML:
Transform **smoothly** Gaussian-like data to Non-Gaussian data via **Linear Transforms** + **Activations**



For prediction (activation):

- sigmoid (binary classes)
- softmax (categorical classes)
- identity (regression)

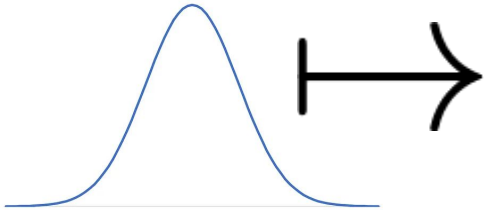
For neural network layers:

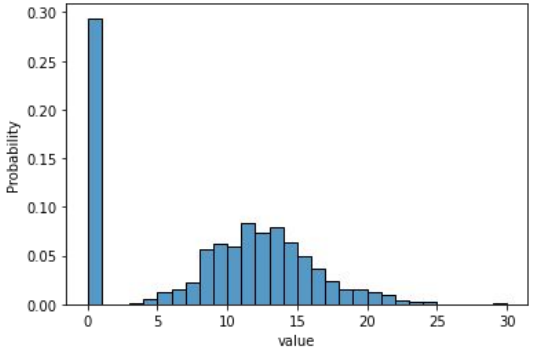
- relu (i.g. max, min)
- sum, mean, var
- softplus
- masking

Non differentiable:

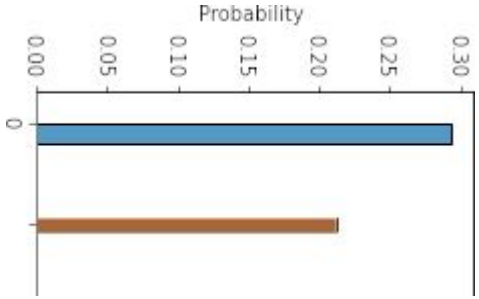
- argmax
- top-k
- binarize

Zero-inflated distribution: hurdle model

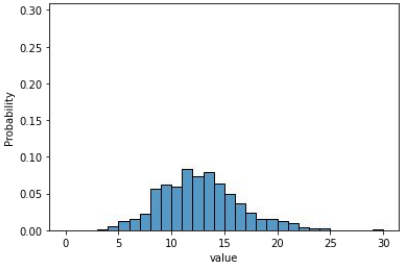




Binary classification



Regression problem



$$Pred(z) = \begin{Bmatrix} active : sigmoid(W_b \cdot z) \\ value : W_r \cdot z \end{Bmatrix}$$

Technically a multi-task problem now!

$$BCE(mask, y_{pred,binary}) + mask \cdot *MSE(y_{true}, y_{pred,reg})$$

Inputs and Outputs, transforming data

Inputs

$$x \mapsto Model \; y$$

Outputs

$$Model(x) = Decoder(Encoder(x))$$

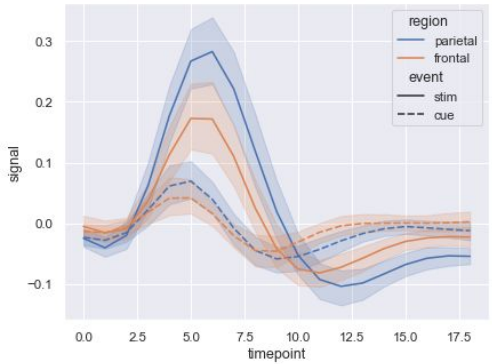
But also applicable to decoders!!!

First: understand your data, what makes it special?

Looking at important relationship we want our models to express

Inputs: time series

	subject	timepoint	event	region	signal
0	s13	18	stim	parietal	-0.017552
1	s5	14	stim	parietal	-0.080883
2	s12	18	stim	parietal	-0.081033



Inputs: text

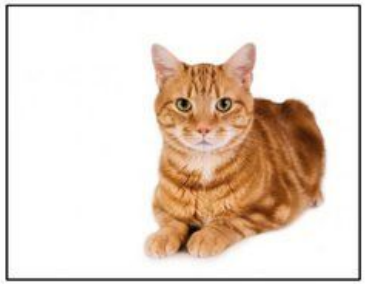
- “I work hard when I am having fun”
- “I am having fun when I work hard.”

First: understand your data, what makes it special?

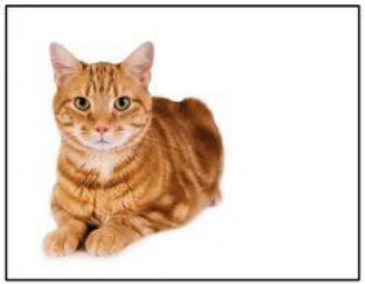
Looking at important relationship we want our models to express

Inputs: images

0-0	1-0	2-0	3-0	4-0
0-1	1-1	2-1	3-1	4-1
0-2	1-2	2-2	3-2	4-2
0-3	**1-3**	2-3	3-3	4-3
0-4	1-4	2-4	3-4	4-4



Cat



Cat

First: understand your data, what makes it special?

Looking at important relationship we want our models to express

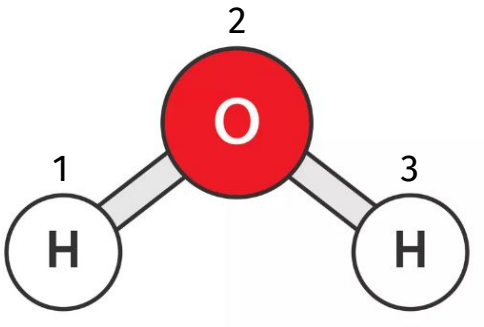
Inputs: tabular data

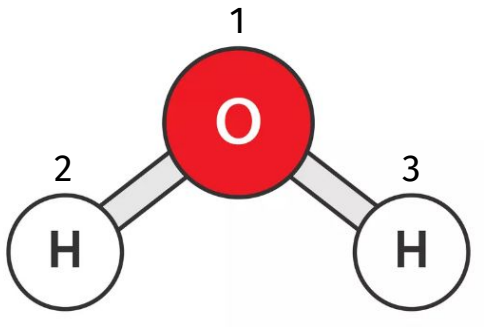
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa

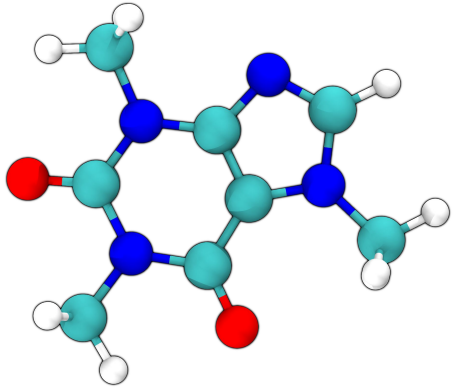
First: understand your data, what makes it special?

Looking at important relationship we want our models to express

Inputs: graph data







First: understand your data, what makes it special?

Looking at important relationship we want our models to express

Inputs: graph data

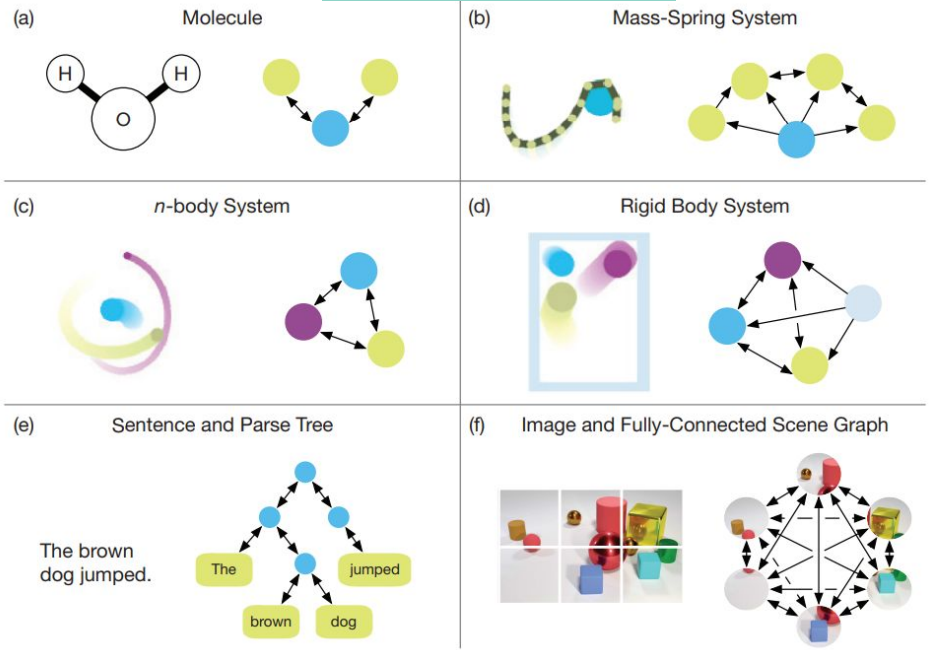


Figure from
Relational inductive biases, deep learning, and graph networks (arXiv:1806.01261v3)

Inductive priors:
Baking in positive biases to our models based on the structure of your data

Component	Entities	Relations	Rel. inductive bias	Invariance
Fully connected	Units	All-to-all	Weak	-
Convolutional	Grid elements	Local	Locality	Spatial translation
Recurrent	Timesteps	Sequential	Sequentiality	Time translation
Graph network	Nodes	Edges	Arbitrary	Node, edge permutations

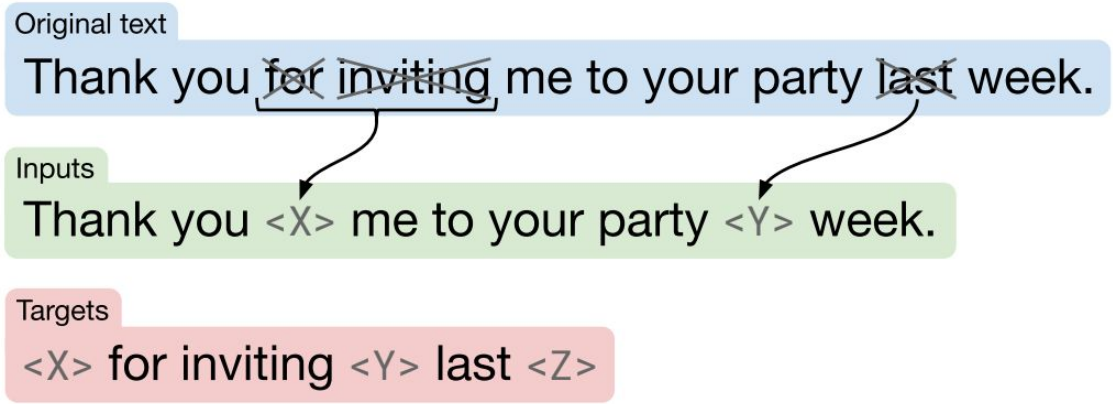
Table from
Relational inductive biases, deep learning, and graph networks (arXiv:1806.01261v3)

Transformer models

State of the art models for text data (and text-like data)

In a nutshell:

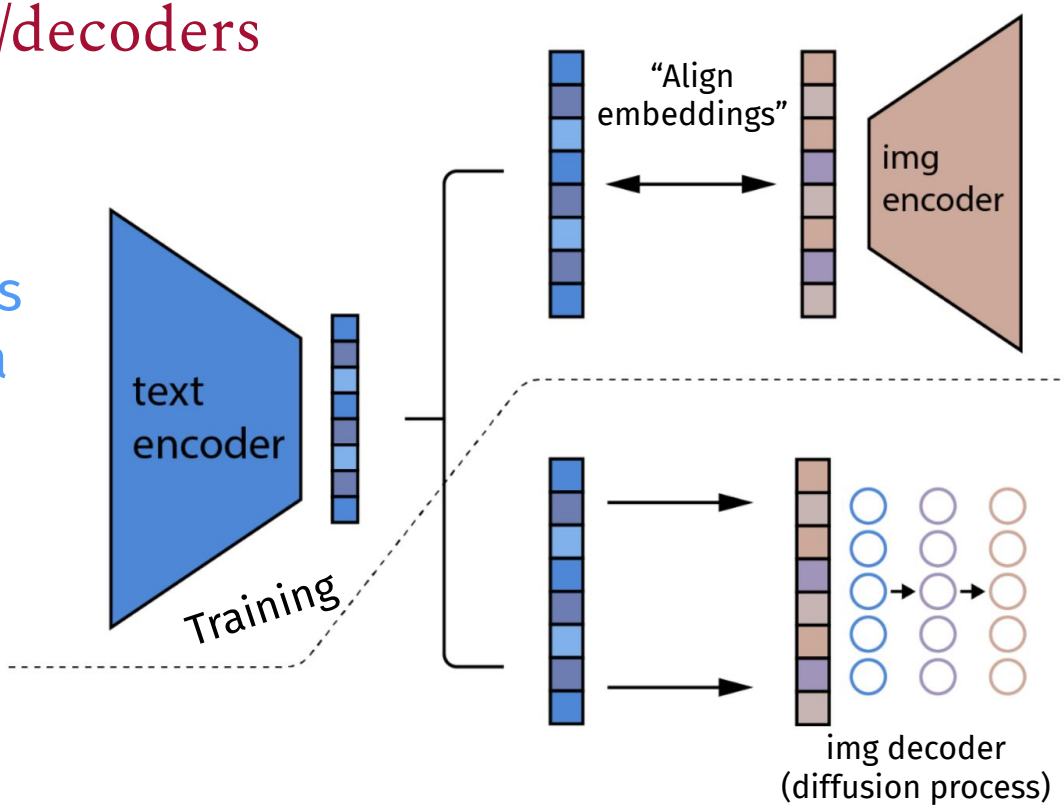
- Layers of Self-attention with positional encodings
- Massive datasets for training
- Self-supervised training
- And huge engineering efforts (multiple GPUs, TPUs)

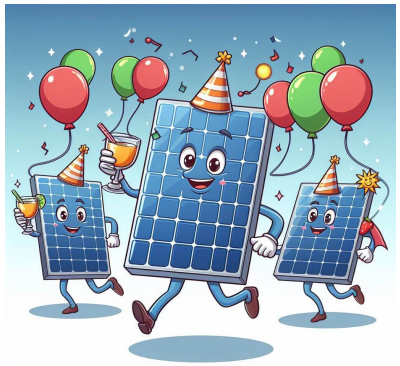


See first, Attention Is All You Need (arXiv:1706.03762)
Figure from Exploring the Limits of Transfer Learning (arxiv:1910.10683)

Text to image: crossing between different modalities of data via encoders/decoders

"solar cells going to a party, cartoon style"





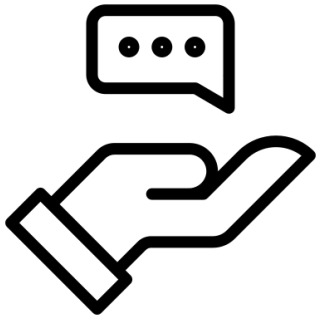
See Hierarchical Text-Conditional Image Generation with CLIP Latents (arxiv:204.06125), 2022

Panorama

1. Learning representations of data (with DL)

2. The (neural) modelling choices

- Data > Algorithms
- Human time > Computer time
- Discovery pipelines
- Feats of strengths
- Statistical significance



3. Miscellaneous practical advice

My practical philosophy for ML research

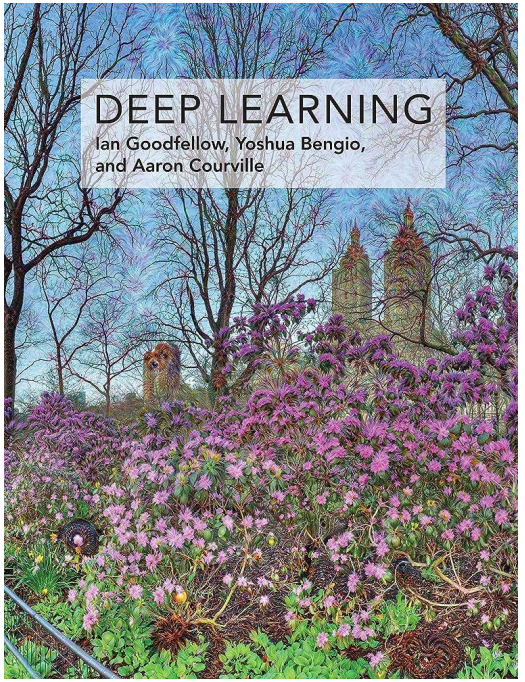
- Prioritize:
  - 1) The problem or scientific question
  - 2) Existing solutions
  - 3) The data
  - 4) The model
- Do you need ML? Perform EDA!
- Can ML solve your problem? Rule of thumb is can a human or expert predict it in less than 30 seconds?
- Collaborate with experts
- Avoiding going into the “modelling rabbit-hole”
- **Human time > computer time**, do quick experiments to learn faster

Software

Learning Deep Learning

Some practical recommendations

- Statistics and Linear Algebra
- Python
- It’s an engineering practice, find something you want to solve, and try it out.



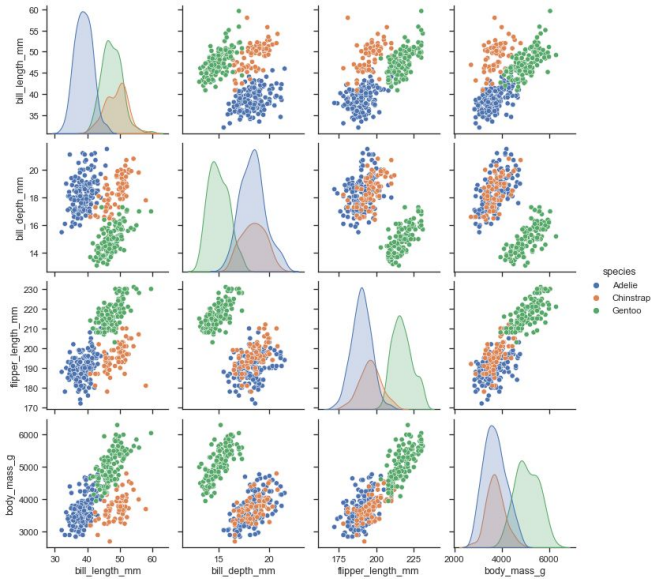
https://www.deeplearningbook.org/

Exploratory Data Analysis: EDA

Do you really need machine learning? Maybe the patterns are in the data,

Use seaborn & matplotlib & pandas

```
sns.pairplot(df, hue="species")
```



Data preprocessing

Make continuous values gaussian-like, cast discrete as numbers

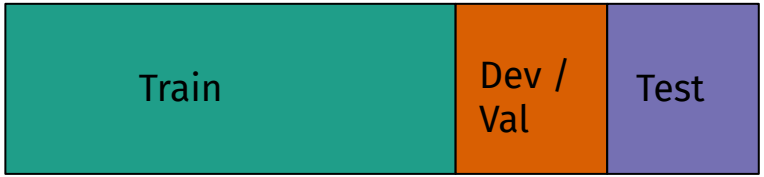
Use sklearn.preprocessing

- [min value, max value] -> [0, 1] (**MinMaxScaler)**
- [min value, max value] -> N(0,1) (Unit Gaussian) (**StandardScaler**)
  - Quantile normalization (**RobustScaler**)
  - log normalization
- False, True or Active/Non-active -> [0, 1]
- “cat”, “dog”, “car”, .. - >
  - 0, 1, 2 (**LabelEncoder**)
  - [[1,0,0], [0,1,0], [0,0,1]] (**OneHotEncoder**)
- “lemon, orange, vanilla” -> [[1,0,0,1,0, 1, ...]] (**MultiLabelBinarizer**)
- “okay”, “bad”, “good” -> See ordinal data

Building splits for training models

Building splits is an art, no solution fits all.

For model exploration or publishing results:



Train model

Make modelling decisions

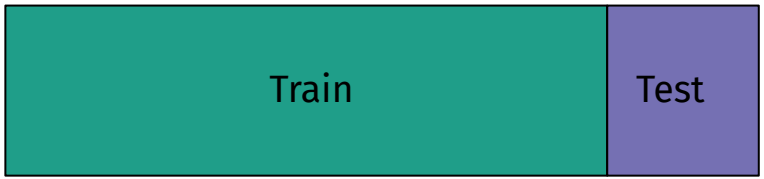
Do not touch, evaluate results

Split ratio

- 80 / 10 / 10
- 65 / 15 / 20
- 33 / 33 / 33

Also see K-fold Cross Validation!

For deploying to the real world:



Train model

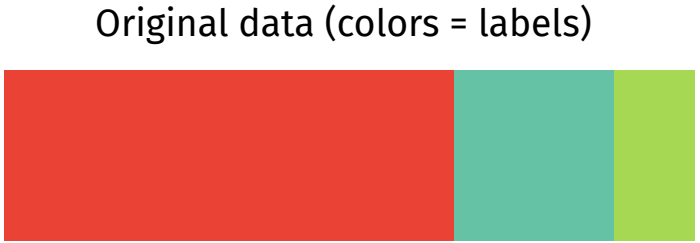
evaluate results
calibrate predictions

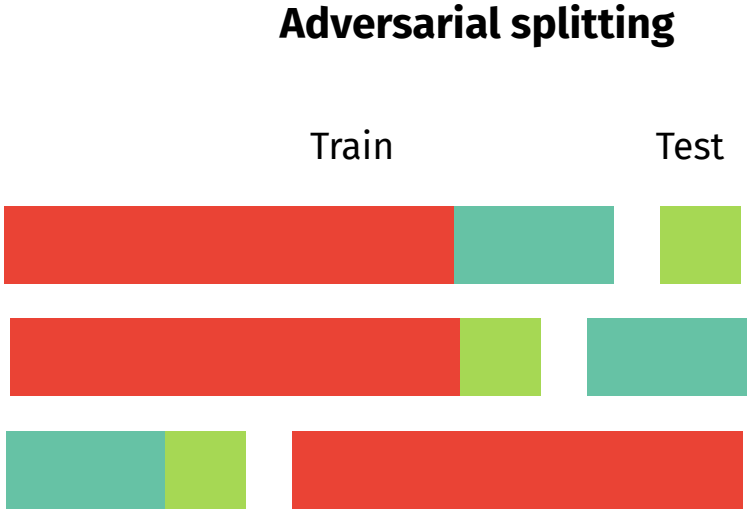
Fix modelling decisions!!

- 90 / 10
- 80 / 20
- 66 / 33

Building splits for smart evaluation

Evaluating the different ways a model can fail or succeed





Calibration and generalizability of probabilistic models on low-data chemical datasets with DIONYSUS, https://doi.org/10.1039/d2dd00146b

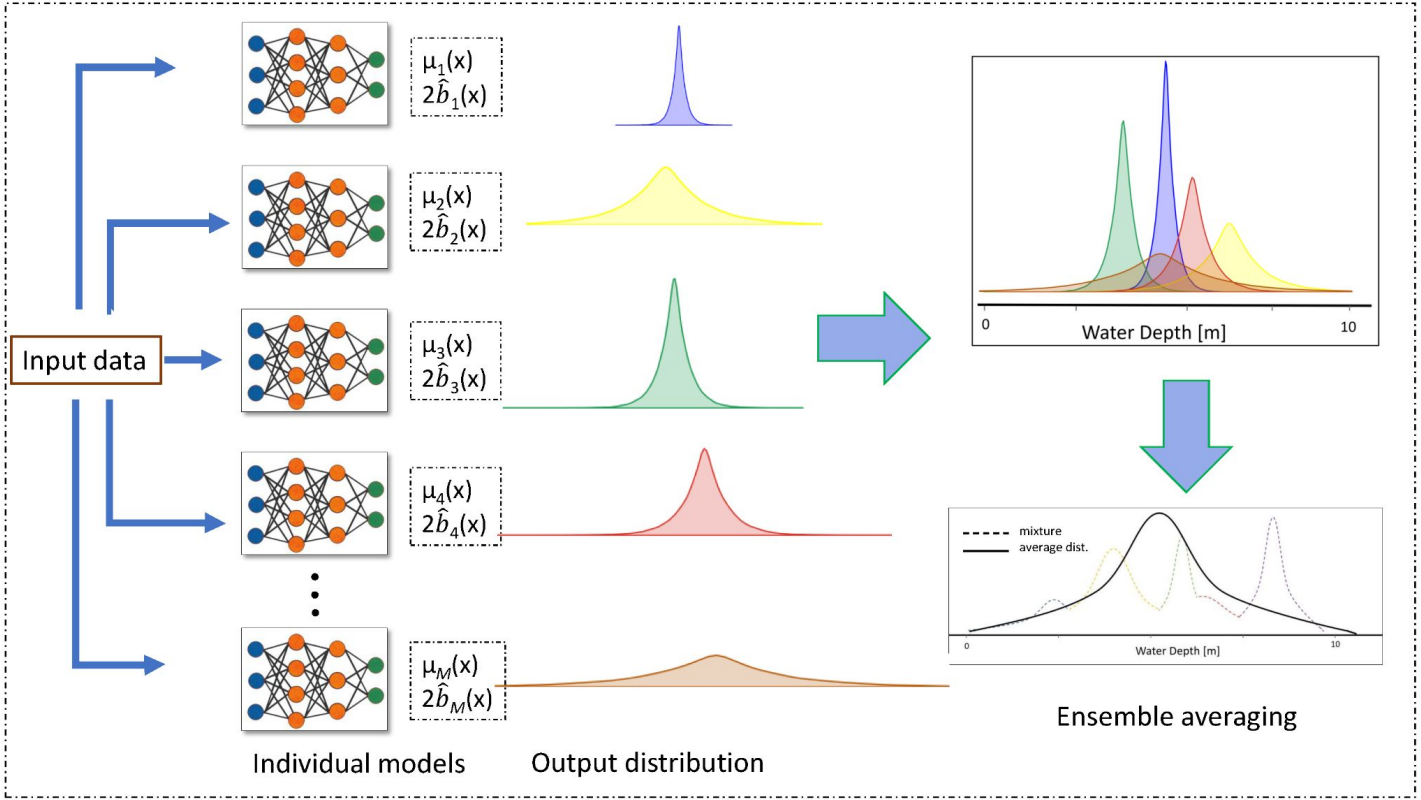
Hyperparameter selection

- Heuristics
- Random Search
- Bayesian optimization

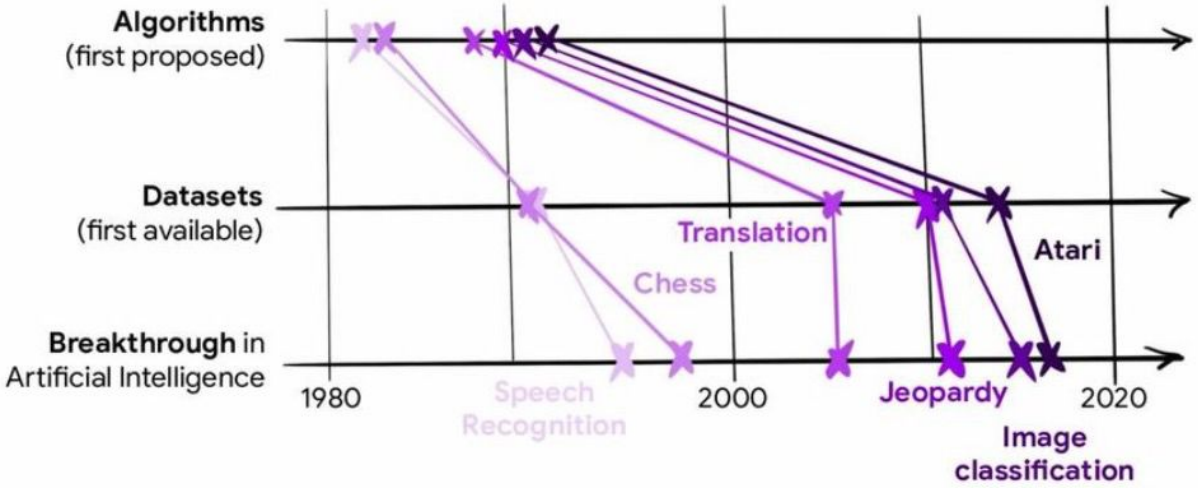
Hyperparameter space

```
{
    layer_dim: [10, 1000],
    n_layers: [1, 5],
    activation: [‘relu’, ‘softplus’],
    learning_rate: (1e-5, 1e-2),
    n_epochs: 1000
}
```

Ensembling models



Computer vision and audio processing have given birth to many datasets, and in turn, breakthroughs in AI.



Algorithms -> Data -> Breakthroughs

And their advances have brought better capabilities to:

- Search
- Generate
- Design
- Manipulate

How much data do you need for DL to be effective?

Highly context dependent, one example in molecule domains

- Uncertainty: GPs > Ensembles > BNN
- Features: Kitchen-sink features tends to be better

DIONYSUS: Calibration and generalizability of probabilistic models on low-data chemical datasets, doi: 10.1039/D2DD00146B

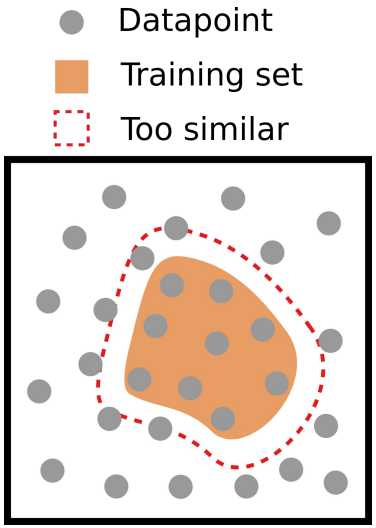
Data Centric-AI

A focus on data instead of models

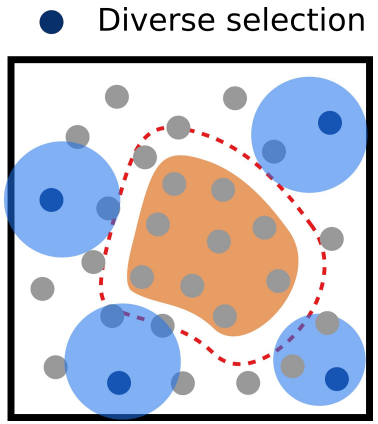
Data-centric AI is the discipline of systematically engineering the data used to build an AI system.

https://datacentricai.org/

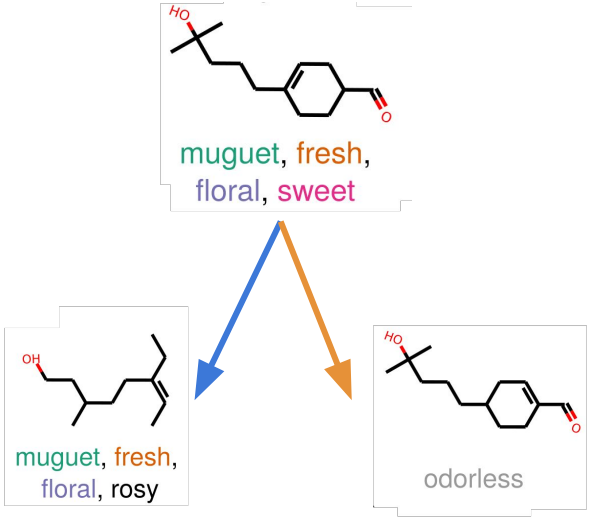
Feats of strength:
Optimizing for "surprise" in new predictions



Novelty:
Minimal distance
from training dataset



Diversity:
Representative subset via
Spherical Exclusion



Challenging:
Pick structure-activity
cliffs

Notes for practitioners

- **Identify** your central discovery **claim** (is this about models, searching algorithms, representations?)
- **Simplify** the discovery **pipeline** (Identify bottlenecks, reduce variability)
- Compare with a **realistic baseline** (How would you have done this before?)
- The **more things you test**, the **more stringent** you have to be **statistically significant** (how many hits do you need?)
- Improve your **claims via feats of strength** (novelty, diversity, prospective, surprise)

Vanilla formula to train a model

- Define model class (XGboost? GNN? CNN? Transformer)
- Preprocess data
- Define splits (train/val/test)
- Explore model space (hyperparameters):
  - Train model on train split
  - Use Early Stopping on val split
- Evaluate metrics on train/val/test
- Keep top 10 models, train on train+val
- Best model: Ensemble of 10 models
- Calibrate predictions on test

Thank you for having me!

Questions?


Calling in for backup
(slides)
POLICE

For color palette

Dark2

Set3

Cividis (continuous)

PiYG (divergent)