HOKKAIDO SUMMER INSTITUTE

ICReDD Institute for Chemical Reaction Design and Discovery HOKKAIDO UNIVERSITY

HSI 2020: Advanced Computational Chemistry

Applied Machine Learning for Chemistry

Aug 26: 10:30~12:00 (90min)
Aug 26: 13:00~14:30 (90min)

Ichigaku Takigawa

ichigaku.takigawa@riken.jp

https://itakigawa.github.io

RIKEN

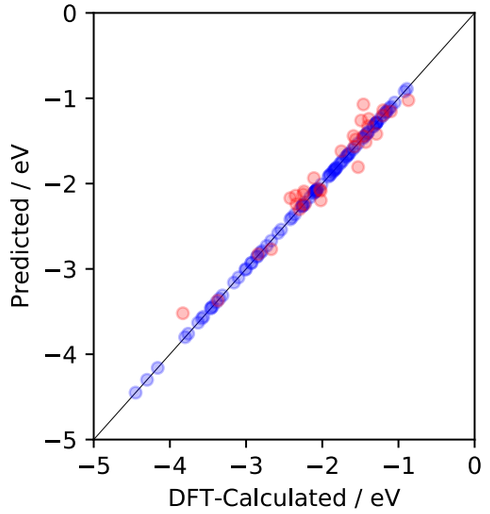
革新知能統合研究センター
Center for Advanced Intelligence Project

Quick recap: An example of entire ML process in action

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe	-0.92		-0.96	-0.97	-1.65	-1.64	-2.24		-1.87	-2.4	-3.11
Co			-1.37	-1.23		-2.12	-2.82	-2.53	-2.26		-3.56
Ni	-0.33	-1.18			-1.92	-2.03		-2.43	-2.15	-2.82	-3.39
Cu	-2.42		-2.49	-2.67	-2.89	-2.94				-3.82	-4.63
Ru	-1.11	-1.04	-1.12		-1.41		-1.88	-1.81	-1.54		-2.27
Rh	-1.42	-1.32		-1.51	-1.7	-1.73	-2.12	-1.81	-1.7	-2.18	-2.3
Pd	-1.47	-1.29	-1.29	-1.03		-1.58	-1.83	-1.68	-1.52	-1.79	
Ag	-3.75	-3.56	-3.62		-3.8		-4.03		-3.5	-3.93	-4.51
Ir	-1.78	-1.71	-1.78	-1.55		-2.14	-2.53	-2.2	-2.11	-2.6	-2.7
Pt			-1.71	-1.47	-2.13	-2.01	-2.23	-2.06	-1.96		-2.33
Au	-3.03	-2.82	-2.85			-2.89		-3.44			-3.56

gradient boosting w/ 6 descriptors

training sets (75%)
test sets (25%)

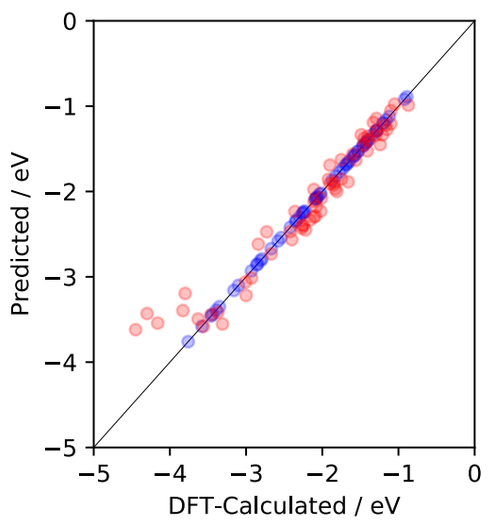


100 times mean RMSE: **0.153** / eV

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe		-0.78			-1.65	-1.64			-1.87		
Co	-1.18	-1.17	-1.37		-1.87	-2.12	-2.82		-2.26		
Ni	-0.33	-1.18		-1.17			-2.61	-2.43	-2.15	-2.82	
Cu	-2.42				-2.89	-2.94		-3.88			-4.63
Ru	-1.11	-1.04	-1.12	-1.11	-1.41			-1.81			-2.27
Rh	-1.42			-1.51			-2.12	-1.81	-1.7		
Pd		-1.29	-1.29	-1.03		-1.58	-1.83		-1.52	-1.79	
Ag				-3.68	-3.8	-3.63					-4.51
Ir						-2.14			-2.11		-2.7
Pt			-1.71	-1.47	-2.13	-2.01	-2.23	-2.06			
Au				-2.86	-3.09	-2.89		-3.44			-3.56

gradient boosting w/ 6 descriptors

training sets (50%)
test sets (50%)

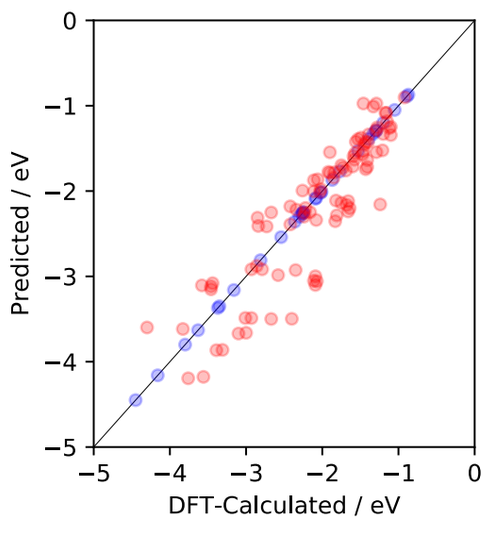


100 times mean RMSE: **0.235** / eV

	Fe	Co	Ni	Cu	Ru	Rh	Pd	Ag	Ir	Pt	Au
Fe								-2.17			-3.11
Co		-1.17	-1.37			-2.12					
Ni	-0.33	-1.18					-2.61	-2.43			
Cu	-2.42	-2.29	-2.49				-3.71				-4.63
Ru										-2.02	
Rh		-1.32				-1.73	-2.12				
Pd					-1.94		-1.83				-1.97
Ag	-3.75			-3.68							-4.51
Ir	-1.78	-1.71									-2.7
Pt					-2.13						
Au					-3.09	-2.89					

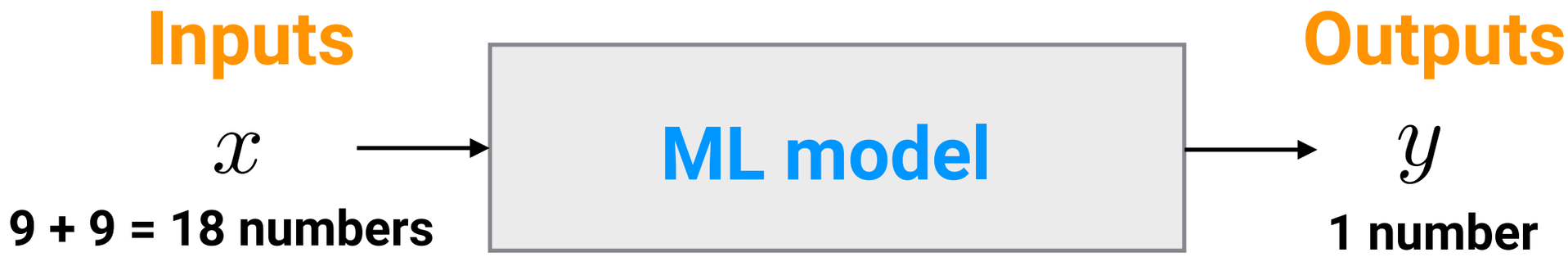
gradient boosting w/ 6 descriptors

training sets (25%)
test sets (75%)



100 times mean RMSE: **0.402** / eV

Our goal here is the following machine learning.




```
[ 8, 2.66, 26, 55.85, 4, 1.83, 7.90, 247.3, 7.87,
 11, 2.67, 29, 63.55, 4, 1.90, 7.73, 203.5, 8.96]
```

```
-1.05
```

9 features (host metal) from Table 3
+
9 features (guest metal) from Table 3

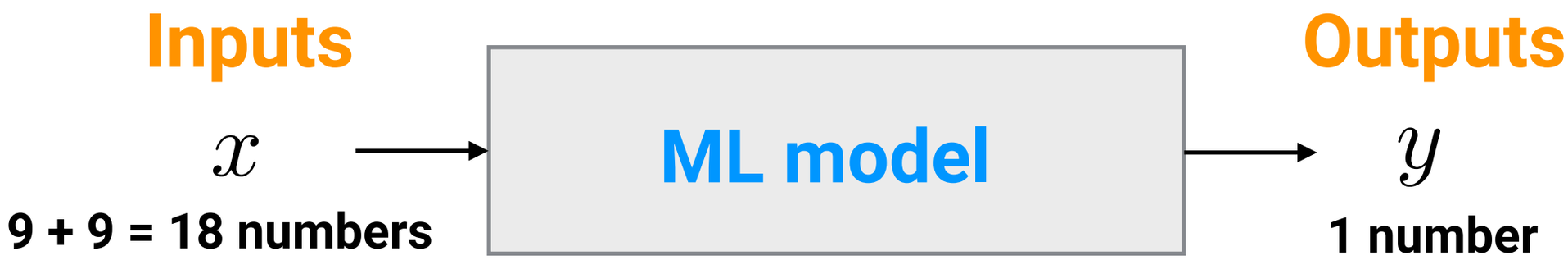
Table 3 Input features (descriptors) used for prediction of d-band centers from ref. 34[a]

Metal	G	R/Å	AN	AM/g mol^{-1}	P	EN	IE/ eV	$\Delta_{fus}H$/J g^{-1}	ρ/g cm^{-3}
Fe	8	2.66	26	55.85	4	1.83	7.90	247.3	7.87
Co	9	2.62	27	58.93	4	1.88	7.88	272.5	8.86
Ni	10	2.60	28	58.69	4	1.91	7.64	290.3	8.90
Cu	11	2.67	29	63.55	4	1.90	7.73	203.5	8.96

the value at Table 1
for the host and guest metal

	Fe	Co	Ni	Cu	Ru	Rh
Fe	-0.92	-0.87	-1.12	-1.05	-1.21	-1.46
Co	-1.16	-1.17	-1.45	-1.33	-1.41	-1.75
Ni	-1.20	-1.10	-1.29	-1.10	-1.43	-1.60
Cu	-2.11	-2.07	-2.40	-2.67	-2.09	-2.35

Ignoring all problem-specific data preparation...




```
[ 8, 2.66, 26, 55.85, 4, 1.83, 7.90, 247.3, 7.87,
 11, 2.67, 29, 63.55, 4, 1.90, 7.73, 203.5, 8.96]
```

```
-1.05
```

```
[23] from sklearn.ensemble import GradientBoostingRegressor
     model = GradientBoostingRegressor()
     model.fit(X_train, y_train)
```

"machine learning" part is only here!

https://scikit-learn.org/stable/

learn Install User Guide API Examples More Go

scikit-learn

Machine Learning in Python

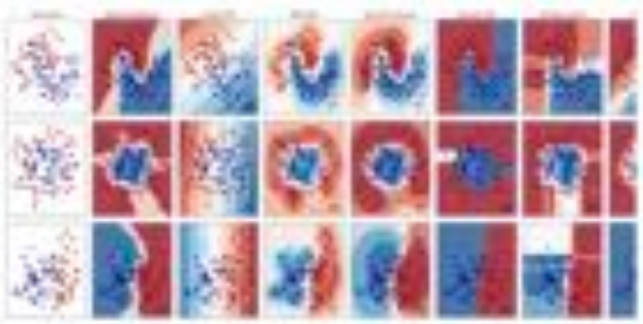
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Getting Started | Release Highlights for 0.23 | GitHub

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

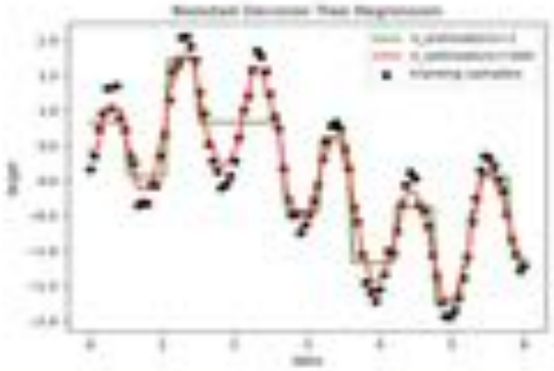


Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...



Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...

Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation,

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...

Aug 26: 10:30~12:00 (90min)

1. What is "machine learning"?
2. Why does it matter to chemists?
3. Let's try it in your browser (with no setup!)

Aug 26: 13:00~14:30 (90min)

4. Five things all beginners should know
   - "The quality of your inputs decide the quality of your output"
   - Training / validation / test data
   - Tuning hyperparameters
   - Identification and design of input variables (or "descriptors")
   - "Correlation does not imply causation"
5. Standard pipeline and deep learning
6. Current efforts and future directions

Aug 26: 10:30~12:00 (90min)

1. What is "machine learning"?
2. Why does it matter to chemists?
3. Let's try it in your browser (with no setup!)

Aug 26: 13:00~14:30 (90min)

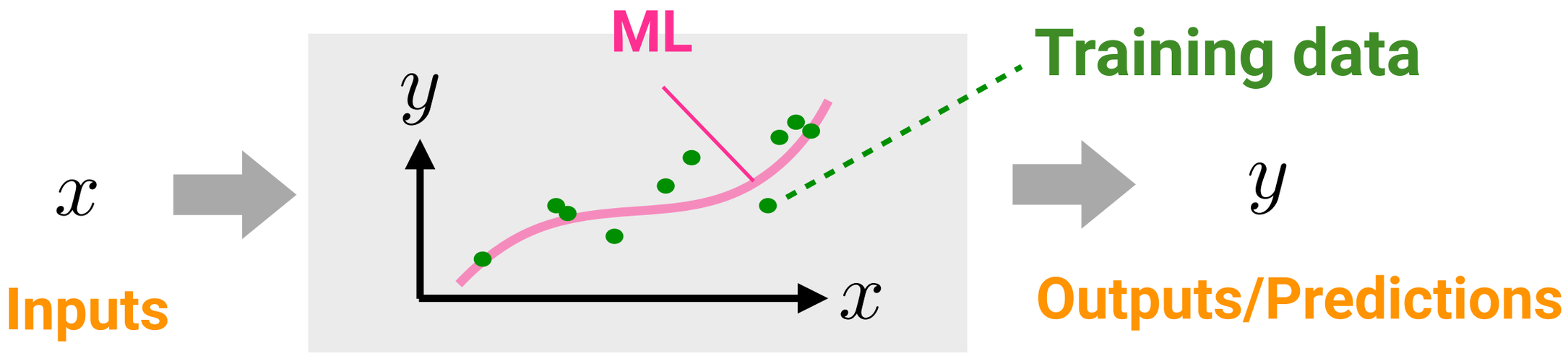
4. Five things all beginners should know
   - "The quality of your inputs decide the quality of your output"
   - Training / validation / test data
   - Tuning hyperparameters
   - Identification and design of input variables (or "descriptors")
   - "Correlation does not imply causation"
5. Standard pipeline and deep learning
6. Current efforts and future directions

Why applied ML is so challenging?

Because it requires expertise about ML and the target problem.

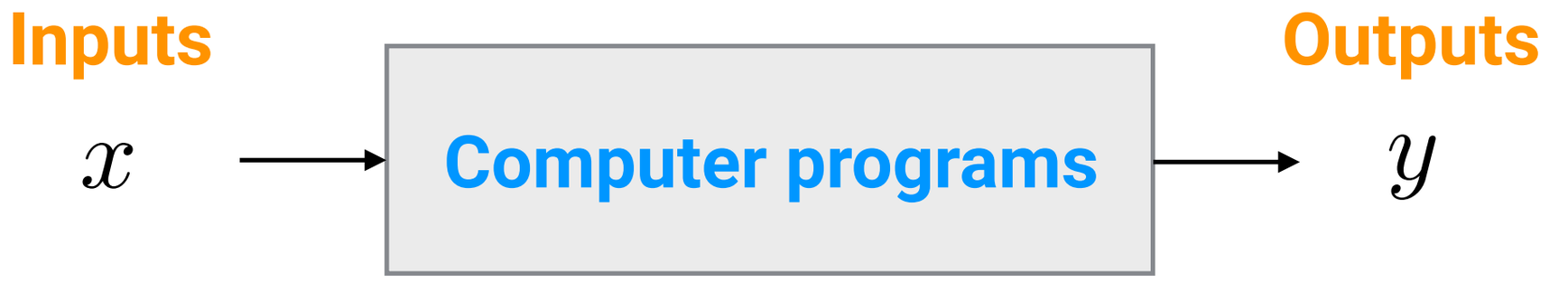
- Choose the right framing for inputs and outputs of problem
  Some unknown but coherent relationship is required between inputs and outputs.
- Collect the good training data
- Assess the quality of data (quality control)
- Choose the relevant representation for inputs
- Choose the right algorithm for ML
  ML approximate an unknown underlying mapping function from inputs to outputs.
- Choose the right algorithm configuration
- Test thoroughly whether the ML prediction works well

Remember the quality of your inputs decide the quality of your output.

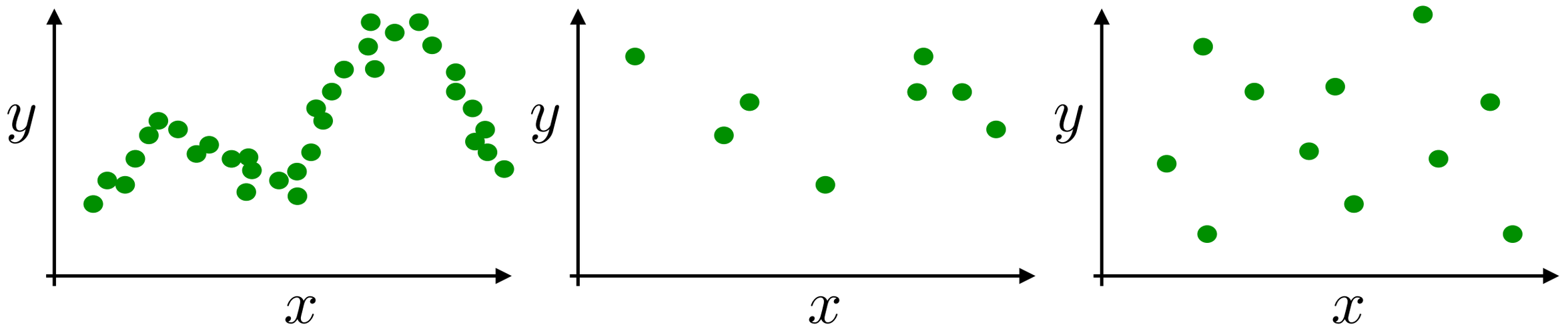


- The computer sees only the data you give it
- "ML is interpolating the training data" means that when the training data is poor-quality or very biased, so is the ML prediction no matter what state-of-the-art ML method is used.
- In other words, **"data-driven" means that prediction is defined by the data itselves in the first place.** All other factors, including what ML models should be used, are secondary issues.
- A first key to success is collecting good training data.

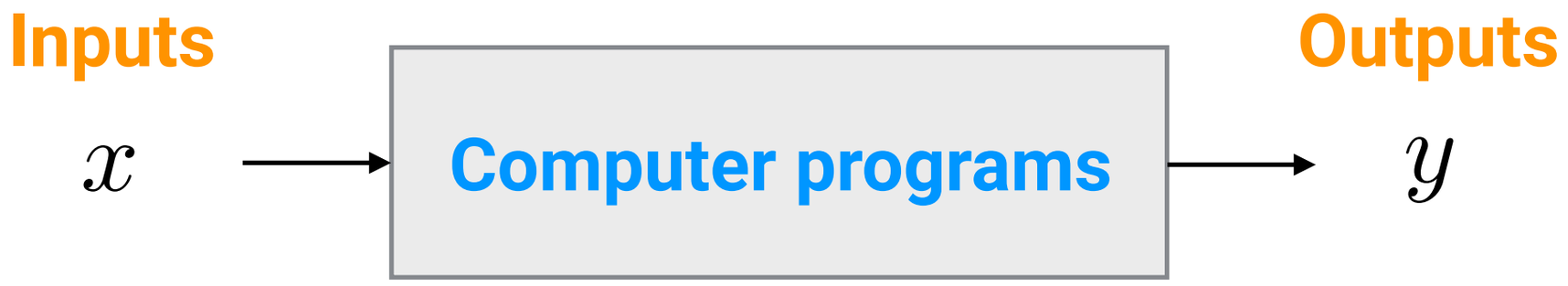
“Garbage in, garbage out”



Training data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$

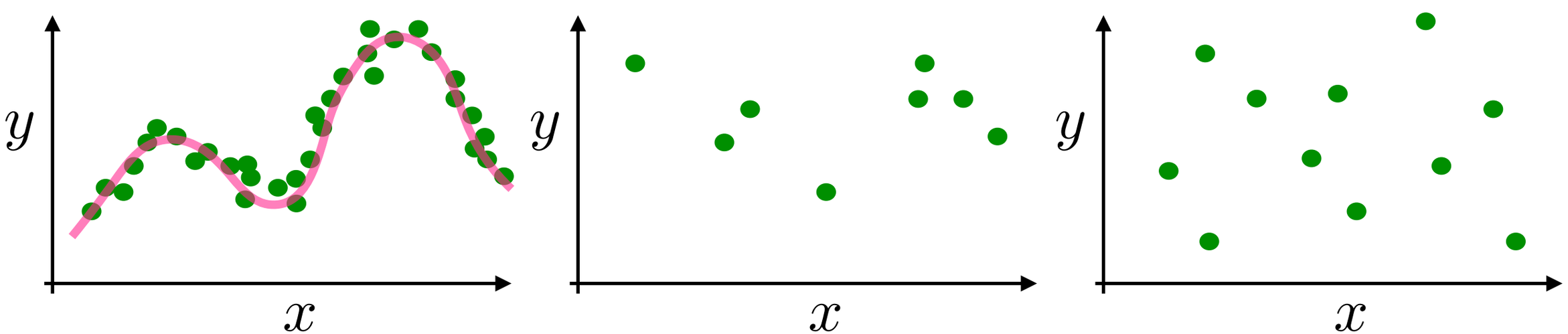


“Garbage in, garbage out”

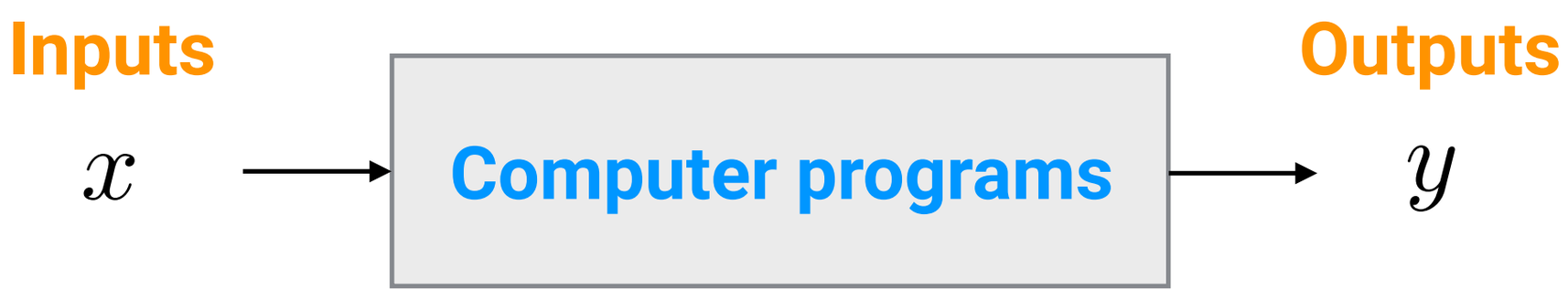


Training data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$

- **The curve seem good representative of the training data?**
- **Sufficient number of data**



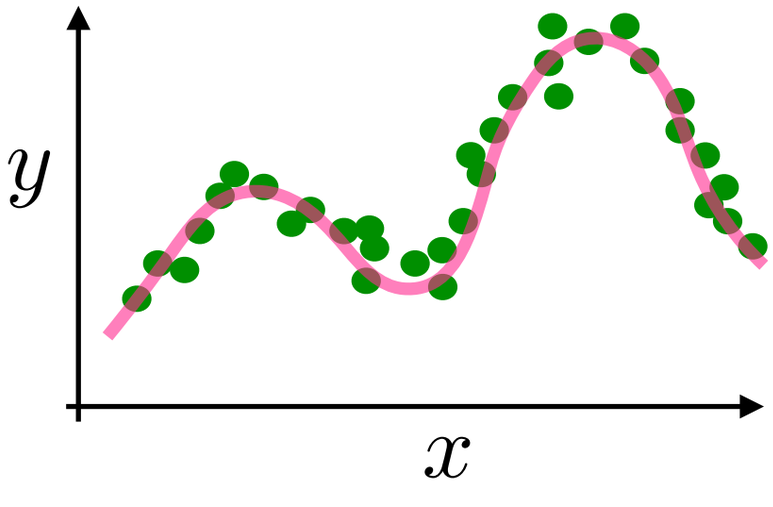
“Garbage in, garbage out”

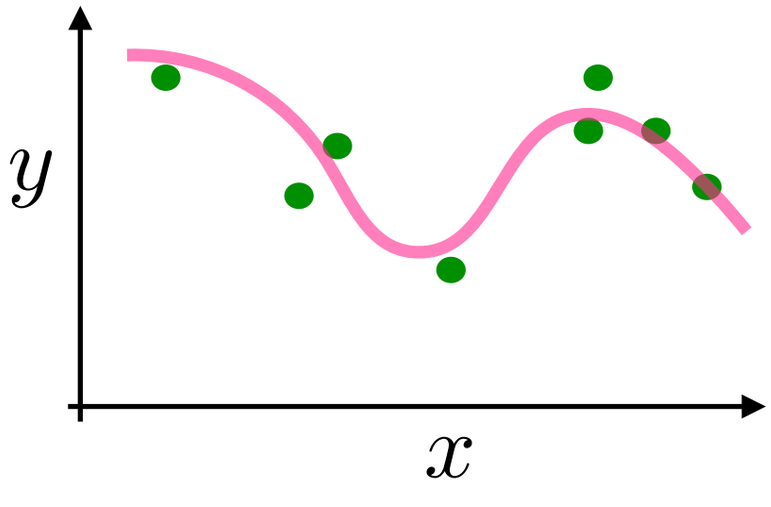


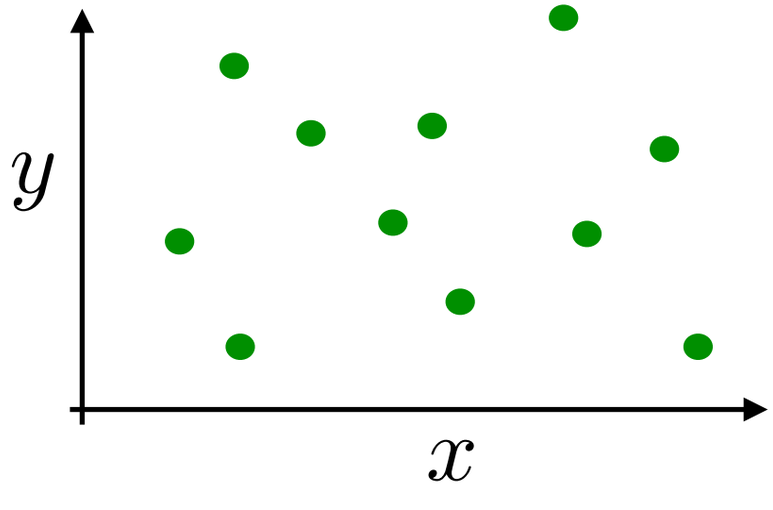
Training data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$

- **The curve seem good representative of the training data?**
- **Sufficient number of data**

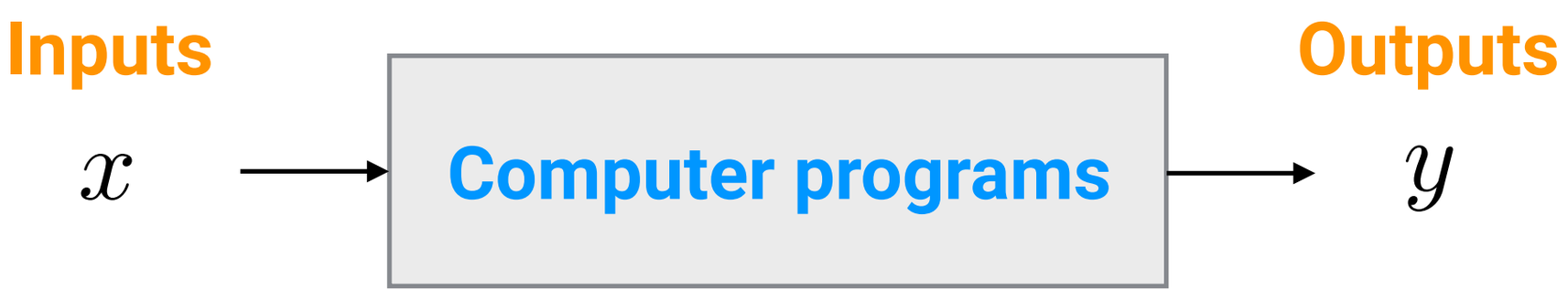
- **What you you think about this case?**
- **So so, but too early to say something due to the small number of data?**







“Garbage in, garbage out”

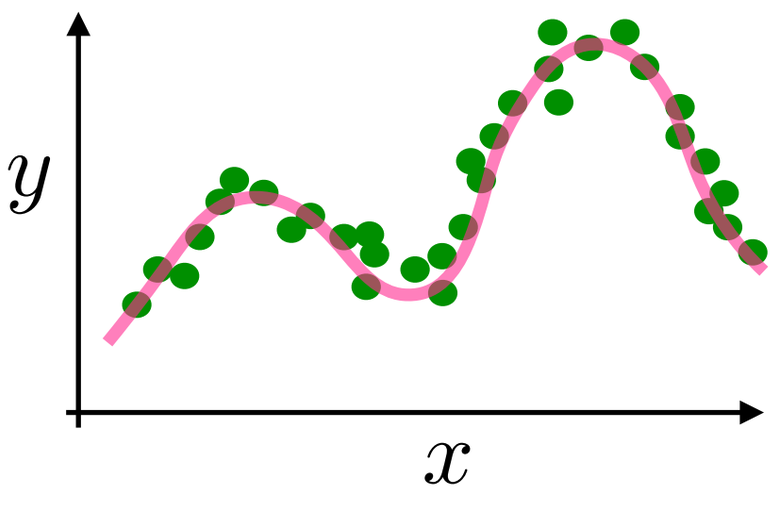


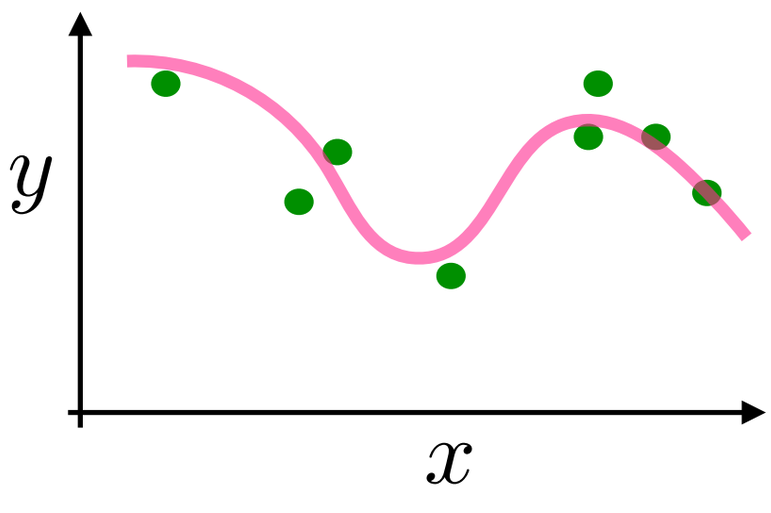
Training data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$

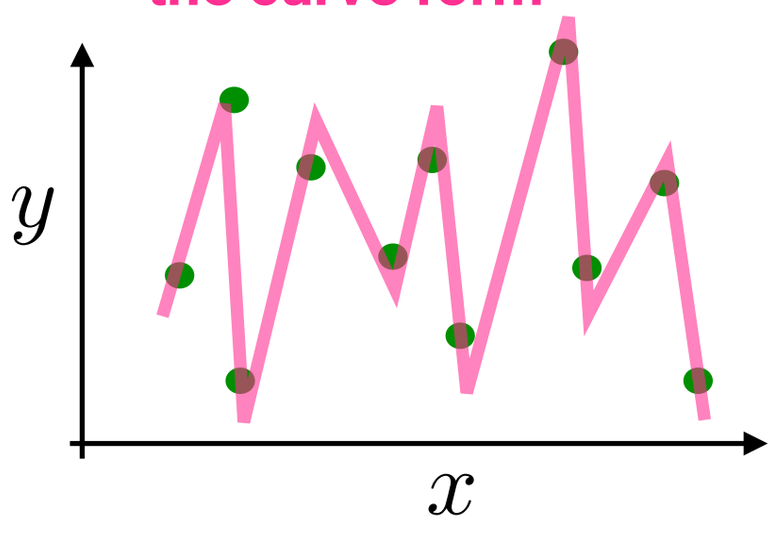
- **The curve seem good representative of the training data?**
- **Sufficient number of data**

- **What you you think about this case?**
- **So so, but too early to say something due to the small number of data?**

- **Hopelessly invalid**
- **No correlation observed between inputs and outputs**
- **Problem is the data, not the curve form**







EDA: Exploratory data analysis

- The first thing to do is always to carefully look at your data
- Check your data with basic descriptive stats and visualization.

```
df = pd.DataFrame(X, columns=[f"x{i+1}" for i in range(X.shape[1])])
df['y'] = y
```

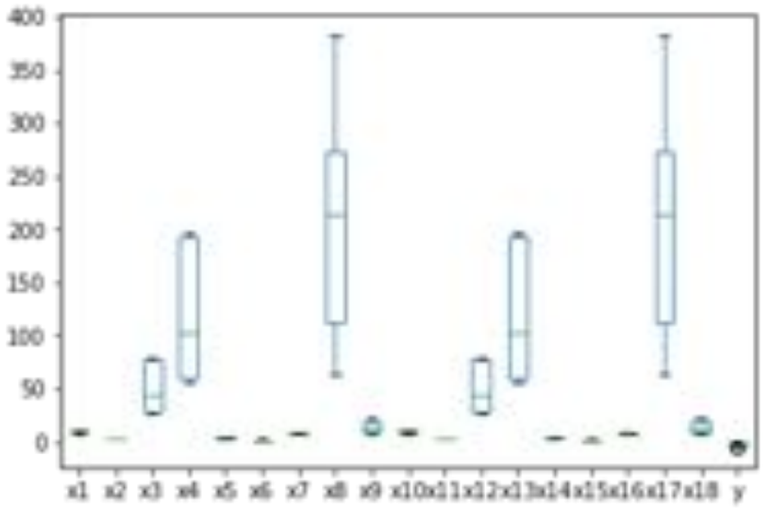
pandas
+ matplotlib

```
df.head(10)
```

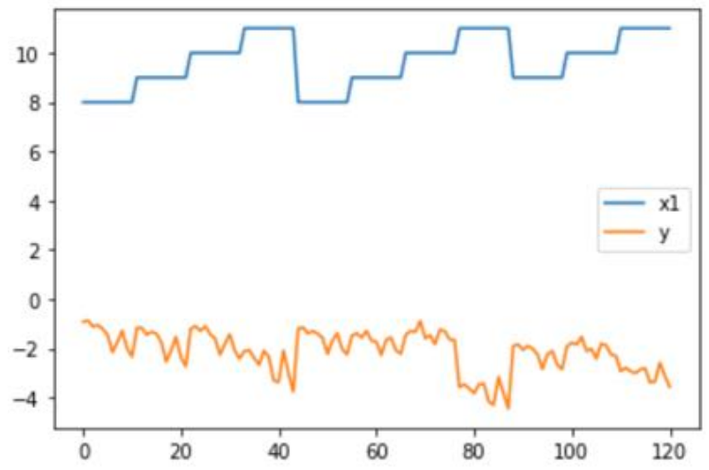
	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	y
0	8.0	2.66	26.0	55.845	4.0	1.83	7.9024	247.3	7.87	8.0	2.66	26.0	55.8450	4.0	1.83	7.9024	247.3	7.87	-0.92
1	8.0	2.66	26.0	55.845	4.0	1.83	7.9024	247.3	7.87	9.0	2.62	27.0	58.9332	4.0	1.88	7.8810	272.5	8.86	-0.87
2	8.0	2.66	26.0	55.845	4.0	1.83	7.9024	247.3	7.87	10.0	2.60	28.0	58.6934	4.0	1.91	7.6398	290.3	8.90	-1.12
3	8.0	2.66	26.0	55.845	4.0	1.83	7.9024	247.3	7.87	11.0	2.67	29.0	63.5460	4.0	1.90	7.7264	203.5	8.96	-1.05
4	8.0	2.66	26.0	55.845	4.0	1.83	7.9024	247.3	7.87	8.0	2.79	44.0	101.0700	5.0	2.20	7.3605	381.8	12.10	-1.21
5	8.0	2.66	26.0	55.845	4.0	1.83	7.9024	247.3	7.87	9.0	2.81	45.0	102.9055	5.0	2.28	7.4589	258.4	12.40	-1.46
6	8.0	2.66	26.0	55.845	4.0	1.83	7.9024	247.3	7.87	10.0	2.87	46.0	106.4200	5.0	2.20	8.3369	157.3	12.00	-2.16
7	8.0	2.66	26.0	55.845	4.0	1.83	7.9024	247.3	7.87	11.0	3.01	47.0	107.8682	5.0	1.93	7.5762	104.6	10.50	-1.75
8	8.0	2.66	26.0	55.845	4.0	1.83	7.9024	247.3	7.87	9.0	2.84	77.0	192.2170	6.0	2.20	8.9670	213.9	22.50	-1.28
9	8.0	2.66	26.0	55.845	4.0	1.83	7.9024	247.3	7.87	10.0	2.90	78.0	195.0780	6.0	2.20	8.9588	113.6	21.50	-2.01

EDA: Exploratory data analysis

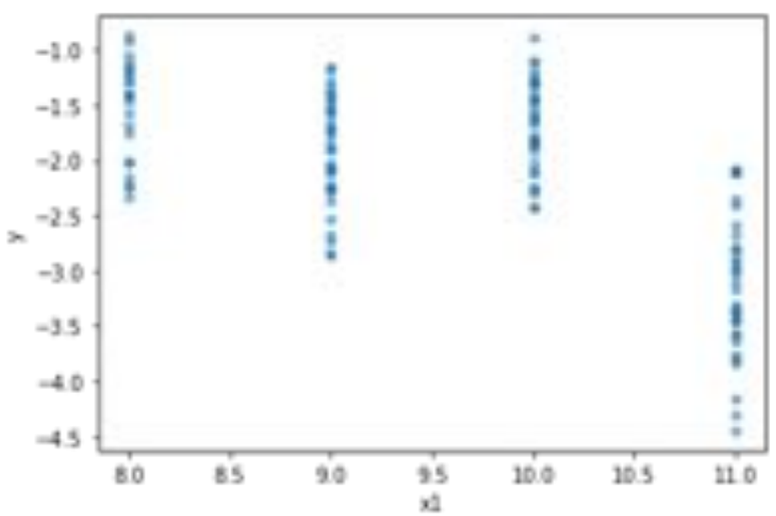
```
df.plot.box()
```




```
idx = ['x1', 'y']
df[idx].plot()
```



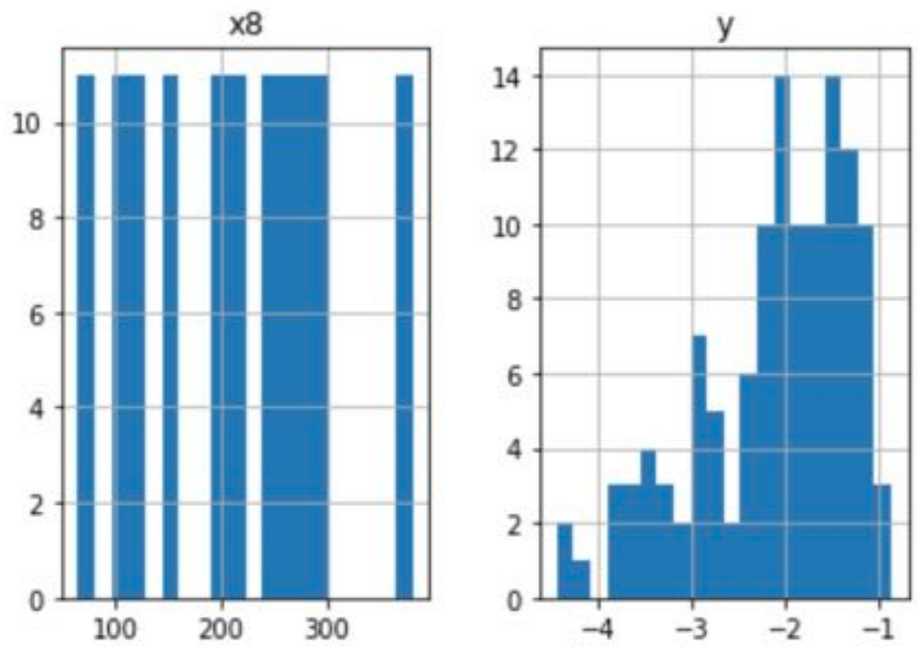

```
df.plot.scatter(x="x1", y="y", alpha=0.5)
```




```
df.describe()
```

	x1	x2	x3
count	121.000000	121.000000	121.000000
mean	9.636364	2.797273	47.818182
std	1.072381	0.138618	20.123369
min	8.000000	2.600000	26.000000
25%	9.000000	2.660000	28.000000
50%	10.000000	2.810000	45.000000
75%	11.000000	2.900000	77.000000
max	11.000000	3.010000	79.000000

```
idx = ['x8', 'y']
df[idx].hist(bins=20)
```

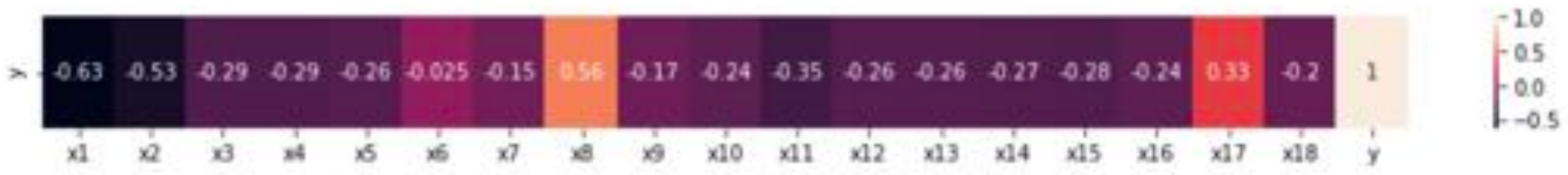



```
df.corr()['y']
```

```
x1    -0.628485
x2    -0.528149
x3    -0.291100
x4    -0.286463
x5    -0.262104
x6    -0.024741
x7    -0.152404
x8     0.564631
x9    -0.169864
x10   -0.236039
x11   -0.347411
x12   -0.264994
x13   -0.260405
x14   -0.274515
x15   -0.279717
x16   -0.240431
x17    0.329626
x18   -0.196517
```

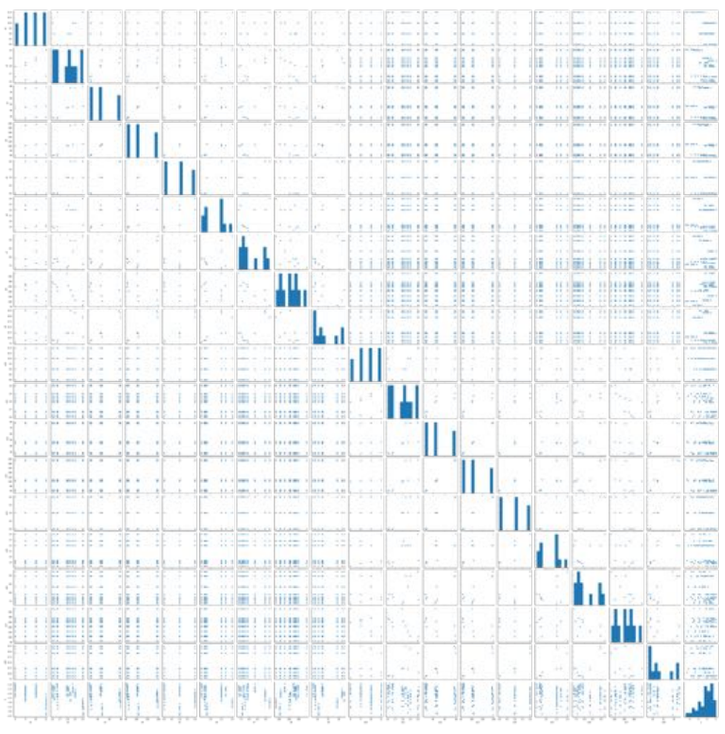
EDA: Exploratory data analysis

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(15, 1))
sns.heatmap(df.corr()['y'].to_frame().T, annot=True)
plt.show()
```

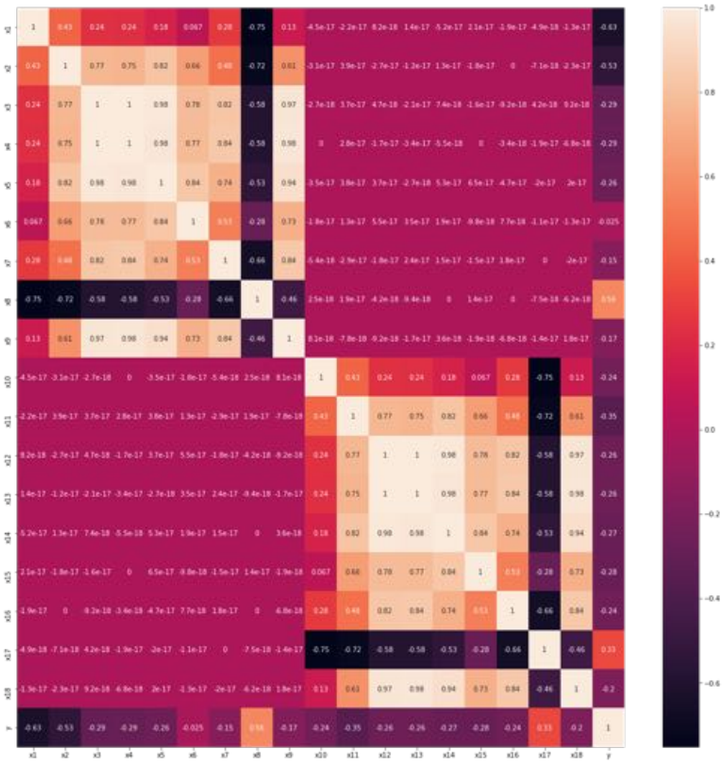


seaborn
plotly

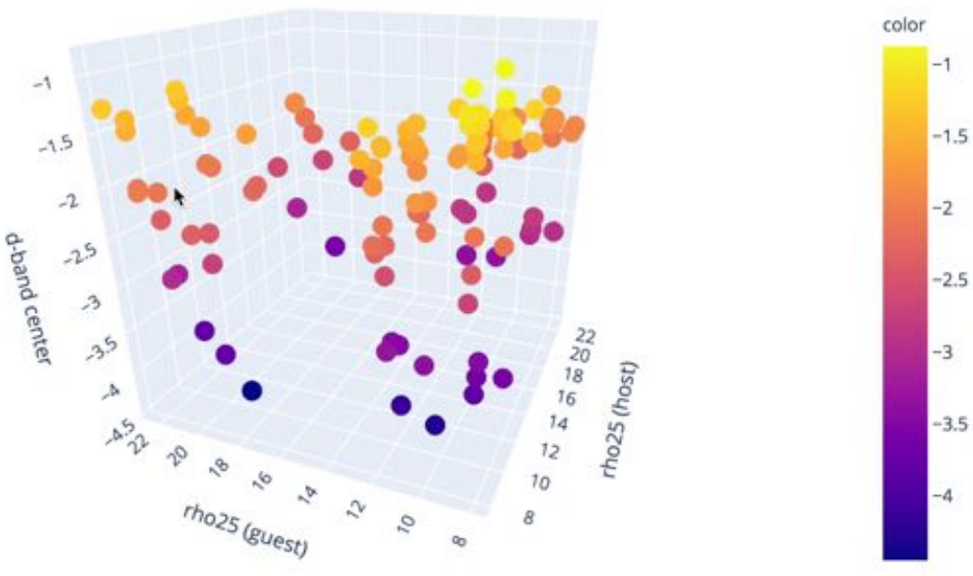
```
import seaborn as sns
sns.pairplot(df)
```



```
import matplotlib.pyplot as plt
plt.figure(figsize=(20,20))
sns.heatmap(df.corr(), annot=True)
```

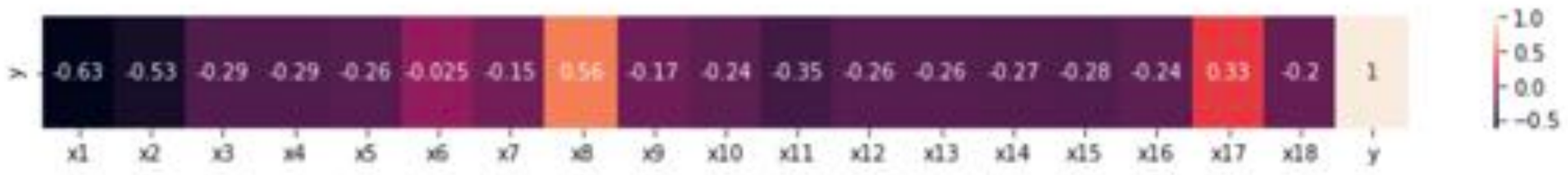


```
import plotly.express as px
fig = px.scatter_3d(df, \
                    x='x8', y='x17', z='y', \
                    color=y)
fig.show()
```



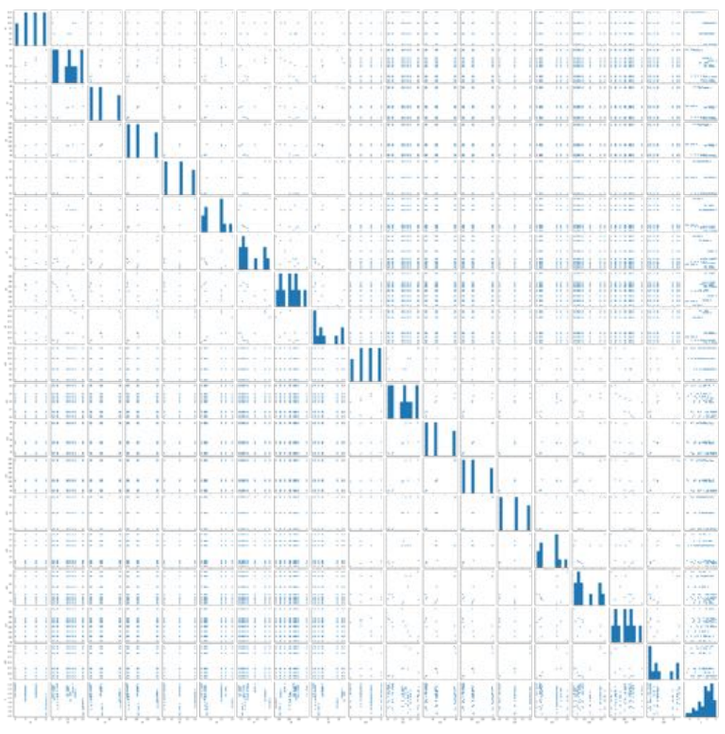
EDA: Exploratory data analysis

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(15, 1))
sns.heatmap(df.corr()['y'].to_frame().T, annot=True)
plt.show()
```

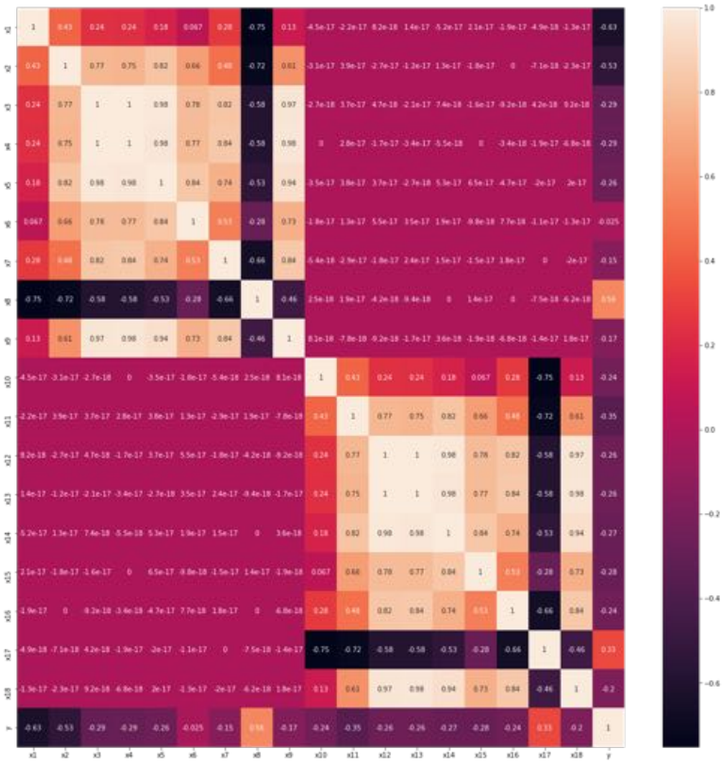


seaborn
plotly

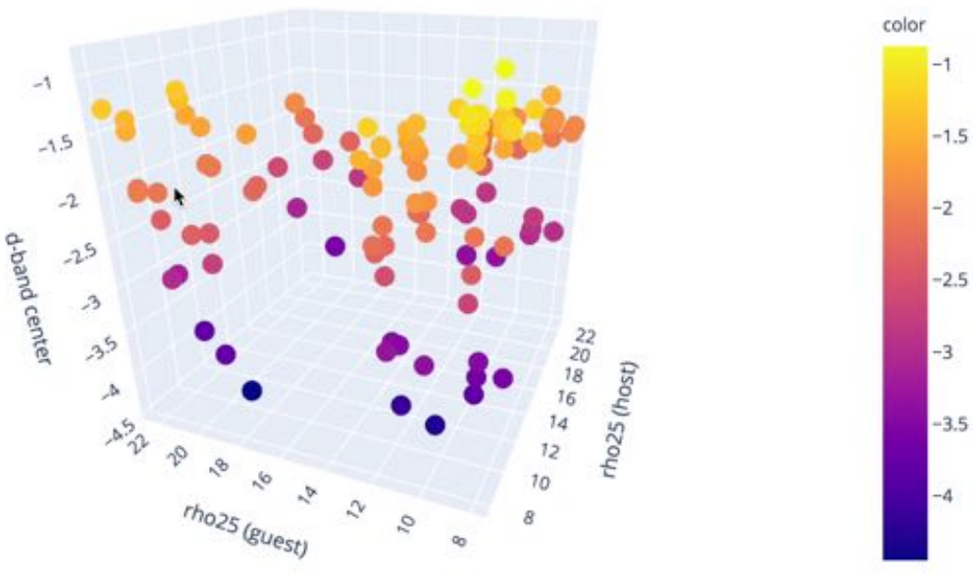
```
import seaborn as sns
sns.pairplot(df)
```



```
import matplotlib.pyplot as plt
plt.figure(figsize=(20,20))
sns.heatmap(df.corr(), annot=True)
```



```
import plotly.express as px
fig = px.scatter_3d(df, \
                    x='x8', y='x17', z='y', \
                    color=y)
fig.show()
```



Aug 26: 10:30~12:00 (90min)

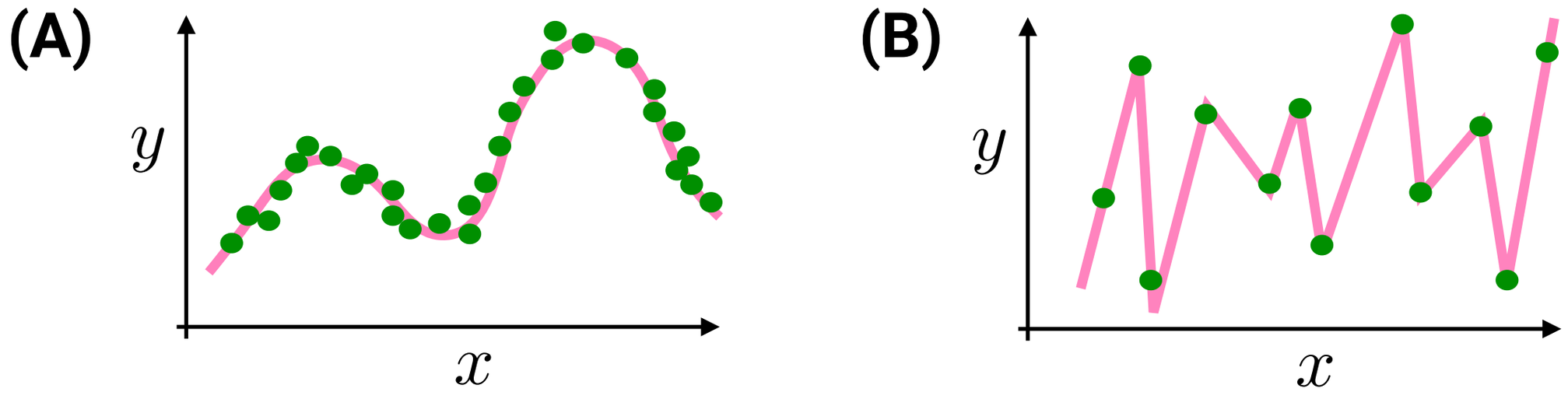
1. What is "machine learning"?
2. Why does it matter to chemists?
3. Let's try it in your browser (with no setup!)

Aug 26: 13:00~14:30 (90min)

4. Five things all beginners should know
   - "The quality of your inputs decide the quality of your output"
   - Training / validation / test data
   - Tuning hyperparameters
   - Identification and design of input variables (or "descriptors")
   - "Correlation does not imply causation"
5. Standard pipeline and deep learning
6. Current efforts and future directions

But how can we evaluate the ML prediction?

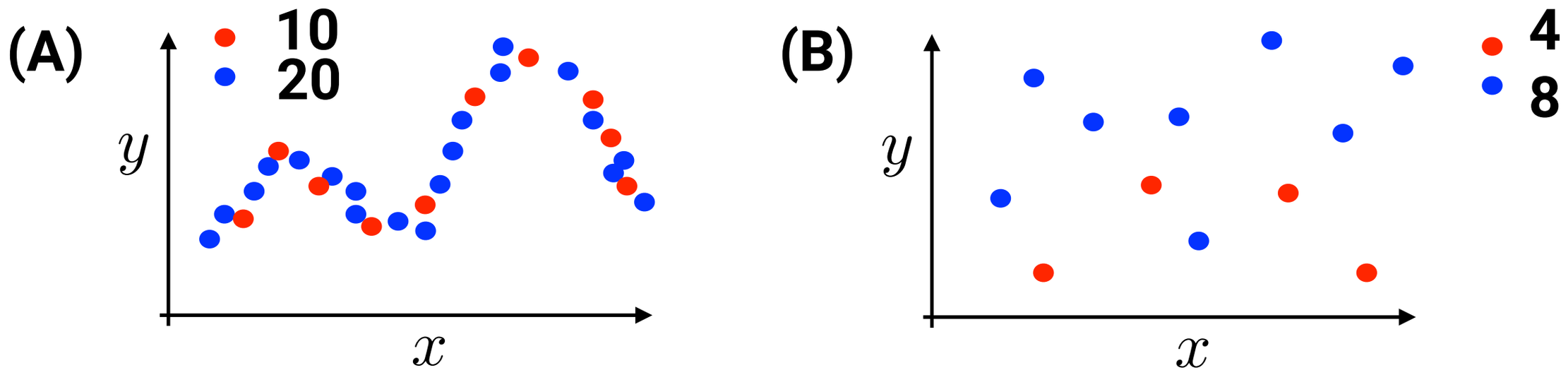
How can we quantitatively know that curve fitting (A) is good but (B) is bad?



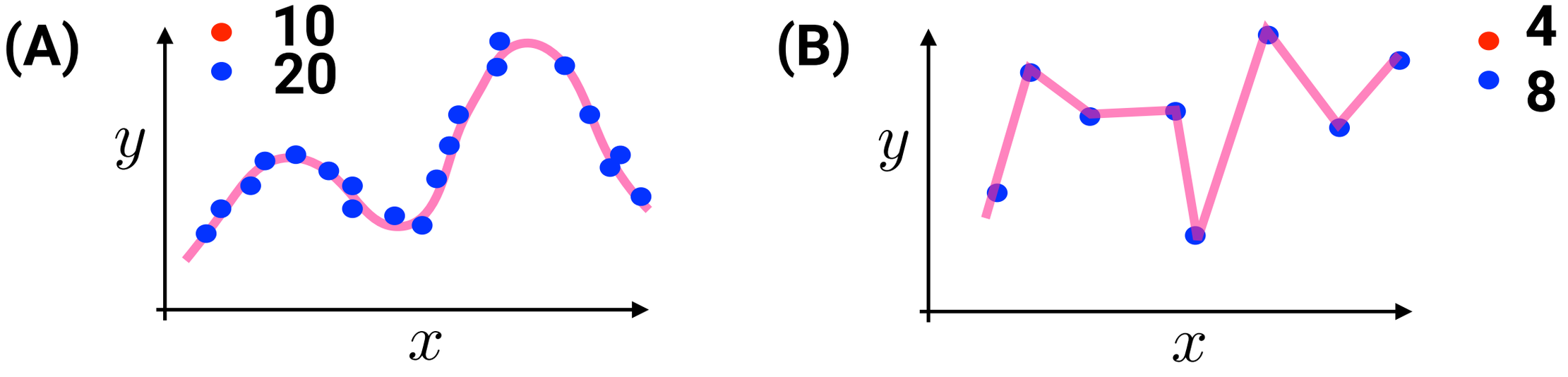
It seems that the prediction errors of (B) are even much smaller than (A)…

Training data and test data

We randomly hold out some (say 1/3) of the data as the test data.

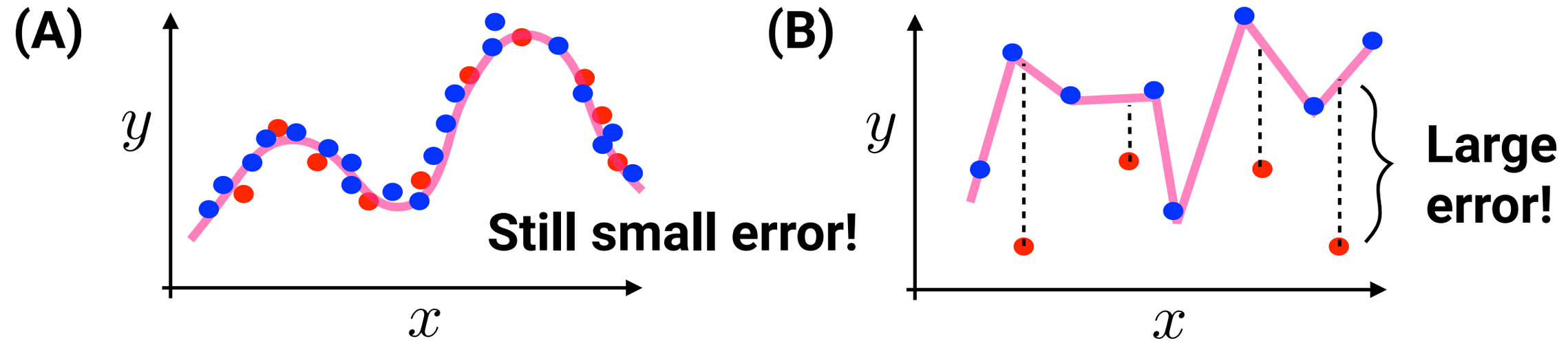


Apply ML to the remaining training data, and obtain the prediction curve.



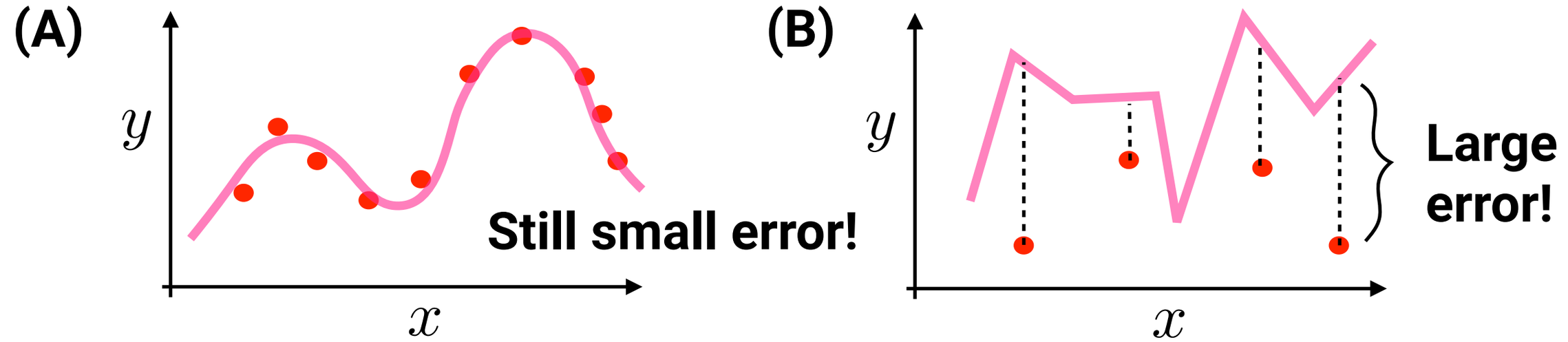
The performance evaluated by training-test split

Test how the predictions by curve are accurate using the test data held out.



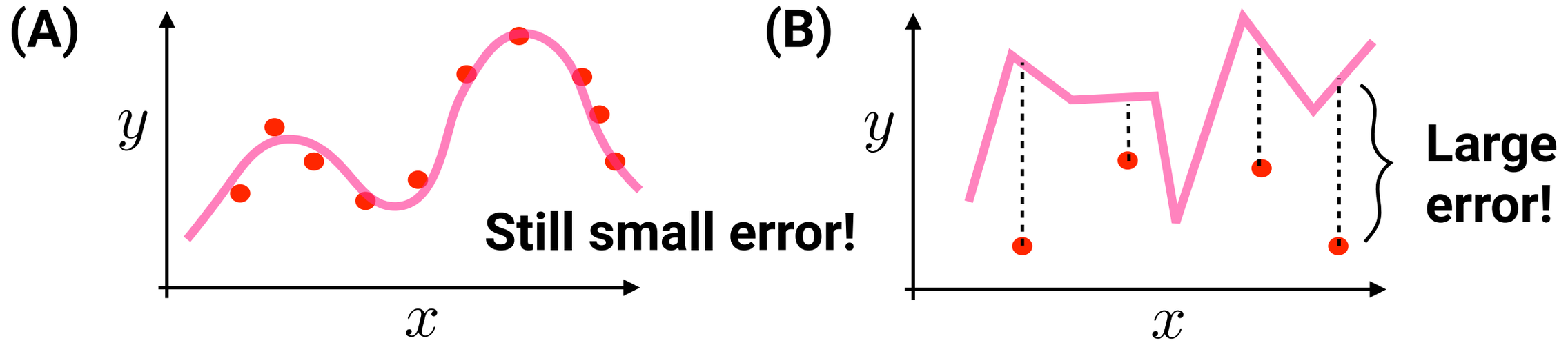
The performance evaluated by training-test split

Test how the predictions by curve are accurate using the test data held out.



The performance evaluated by training-test split

Test how the predictions by curve are accurate using the test data held out.



- **The performance of ML should be measured by the "test error".**
- **Note that making the training errors zero is easy.**
  Just memorize the training data.
- **In well-posed situations, the test errors are always larger than the training error.**
- **Ideally, both training errors and test errors are small.**

Example:

Let's teach "addition" by machine learning.

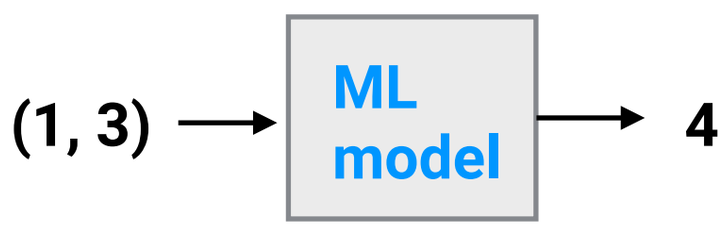
```
[1]  from sklearn.ensemble import ExtraTreesRegressor
     from sklearn.linear_model import LinearRegression
```

```
X = [[1, 3],
     [2, 5],
     [5, 9],
     [3, 10],
     [5, 6]]
y = [4,
     7,
     14,
     13,
     11]
```

1 + 3 = 4
2 + 5 = 7
5 + 9 = 14
3 + 10 = 13
5 + 6 = 11

```
[7]  model = LinearRegression()
     model.fit(X, y)
     model.predict(X)
```

```
array([ 4.,  7., 14., 13., 11.])
```



The model gives perfect answers that we taught.

Example:

Hey my A.I. learned "addition" only by 5 examples! Cool!

```
[8] model.predict([[3, 1], [5, 2], [9, 5], [10, 3], [6, 5]])
```

```
array([ 4.,  7., 14., 13., 11.])
```

```
model.predict([[1, 1],
               [2, 9],
               [45, 1]])
```

```
array([ 2., 11., 46.])
```

```
[9] model.predict([[123489, 82394732]])
```

```
array([82518221.])
```

```
[10] 123489 + 82394732
```

```
82518221
```

ML understood
1 + 3 = 3 + 1 = 4
2 + 5 = 5 + 2 = 7
5 + 9 = 9 + 5 = 14
3 + 10 = 10 + 3 = 13
5 + 6 = 6 + 5 = 11

ML gave perfect answers for unseen problems!

What if we change the model?

```
model = ExtraTreesRegressor()
model.fit(X, y)
model.predict(X)
```

```
array([ 4.,  7., 14., 13., 11.])
```

The model gives perfect answers that we taught.

```
[12] model.predict([[3, 1], [5, 2], [9, 5], [10, 3], [6, 5]])
```

```
array([7.33, 7.97, 9.52, 7.97, 9.52])
```

But ...

```
[13] model.predict([[1, 1],
                    [2, 9],
                    [45, 1]])
```

```
array([ 4.  , 11.66,  7.97])
```

Seems totally wrong for unseen ones...

```
[14] model.predict([[123489, 82394732]])
```

```
array([13.49])
```

What if you try some "deep learning" here?

```
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    layers.Dense(20, activation="relu"),
    layers.Dense(40, activation="relu"),
    layers.Dense(20, activation="relu"),
    layers.Dense(50, activation="relu"),
    layers.Dense(1)
])
model.compile(loss='mean_squared_error', \
              optimizer='adam')
```

```
model.fit(X, y, epochs=200)
```

```
model.predict(X)
```

```
array([[ 4.31061 ],
       [ 7.160423],
       [13.888492],
       [12.882807],
       [11.047327]], dtype=float32)
```

```
[35] y
```

```
[4, 7, 14, 13, 11]
```

```
[29] model.predict([[3, 1], [5, 2], [9, 5], [10, 3], [6, 5]])
```

```
array([[ 3.0535243],
       [ 5.1600466],
       [10.674399 ],
       [ 8.710056 ],
       [ 9.379226 ]], dtype=float32)
```

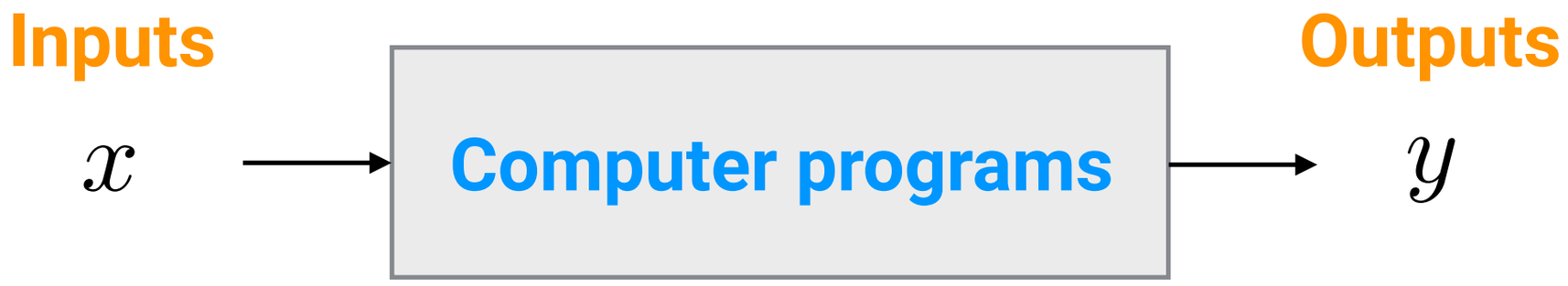
```
[36] model.predict([[1, 1],
                    [2, 9],
                    [45, 1]])
```

```
array([[ 2.405819],
       [11.07457 ],
       [31.65177 ]], dtype=float32)
```

```
[37] model.predict([[123489, 82394732]])
```

```
array([[81370680.]], dtype=float32)
```

Training / validation / test split



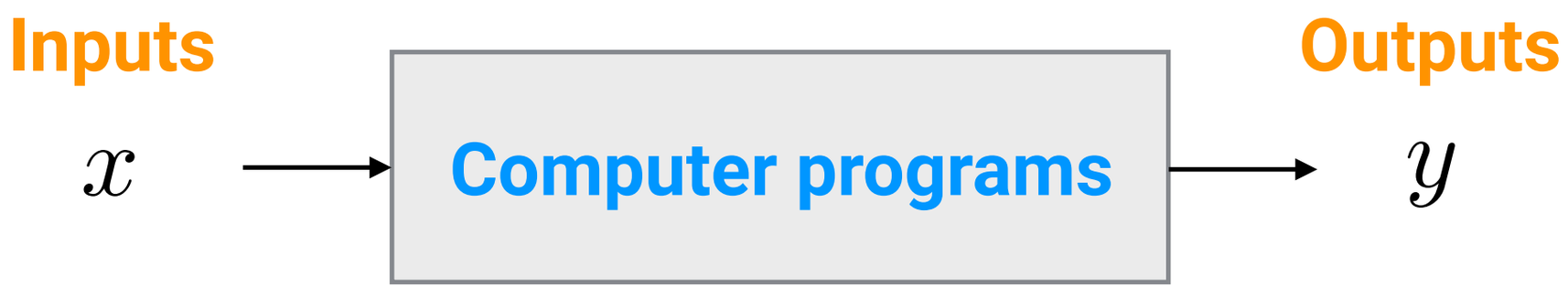
Available data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$

Training data — Used for "model training (fitting)"

Validation data — Used for "hyperparameter tuning" (or "model selection")

Test data — Used for evaluating the prediction performance

Cross validation



Available data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$

→ **subset-1, subset-2, subset-3, ..., subset-k**

Try:

Training data		**Test data**
All except subset-1	→	**subset-1**
All except subset-2	→	**subset-2**
⋮		⋮
All except subset-k	→	**subset-k**

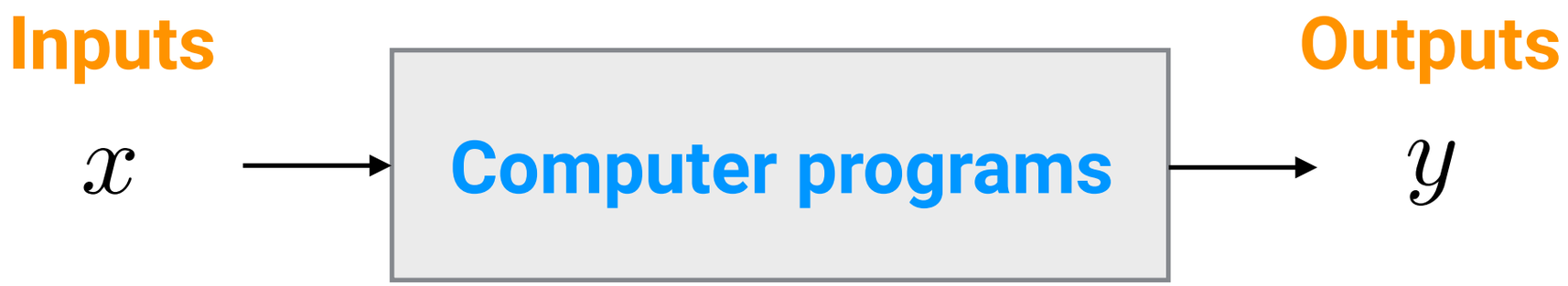
Cross validation

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import cross_validate

model = GradientBoostingRegressor()
scores = cross_validate(model, X, y, cv=5, \
                        scoring='neg_root_mean_squared_error', \
                        return_train_score=True)
```

```
print("train RMSE %.3f" % -scores['train_score'].mean())
print("test RMSE %.3f" % -scores['test_score'].mean())
```

```
train RMSE 0.042
test RMSE 0.139
```

Cross validation (double/nested cross validation)



Available data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$

→ subset-1, subset-2, subset-3, ..., subset-k

Try:

Training + validation data

All except subset-1

→ subset-a, subset-b, ...

training data	validation data
all except subset-a	subset-a

⋮

→ **Test data**

subset-1

⋮

Cross validation (double/nested cross validation)

```
[43] model
```

```
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.1, loss='ls', max_depth=3,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=100,
                          n_iter_no_change=None, presort='deprecated',
                          random_state=None, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
```

```
from sklearn.model_selection import GridSearchCV
model = GradientBoostingRegressor()
hparam_grid = {'learning_rate': [0.1, 0.5, 1.0], \
               'max_leaf_nodes':[2, 8, 16, 32, None]}
cvmodel = GridSearchCV(model, hparam_grid, cv=3)
scores = cross_validate(cvmodel, X, y, cv=5, \
                        scoring='neg_root_mean_squared_error', \
                        return_train_score=True)
print("train RMSE %.3f" % -scores['train_score'].mean())
print("test RMSE %.3f" % -scores['test_score'].mean())
```

```
train RMSE 0.025
test RMSE 0.130
```

`sklearn.model_selection`: Model Selection

User guide: See the Cross-validation: evaluating estimator performance, Tuning the hyper-parameters of an estimator and Learning curve sections for further details.

Splitter Classes

`model_selection.GroupKFold`([n_splits])	K-fold iterator variant with non-overlapping groups.
`model_selection.GroupShuffleSplit`([...])	Shuffle-Group(s)-Out cross-validation iterator
`model_selection.KFold`([n_splits, shuffle, ...])	K-Folds cross-validator
`model_selection.LeaveOneGroupOut`	Leave One Group Out cross-validator
`model_selection.LeavePGroupsOut`(n_groups)	Leave P Group(s) Out cross-validator
`model_selection.LeaveOneOut`	Leave-One-Out cross-validator
`model_selection.LeavePOut`(p)	Leave-P-Out cross-validator
`model_selection.PredefinedSplit`(test_fold)	Predefined split cross-validator
`model_selection.RepeatedKFold`(*[, n_splits, ...])	Repeated K-Fold cross validator.
`model_selection.RepeatedStratifiedKFold`(*[, ...])	Repeated Stratified K-Fold cross validator.
`model_selection.ShuffleSplit`([n_splits, ...])	Random permutation cross-validator
`model_selection.StratifiedKFold`([n_splits, ...])	Stratified K-Folds cross-validator
`model_selection.StratifiedShuffleSplit`([...])	Stratified ShuffleSplit cross-validator
`model_selection.TimeSeriesSplit`([n_splits, ...])	Time Series cross-validator

Splitter Functions

`model_selection.check_cv`([cv, y, classifier])	Input checker utility for building a cross-validator
`model_selection.train_test_split`(*arrays, ...)	Split arrays or matrices into random train and test subsets

Hyper-parameter optimizers

`model_selection.GridSearchCV`(estimator, ...)	Exhaustive search over specified parameter values for an estimator.
`model_selection.ParameterGrid`(param_grid)	Grid of parameters with a discrete number of values for each.
`model_selection.ParameterSampler`(...[, ...])	Generator on parameters sampled from given distributions.
`model_selection.RandomizedSearchCV`(...[, ...])	Randomized search on hyper parameters.

Model validation

`model_selection.cross_validate`(estimator, X)	Evaluate metric(s) by cross-validation and also record fit/score times.
`model_selection.cross_val_predict`(estimator, X)	Generate cross-validated estimates for each input data point
`model_selection.cross_val_score`(estimator, X)	Evaluate a score by cross-validation
`model_selection.learning_curve`(estimator, X, ...)	Learning curve.
`model_selection.permutation_test_score`(...)	Evaluate the significance of a cross-validated score with permutations
`model_selection.validation_curve`(estimator, ...)	Validation curve.

Aug 26: 10:30~12:00 (90min)

1. What is "machine learning"?
2. Why does it matter to chemists?
3. Let's try it in your browser (with no setup!)

Aug 26: 13:00~14:30 (90min)

4. Five things all beginners should know
    - "The quality of your inputs decide the quality of your output"
    - Training / validation / test data
    - Tuning hyperparameters
    - Identification and design of input variables (or "descriptors")
    - "Correlation does not imply causation"
5. Standard pipeline and deep learning
6. Current efforts and future directions

Many models have hyperparameters to be tuned

```
from sklearn.gaussian_process import GaussianProcessRegressor
model = GaussianProcessRegressor()
print(model)
```

```
GaussianProcessRegressor(alpha=1e-10, copy_X_train=True, kernel=None,
                         n_restarts_optimizer=0, normalize_y=False,
                         optimizer='fmin_l_bfgs_b', random_state=None)
```

```
from sklearn.svm import SVR
model = SVR()
print(model)
```

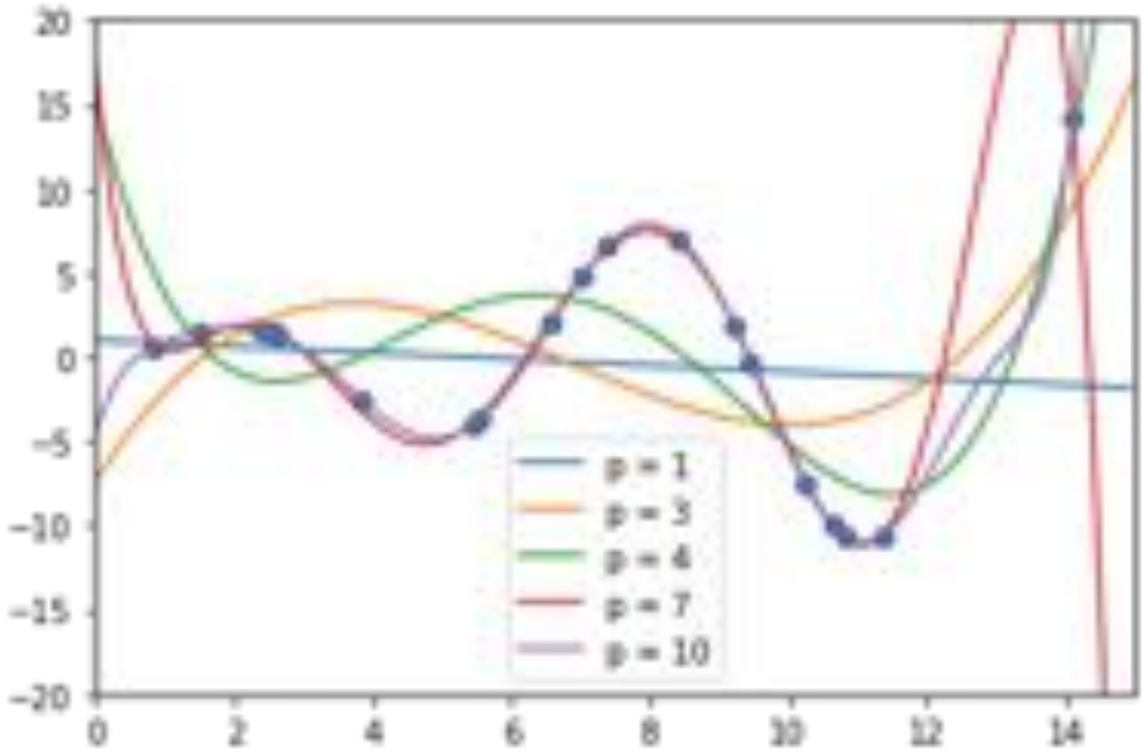
```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

```
from sklearn.ensemble import GradientBoostingRegressor
model = GradientBoostingRegressor()
print(model)
```

```
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.1, loss='ls', max_depth=3,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=100,
                          n_iter_no_change=None, presort='deprecated',
                          random_state=None, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
```

The *default values* often not appropriate for your problem

Parameters vs Hyperparameters



Hyperparameters are not tuned in the training process. (we need to specify it beforehand properly)

When we try to fit polynomial functions:

$$\hat{y} = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_p x_p^p$$

Parameters: $\beta_0, \beta_1, \ldots, \beta_p$ (coefficients)

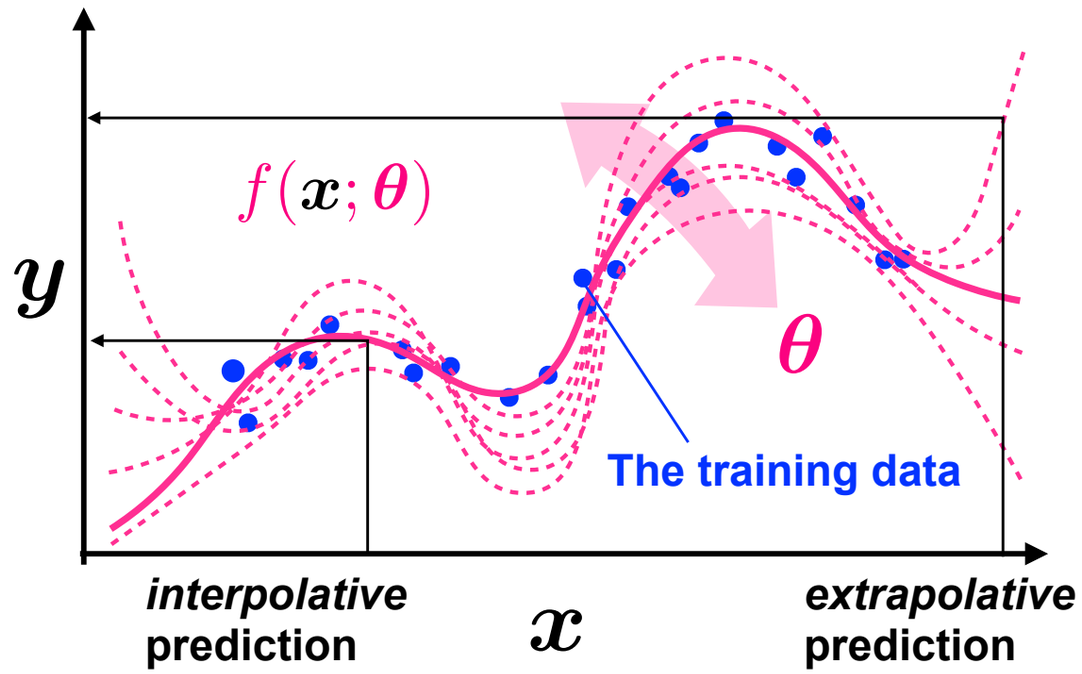
Hyperarameters: p (order of polynomials)

(Supervised) machine learning

A function $f(x;\theta)$ best fitted to
a given set of example input-output pairs
(the training data).

$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$

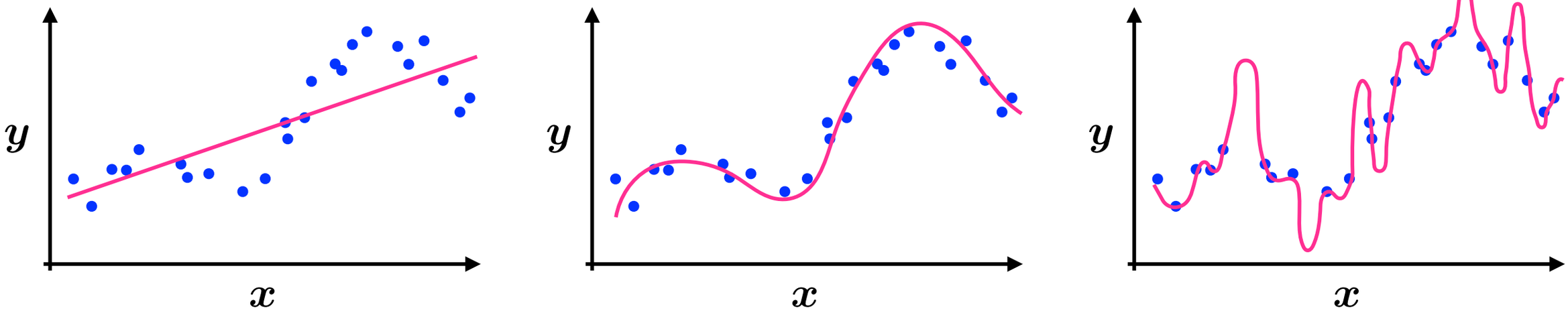
(High-dimensional)



"The bias-variance tradeoff"

Underfitting
(High bias, Low variance)

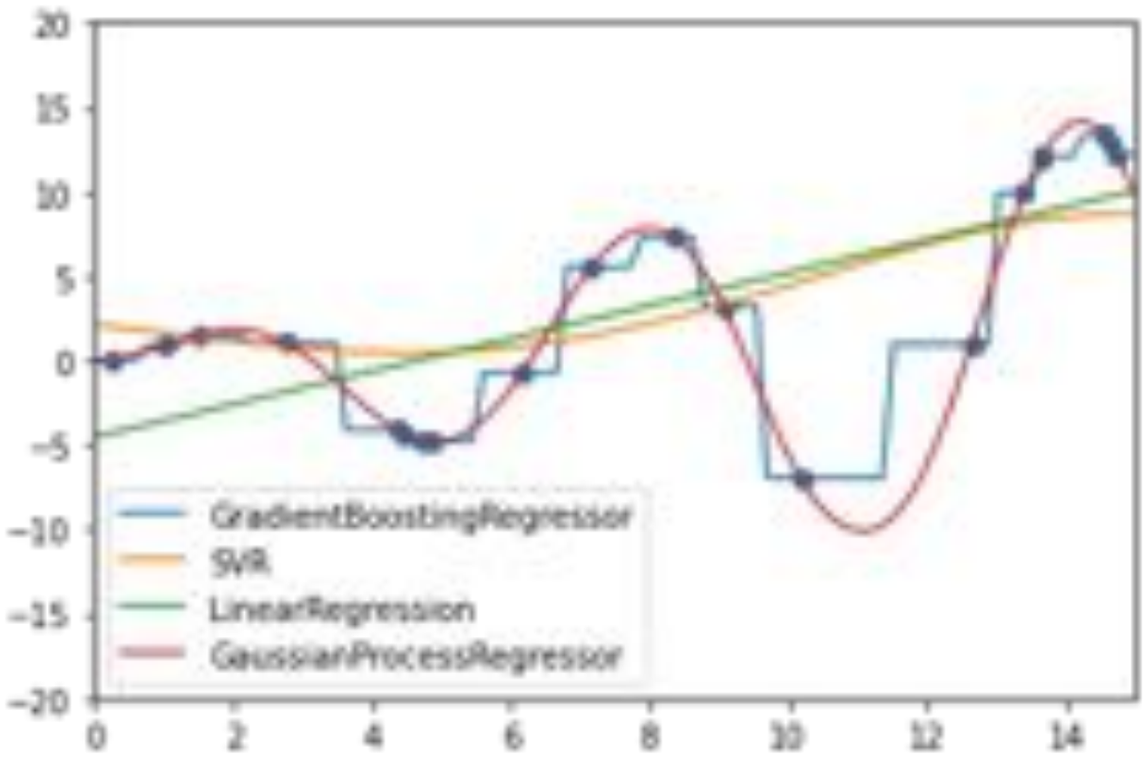
Overfitting
(Low bias, High variance)



Low **Model Complexity** **High**

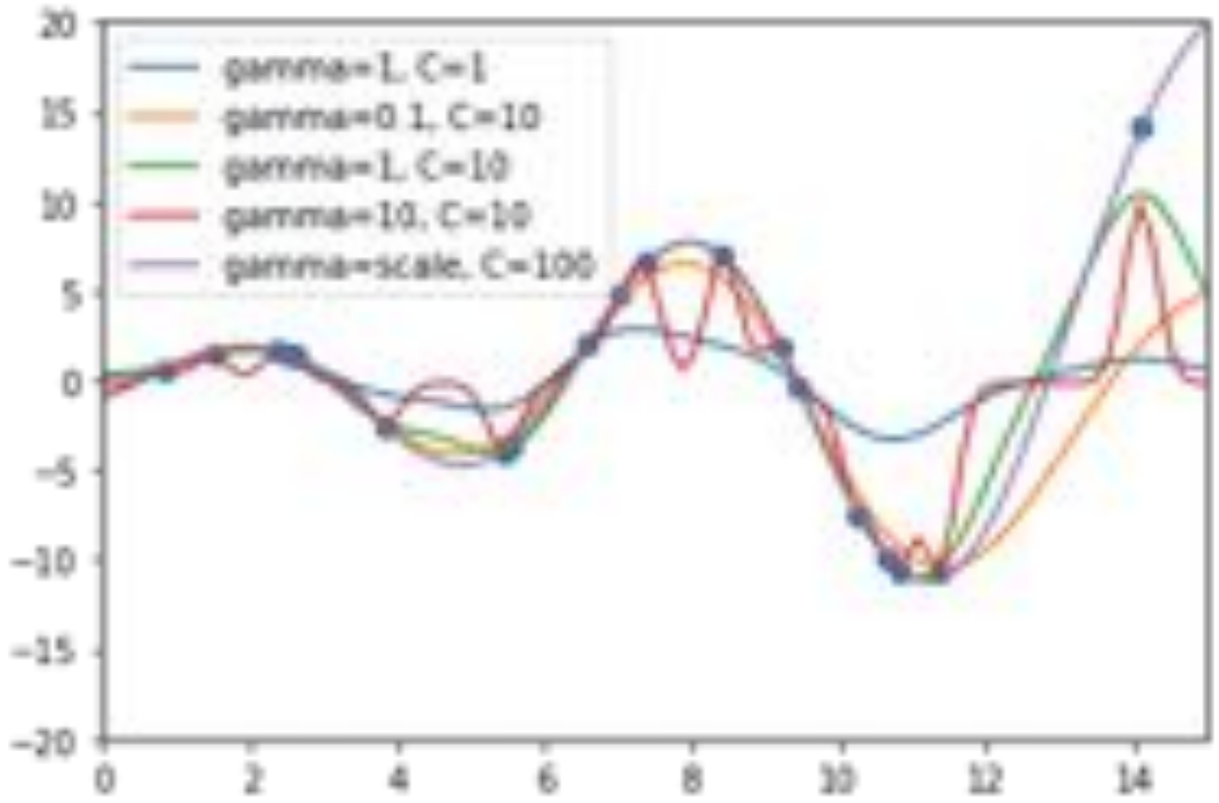
Different models

- Different models uses different forms of fitting functions
- Remember that **high-dimension can be counter-intuitive**




```
x = np.random.uniform(low=0, high=15, size=20)
y = x * np.sin(x)

fig, ax = plt.subplots()
ax.scatter(x, y)

models = [GradientBoostingRegressor(),
          SVR(), LinearRegression(),
          GaussianProcessRegressor()]

for model in models:
  model.fit(x.reshape(-1,1), y)
  x_test = np.linspace(0, 15, 100).reshape(-1,1)
  y_pred = model.predict(x_test)
  ax.plot(x_test, y_pred, \
          label=model.__class__.__name__)

ax.set_xlim(0, 15)
ax.set_ylim(-20, 20)
ax.legend()
plt.show()
```

Control the model complexity properly



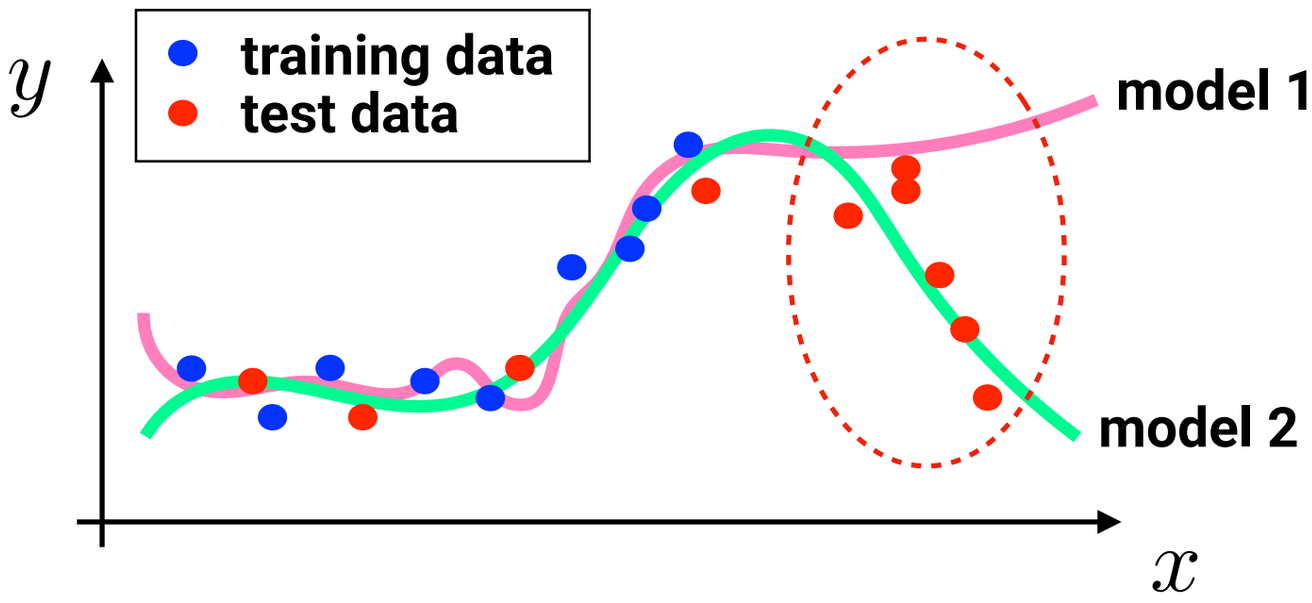

```
from sklearn.svm import SVR
model = SVR(gamma=10.0, C=10.0)
```

```
from sklearn.svm import SVR
model = SVR()
model
```

```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

Needs for "validation" data

Seemingly the test data is enough also to determine which hyperparameters are good, but ...



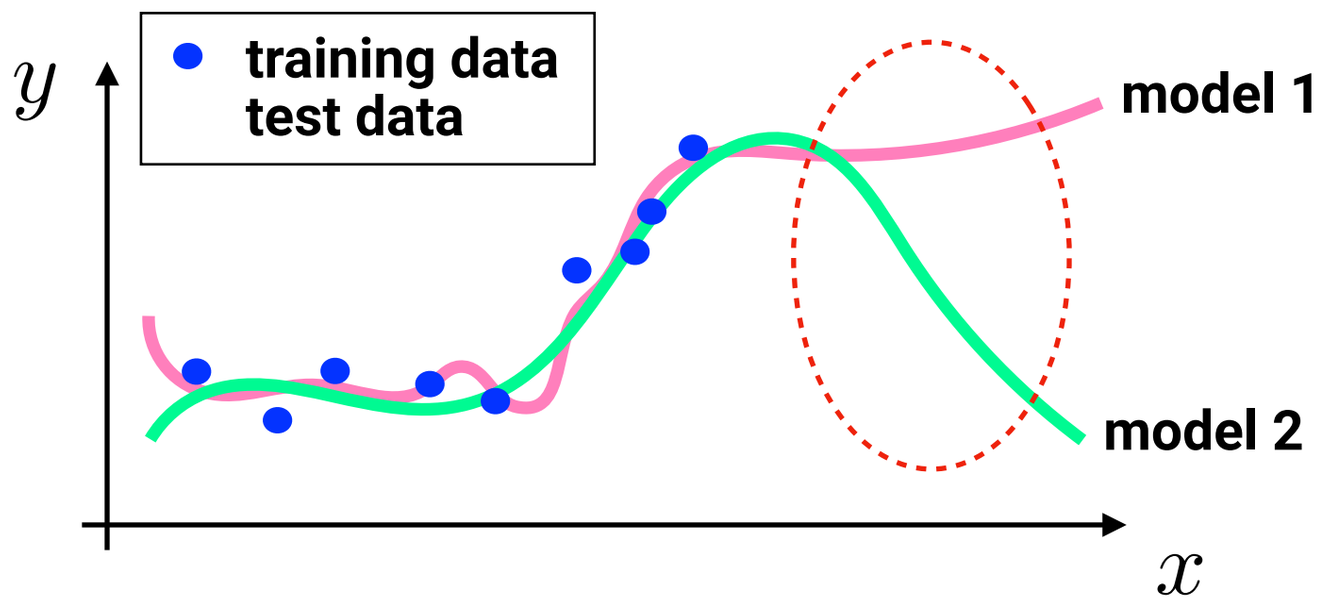
The model 2 performs better over "test data", but this would be sort of cheating (called "data leakage")

If we have other "test data", we can detect this fallacy, but if we conclude here, it doesn't make any sense.

https://en.wikipedia.org/wiki/Data_dredging

Needs for "validation" data

Seemingly the test data is enough also to determine which hyperparameters are good, but ...



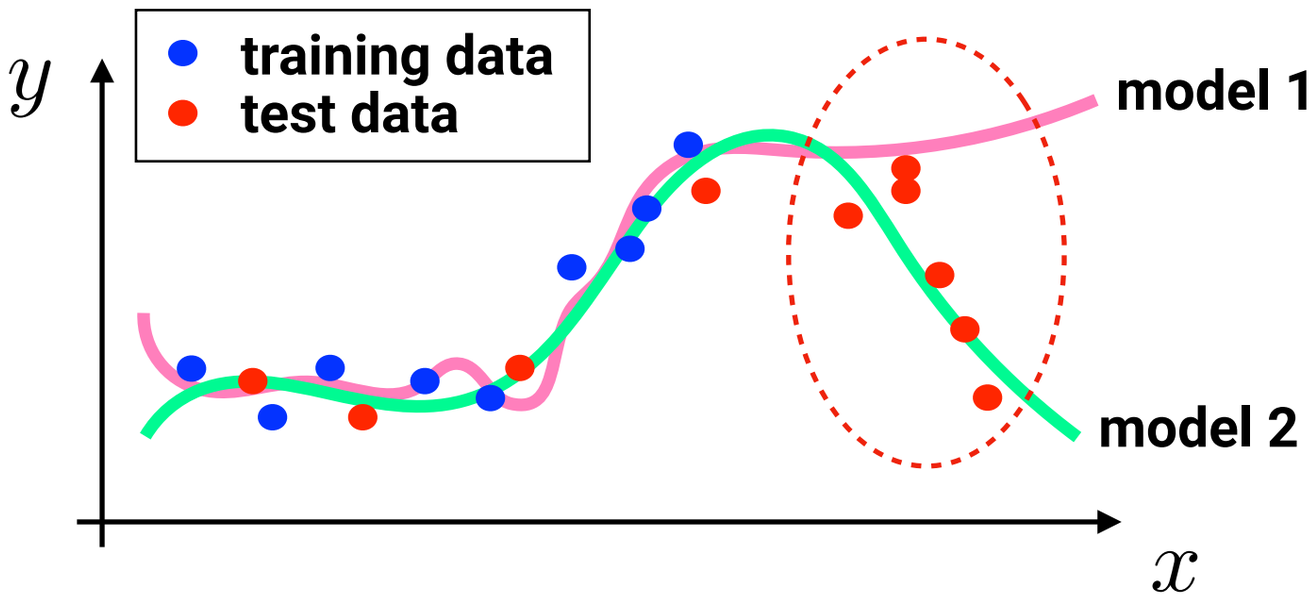
The model 2 performs better over "test data", but this would be sort of cheating (called "data leakage")

If we have other "test data", we can detect this fallacy, but if we conclude here, it doesn't make any sense.

https://en.wikipedia.org/wiki/Data_dredging

Needs for "validation" data

Seemingly the test data is enough also to determine which hyperparameters are good, but ...

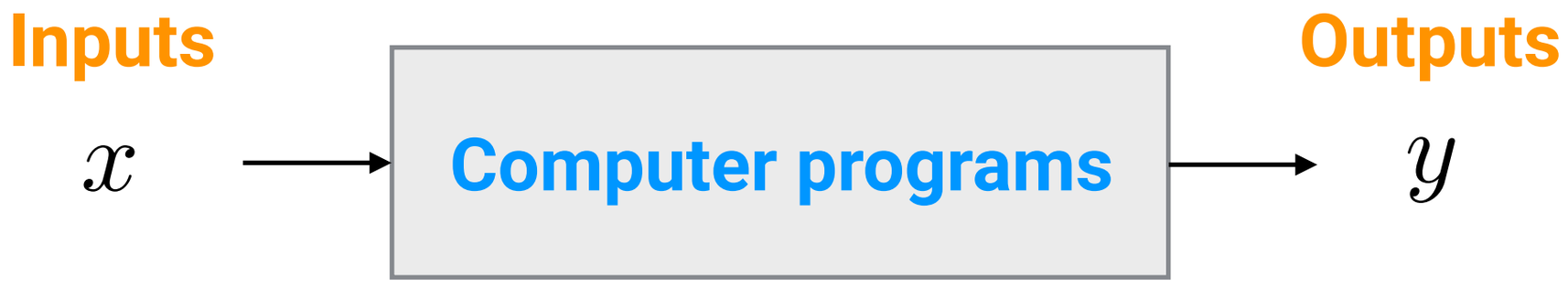


The model 2 performs better over "test data", but this would be sort of cheating (called "data leakage")

If we have other "test data", we can detect this fallacy, but if we conclude here, it doesn't make any sense.

https://en.wikipedia.org/wiki/Data_dredging

Training / validation / test split



Available data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$

Training data — Used for "model training (fitting)"

Validation data — Used for "hyperparameter tuning" (or "model selection")

Test data — Used for evaluating the prediction performance

Aug 26: 10:30~12:00 (90min)

1. What is "machine learning"?
2. Why does it matter to chemists?
3. Let's try it in your browser (with no setup!)

Aug 26: 13:00~14:30 (90min)

4. Five things all beginners should know
    - "The quality of your inputs decide the quality of your output"
    - Training / validation / test data
    - Tuning hyperparameters
    - Identification and design of input variables (or "descriptors")
    - "Correlation does not imply causation"
5. Standard pipeline and deep learning
6. Current efforts and future directions

When can (or should) we use ML?

In the 'addition' example,
the ML version has no practical advantages...

accurate, fast

```
def my_add(x1, x2):
  return x1 + x2
```

```
my_add(131, 28)
```

```
159
```

inaccurate, not that fast

```
model = LinearRegression()
model.fit(X, y)

def ml_add(x1, x2):
  return model.predict([[x1, x2]])[0]
```

```
ml_add(131, 28)
```

```
159.0000000000001
```

```
X = [[1, 3],
     [2, 5],
     [5, 9],
     [3, 10],
     [5, 6]]
y = [4,
     7,
     14,
     13,
     11]
```

Inputs → **my_add / ml_add** → **Outputs**

(x_1, x_2) → $x_1 + x_2$

- **Too unclear to be explicitly modeled**
- **Too complicated to be explicitly modeled**
- **Very time-consuming**

Variable identification is very important!

Inputs

$(x_1, x_2, \ldots, x_p)$

→ **ML_model** f →

Outputs

$(y_1, y_2, \ldots, y_q)$

- **Too unclear to be explicitly modeled**
- **Too complicated to be explicitly modeled**
- **Very time-consuming**

$$(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_q) = f(x_1, x_2, \ldots, x_p)$$

The prediction is made directly by the outputs of the function **defined on chosen variables**.

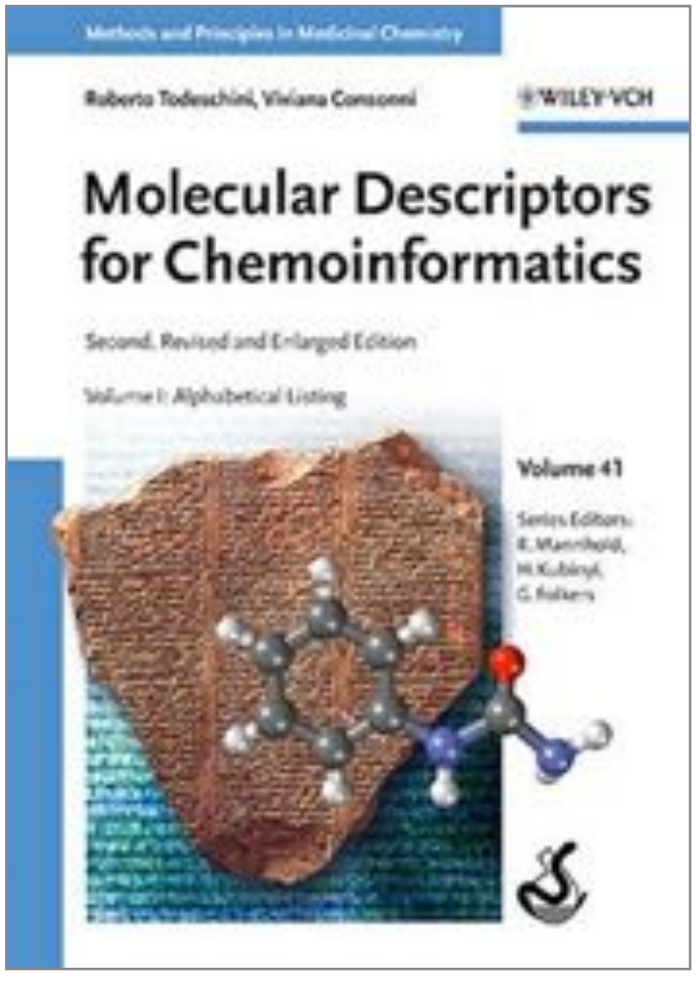
If you want ML to take into accout some factors or effects, you need to input some variables having the relevant information!

https://en.wikipedia.org/wiki/Omitted-variable_bias

Many "molecular descriptor" developed

- Quantitative Structure–Property Relationship Modeling of Diverse Materials Properties. Chem. Rev., 2012, 112 (5), pp 2889–2919

...more than 3,300 descriptors



5,270 descriptors

DRAGON 7.0

kode chemoinformatics

DScribe

https://github.com/SINGROUP/dscribe

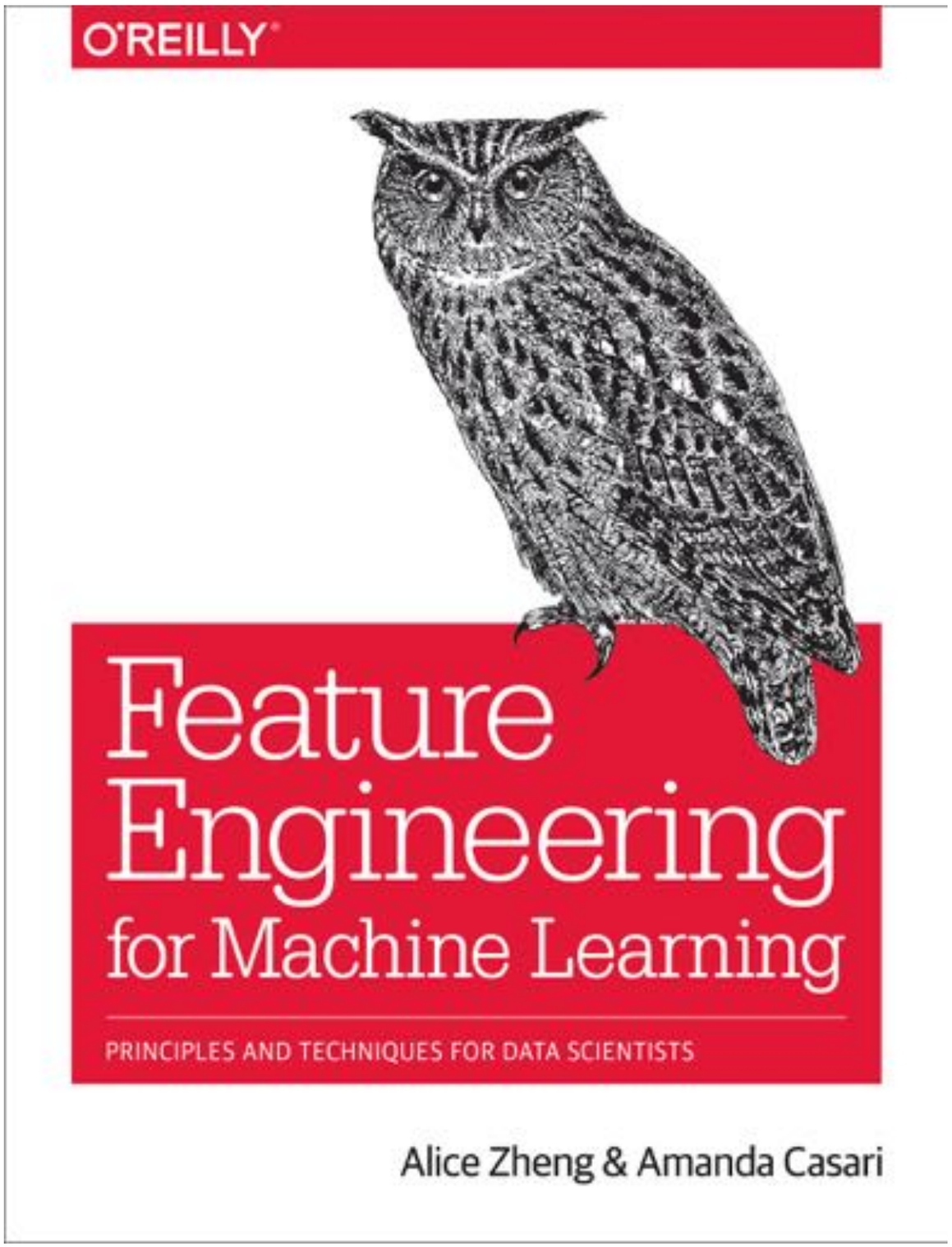
RDKit
Open-Source Cheminformatics and Machine Learning

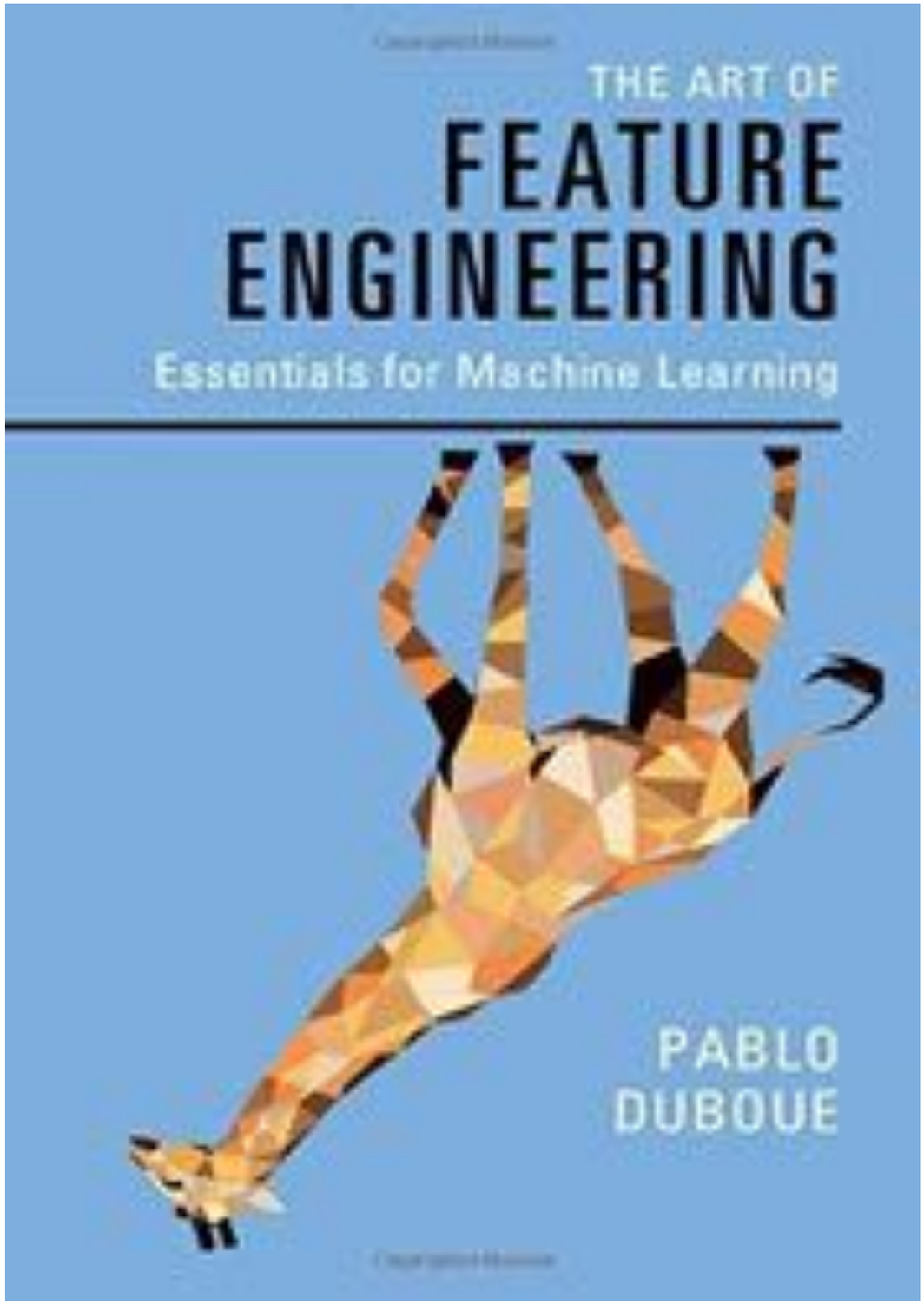
rdkit.Chem

- **Descriptors**
- **Descriptors3D**
- **GraphDescriptors**
- **Fingerprints**
- **ChemicalFeatures**
- **ChemicalForceFields**

rdkit.ML.Descriptors

Feture engineering is also very influential





Aug 26: 10:30~12:00 (90min)

1. What is "machine learning"?
2. Why does it matter to chemists?
3. Let's try it in your browser (with no setup!)

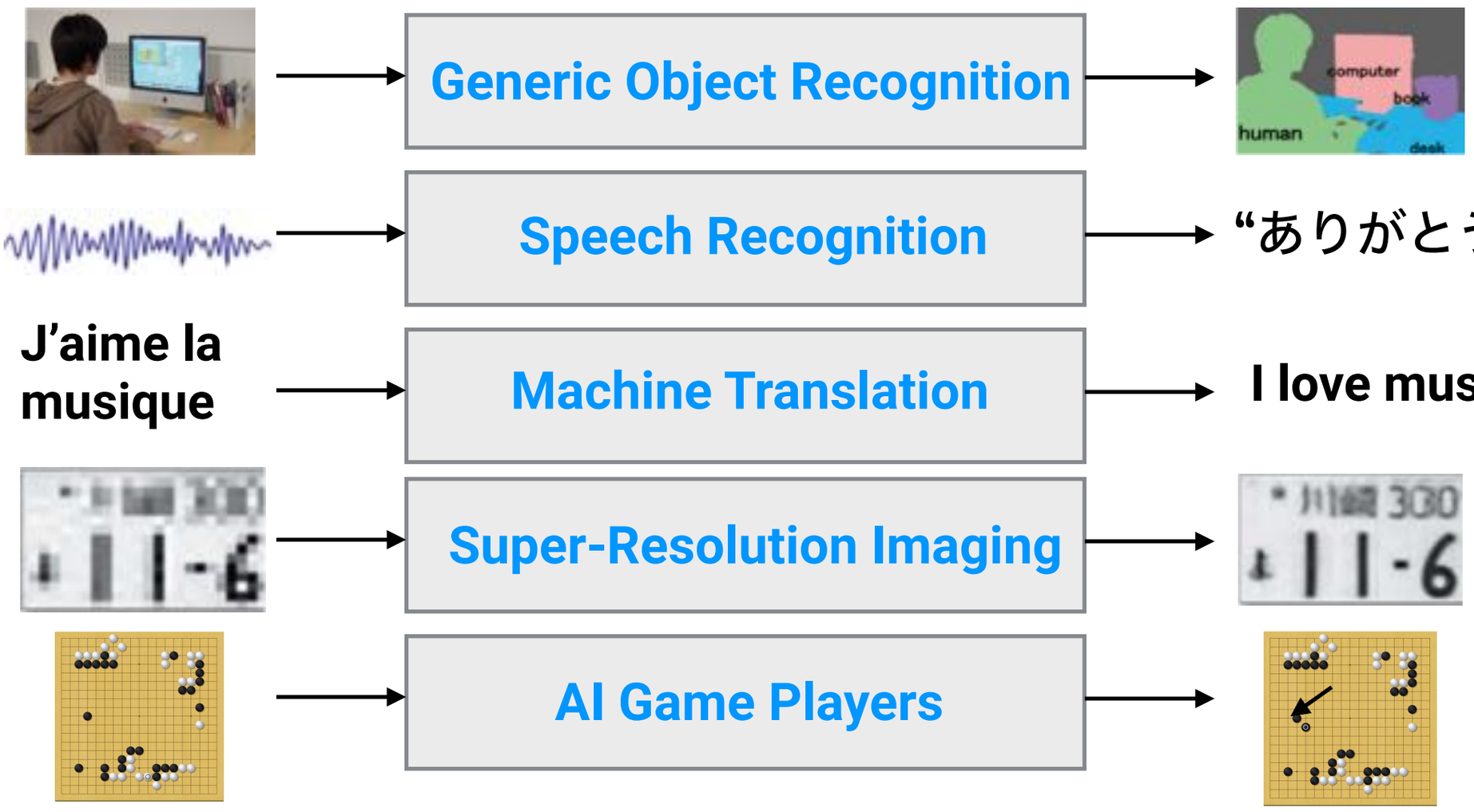
Aug 26: 13:00~14:30 (90min)

4. Five things all beginners should know
    - "The quality of your inputs decide the quality of your output"
    - Training / validation / test data
    - Tuning hyperparameters
    - Identification and design of input variables (or "descriptors")
    - "Correlation does not imply causation"
5. Standard pipeline and deep learning
6. Current efforts and future directions

"Correlation does not imply causation"

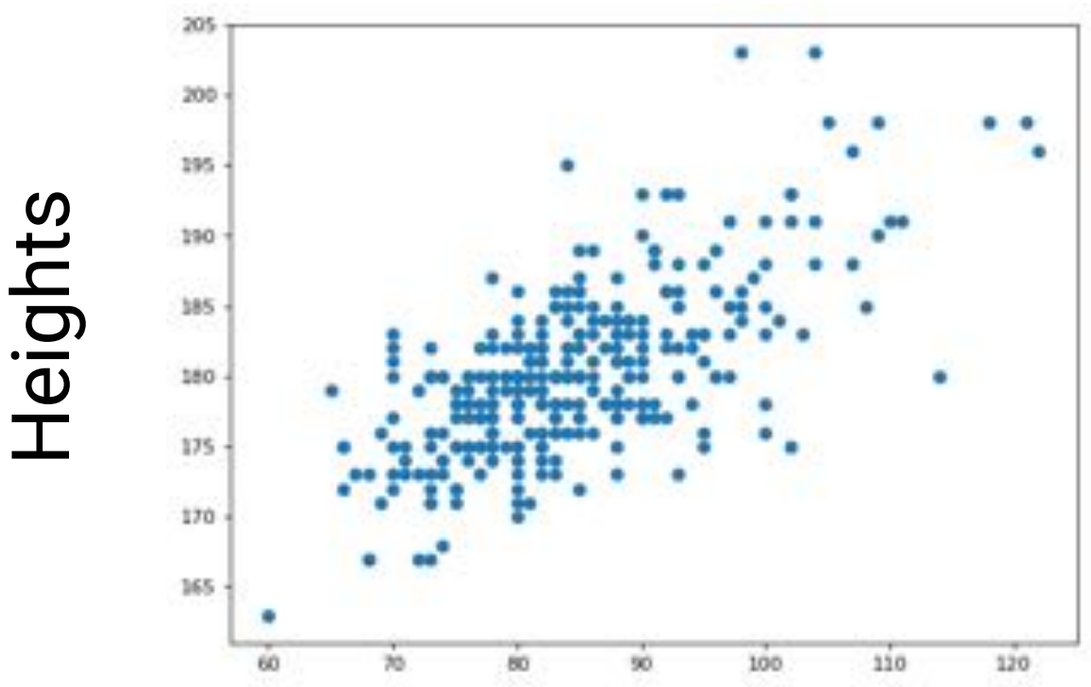
Machine learning **finds any input-output correlation in high dimensions**. And this does not directly mean any causation.

For example, current machine learning can solve those accurately, but it doesn't mean we understand the underlying process.



Stats 101 “Correlation does not imply causation”

Heights and weights of Japanese professional baseball players (handmade from 2016 website data)



This would imply “Taller players tend to be heavier”

But NOT necessarily imply “gaining weights cause tall heights”

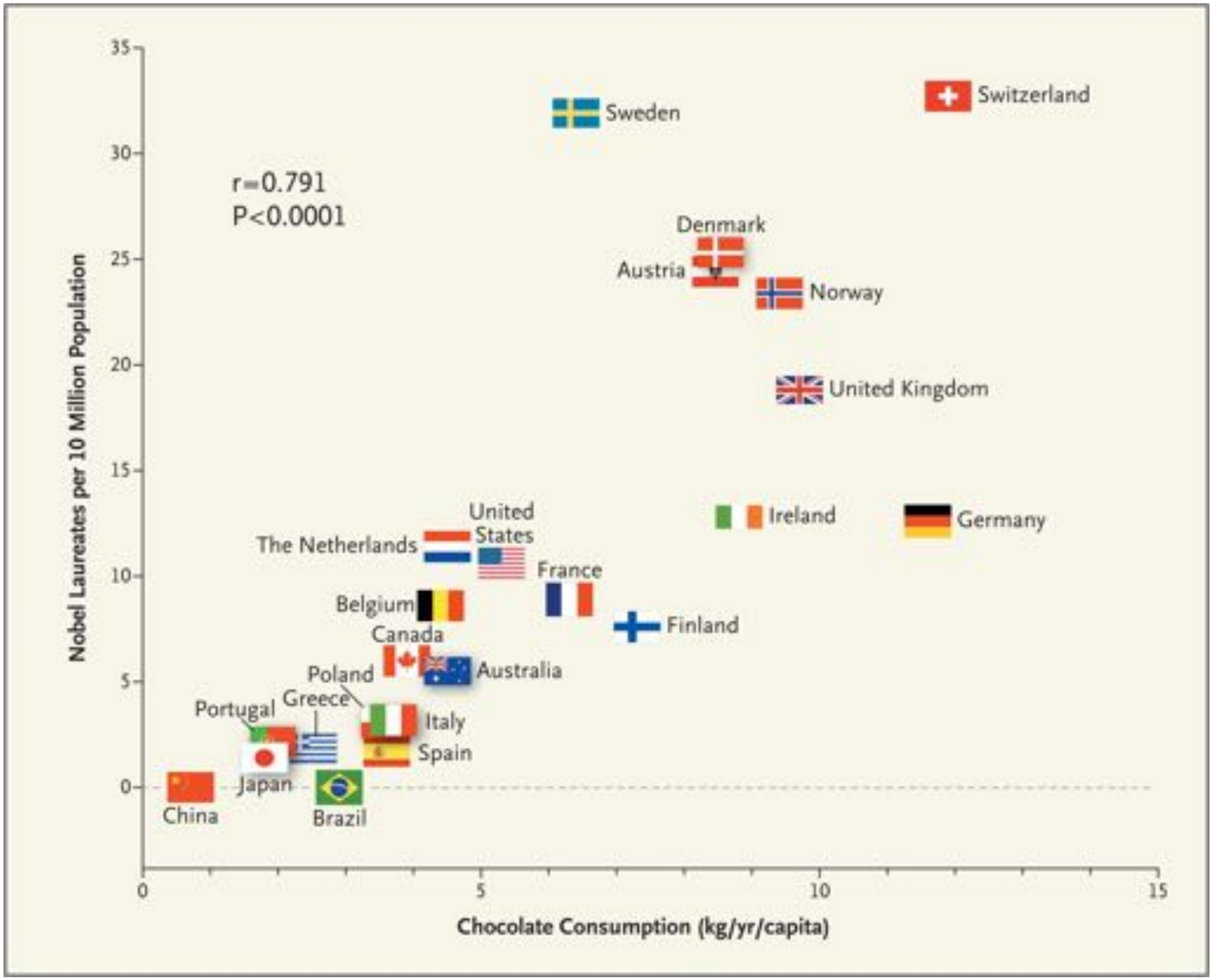
This is our domain knowledge.

We CANNOT know this from only given observational data.

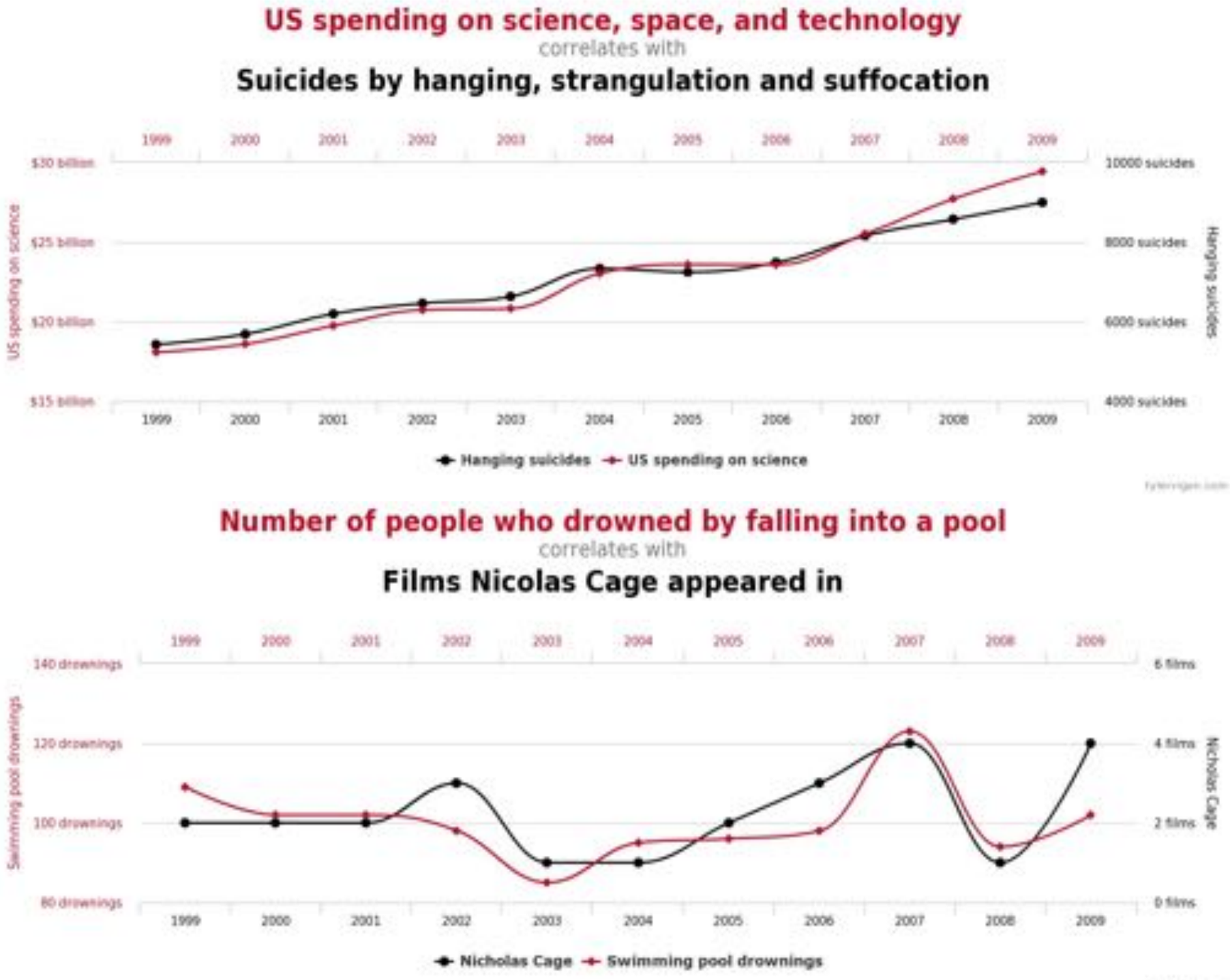
https://en.wikipedia.org/wiki/Correlation_does_not_imply_causation
https://en.wikipedia.org/wiki/Spurious_relationship
https://en.wikipedia.org/wiki/Illusory_correlation
https://en.wikipedia.org/wiki/Post_hoc_ergo_propter_hoc

N Engl J Med 2012; 367:1562-1564

One of the most prestigious medical journal with IF(2016) 72.406😳



Cannot get causation from observational data only



http://phenomena.nationalgeographic.com/2015/09/11/nick-cage-movies-vs-drownings-and-more-strange-but-spurious-correlations/

The interest of science

Related factors and their states ⟹ an unknown mechanism ⟹ observations (data)

- **Theory-driven (rational, deductive) approach**

Related factors and their states ⟹ An **explicit** model from first principles ⟹ observations (data)

Ideally, the smallest number of dominant factors

Whether or not theory is correct can be validated

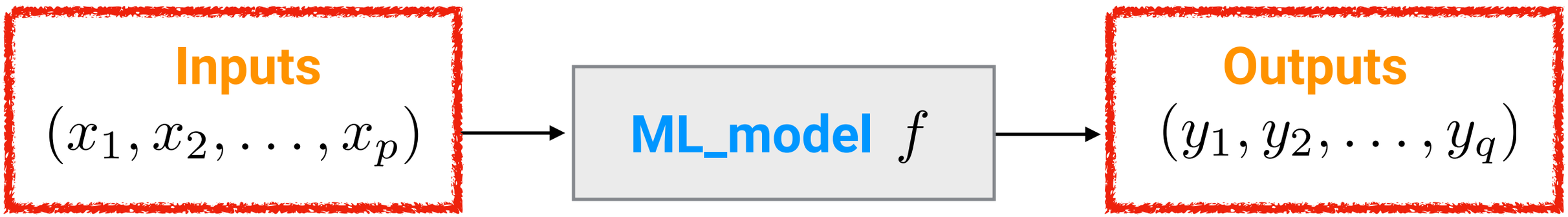
- **Data-driven (empirical, inductive) approach**

Related factors and their states ⟹ A versatile **unfixed** function that can take a variety of forms by tuning parameters ⟹ observations (data)

All potentially related factors (the number can be very large as long as they are sensible)

Function is best fitted to data by tuning parameters

When can (or should) we use ML?



- **Too unclear to be explicitly modeled**
- **Too complicated to be explicitly modeled**
- **Very time-consuming**

Some words by David Hand at KDD2018@London:

- Theory-driven models can be wrong
- But data-driven models cannot be wrong or even right. 😩
- Data-driven are not trying to describe an underlying reality.
- But are merely intended to be useful. 😆
- So they could be poor or useless, but not wrong

Aug 26: 10:30~12:00 (90min)

1. What is "machine learning"?
2. Why does it matter to chemists?
3. Let's try it in your browser (with no setup!)

Aug 26: 13:00~14:30 (90min)

4. Five things all beginners should know
    - "The quality of your inputs decide the quality of your output"
    - Training / validation / test data
    - Tuning hyperparameters
    - Identification and design of input variables (or "descriptors")
    - "Correlation does not imply causation"
5. Standard pipeline and deep learning
6. Current efforts and future directions

What ML methods should I use for my problem?

Inputs

$(x_1, x_2, \ldots, x_p)$

Outputs

$(y_1, y_2, \ldots, y_q)$

tabular data
with hand-crafted variables

x1	x2	x3	x4	x5	x6	x7	x8	x9	x10
11.0	2.67	29.0	63.54600	4.0	1.90	7.7264	203.5	8.96	9.0
9.0	2.62	27.0	58.93320	4.0	1.88	7.8810	272.5	8.86	11.0
10.0	2.90	78.0	195.07800	6.0	2.20	8.9588	113.6	21.50	8.0
11.0	3.00	79.0	196.96655	6.0	2.40	9.2255	64.6	19.30	9.0
10.0	2.90	78.0	195.07800	6.0	2.20	8.9588	113.6	21.50	9.0
...	...	...	...	...	...	...	...	...	...
9.0	2.62	27.0	58.93320	4.0	1.88	7.8810	272.5	8.86	9.0

raw signal data containing
full observed information

images

videos

audio signals

Standard pipeline

deep learning

A standard pipeline (start with smaller models!)

1. Constant prediction

sklearn.dummy.**DummyClassifier**

sklearn.dummy.**DummyRegressor**

2. Linear prediction

sklearn.linear_model.**LogisticRegression**
(penalty = "l1", "l2", "elasticnet", or "none")

sklearn.linear_model.**LinearRegression**
sklearn.linear_model.**Ridge**
sklearn.linear_model.**Lasso**
sklearn.cross_decomposition.**PLSRegression**

3. Simple nonlinear prediction

sklearn.neighbors.**KNeighborsClassifier**
sklearn.ensemble.**ExtraTreesClassifier**
sklearn.ensemble.**RandomForestClassifier**

sklearn.ensemble.**ExtraTreesRegressor**
sklearn.ensemble.**RandomForestRegressor**

4. Complex nonlinear prediction (w/ careful hyperparameter tuning)

Everything else (including deep learning) such as

sklearn.ensemble.**GradientBoostingClassifier**
sklearn.svm.**SVC**
sklearn.gaussian_process.**GaussianProcessClassifier**
sklearn.neural_network.**MLPClassifier**

sklearn.ensemble.**GradientBoostingRegressor**
sklearn.svm.**SVR**
sklearn.gaussian_process.**GaussianProcessRegressor**
sklearn.neural_network.**MLPRegressor**
sklearn.kernel_ridge.**KernelRidge**

Example

```
def validate_model(model):
  print('model:', model.__class__.__name__)
  scores = cross_validate(model, X, y, cv=5, \
                          scoring='neg_root_mean_squared_error', \
                          return_train_score=True)
  print("train RMSE %.3f" % -scores['train_score'].mean())
  print("test RMSE %.3f" % -scores['test_score'].mean())
```

```
from sklearn.dummy import DummyRegressor
validate_model(DummyRegressor(strategy='mean'))
```

```
model: DummyRegressor
train RMSE 0.813
test RMSE 0.809
```

1. Constant prediction

prediction doesn't make any sense
if the observed "test RMSE" is lower than this value

Example

2. Linear prediction

```
from sklearn.linear_model import LinearRegression
validate_model(LinearRegression())

from sklearn.linear_model import Ridge
validate_model(Ridge(alpha=0.1))

from sklearn.linear_model import Lasso
validate_model(Lasso(alpha=0.1))

from sklearn.cross_decomposition import PLSRegression
validate_model(PLSRegression(n_components=10))
```

```
model: DummyRegressor
train RMSE 0.813
test RMSE 0.809
```

```
model: LinearRegression
train RMSE 0.203
test RMSE 0.260
model: Ridge
train RMSE 0.314
test RMSE 0.391
model: Lasso
train RMSE 0.544
test RMSE 0.582
model: PLSRegression
train RMSE 0.345
test RMSE 0.423
```

3. Simple nonlinear prediction

```
from sklearn.ensemble import ExtraTreesRegressor
validate_model(ExtraTreesRegressor(n_estimators=100))

from sklearn.ensemble import RandomForestRegressor
validate_model(RandomForestRegressor(n_estimators=100))
```

```
model: ExtraTreesRegressor
train RMSE 0.000
test RMSE 0.219
model: RandomForestRegressor
train RMSE 0.082
test RMSE 0.230
```

Example

4. Complex nonlinear prediction (w/ careful hyperparameter tuning)

```
from sklearn.ensemble import GradientBoostingRegressor
validate_model(GradientBoostingRegressor(n_estimators=100,\
                                         max_leaf_nodes=8))

from sklearn.svm import SVR
validate_model(SVR(kernel='rbf', C=2.0, gamma=0.1))

from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF
validate_model(GaussianProcessRegressor(kernel=RBF(10.0)))

from sklearn.neural_network import MLPRegressor
validate_model(MLPRegressor(hidden_layer_sizes=(10, 10, 5, ), \
                            learning_rate_init=0.001, \
                            max_iter=1000, tol=1e-3))

from sklearn.kernel_ridge import KernelRidge
validate_model(KernelRidge(kernel='rbf', alpha=0.5))
```

```
model: DummyRegressor
train RMSE 0.813
test RMSE 0.809

model: LinearRegression
train RMSE 0.203
test RMSE 0.260

model: GradientBoostingRegressor
train RMSE 0.038
test RMSE 0.148
```

😆 **we used this in the paper**

```
model: SVR
train RMSE 0.101
test RMSE 0.809
```

😱

```
model: GaussianProcessRegressor
train RMSE 0.000
test RMSE 1.023
```

😱

```
model: MLPRegressor
train RMSE 1.138
test RMSE 1.329
```

😱

```
model: KernelRidge
train RMSE 0.756
test RMSE 2.260
```

😱

We need to carefully tune hyperparameters, and it is often difficult or time-consuming to beat the previous models.

Here, preprocessing is required to improve these

Example

4. Complex nonlinear prediction (w/ careful hyperparameter tuning)

Anyway, these complex models usually requires careful preprocessing and hyperparameter tuning, which also requires knowledge on each algorithm...

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

def wrap(model):
  return make_pipeline(StandardScaler(), model)
```

```
from sklearn.svm import SVR
validate_model(wrap(SVR(kernel='rbf', C=10.0, gamma=0.1)))

from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF
validate_model(wrap(GaussianProcessRegressor(kernel=RBF(0.5))))

from sklearn.neural_network import MLPRegressor
validate_model(wrap(MLPRegressor(hidden_layer_sizes=(10, 10, 5, 5, ), \
                                 learning_rate_init=0.01, \
                                 max_iter=1000, tol=1e-3)))

from sklearn.kernel_ridge import KernelRidge
validate_model(wrap(KernelRidge(kernel='rbf', alpha=0.01)))
```

```
model: LinearRegression
train RMSE 0.203
test RMSE 0.260

model: SVR
train RMSE 0.101
test RMSE 0.809
model: GaussianProcessRegressor
train RMSE 0.000
test RMSE 1.023
model: MLPRegressor
train RMSE 1.138
test RMSE 1.329
model: KernelRidge
train RMSE 0.756
test RMSE 2.260

model: Pipeline
train RMSE 0.091
test RMSE 0.390
model: Pipeline
train RMSE 0.000
test RMSE 0.330
model: Pipeline
train RMSE 0.262
test RMSE 0.417
model: Pipeline
train RMSE 0.046
test RMSE 0.344
```

a bit improved (but still worse than linear regression)

"Deep learning"?

Inputs

$(x_1, x_2, \ldots, x_p)$

ML_model f

Outputs

$(y_1, y_2, \ldots, y_q)$

tabular data
with hand-crafted variables

x1	x2	x3	x4	x5	x6	x7	x8	x9	x10
11.0	2.67	29.0	63.54600	4.0	1.90	7.7264	203.5	8.96	9.0
9.0	2.62	27.0	58.93320	4.0	1.88	7.8810	272.5	8.86	11.0
10.0	2.90	78.0	195.07800	6.0	2.20	8.9588	113.6	21.50	8.0
11.0	3.00	79.0	196.96655	6.0	2.40	9.2255	64.6	19.30	9.0
10.0	2.90	78.0	195.07800	6.0	2.20	8.9588	113.6	21.50	9.0
...	...	...	...	...	...	...	...	...	...
9.0	2.62	27.0	58.93320	4.0	1.88	7.8810	272.5	8.86	9.0

Standard pipeline

raw signal data containing
full observed information

images

videos

audio signals

deep learning

"Deep learning" = ML by "deep" neural networks

Inputs

$(x_1, x_2, \ldots, x_p)$

Outputs

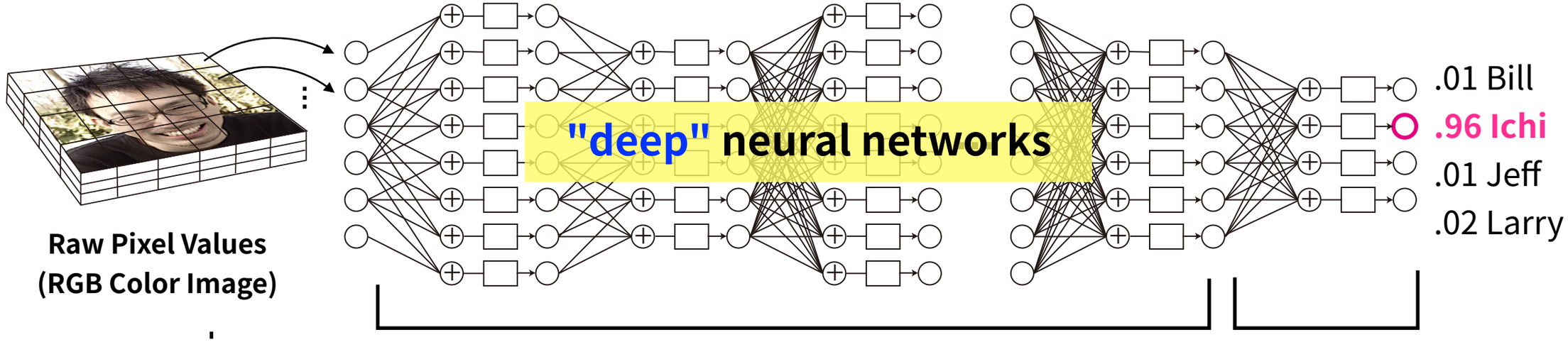
$(y_1, y_2, \ldots, y_q)$

The **adjective "deep"** in deep learning comes from **the use of many layers in the network.**

End
(what you get)

"End to End" (fully data-driven)

End
(what you want)



Raw Pixel Values
(RGB Color Image)

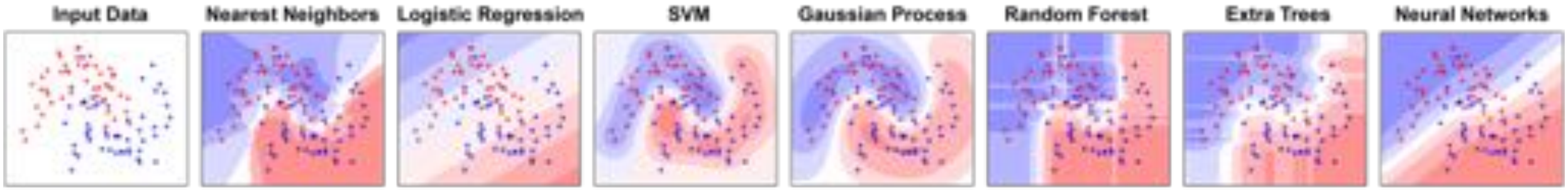
standard pipeline

multi-stage "transformations"
to better representation for the task

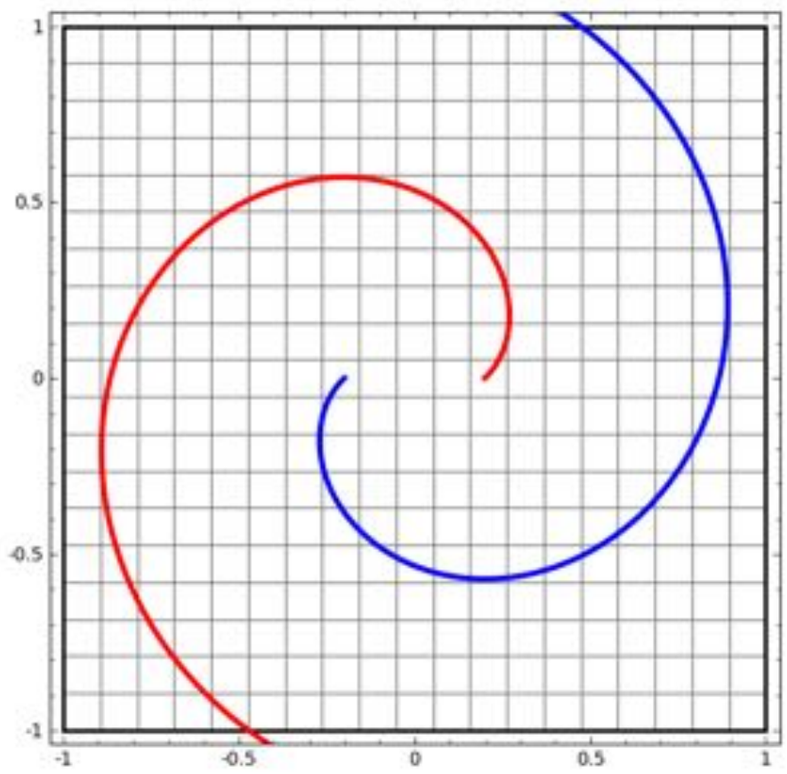
manually design and engineer the input variables or "descriptors"

classifier or regressor

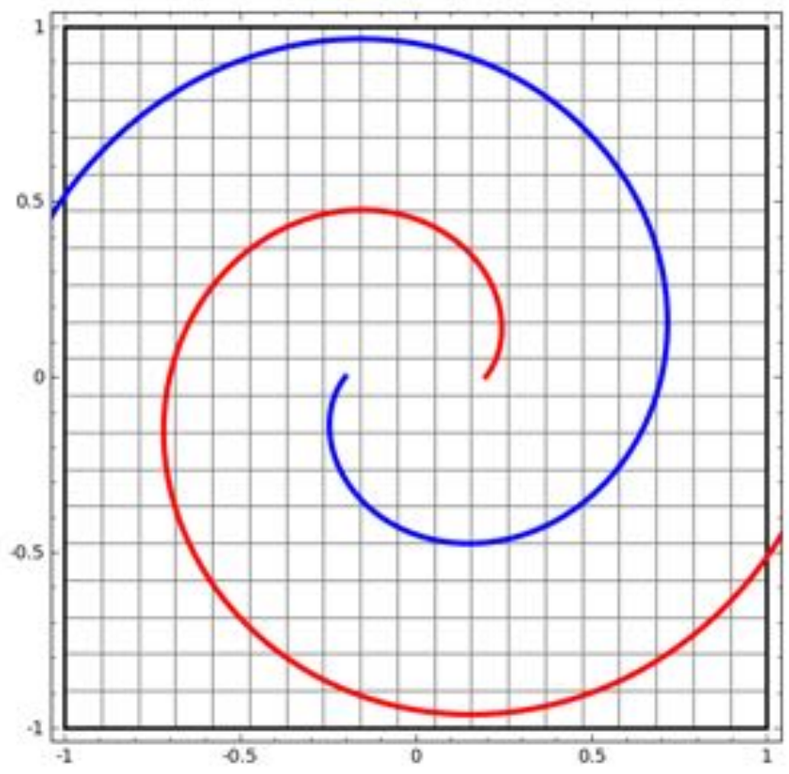
Deep learning try to learn better "representation"



try to learn any "(linearly) separable" representation

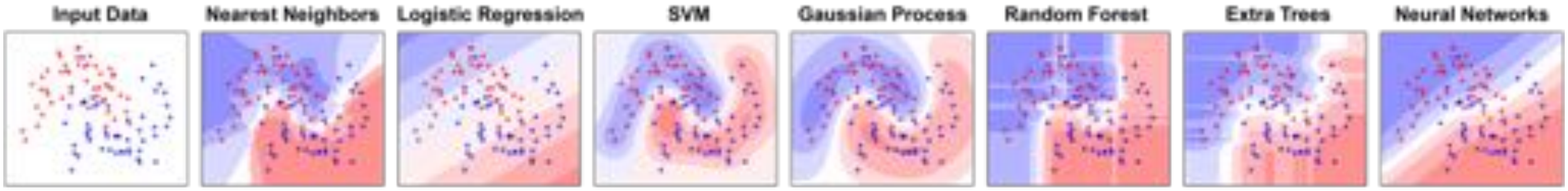


but not always successful... (this task is very difficult)

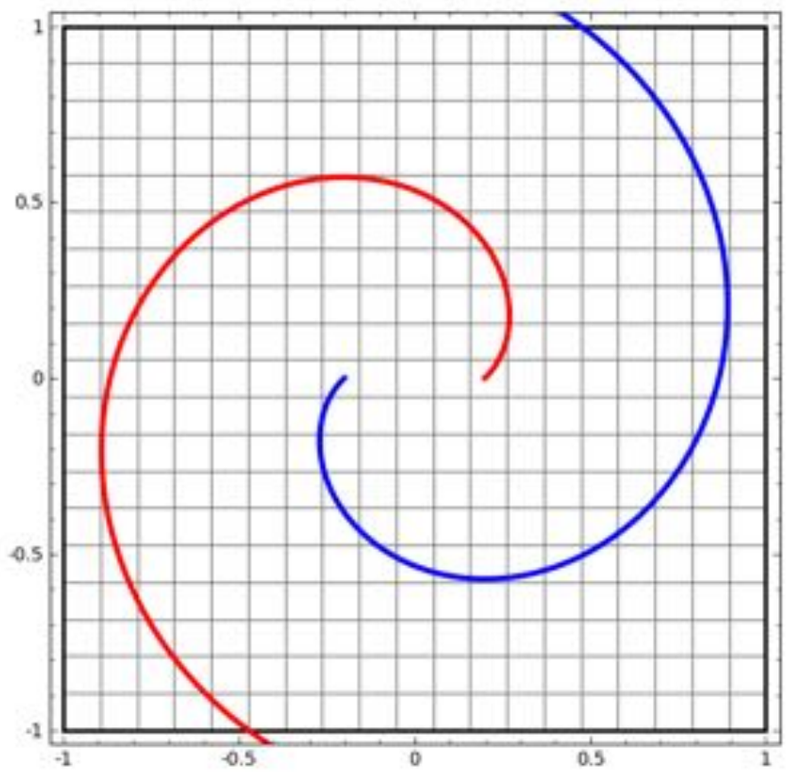


https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

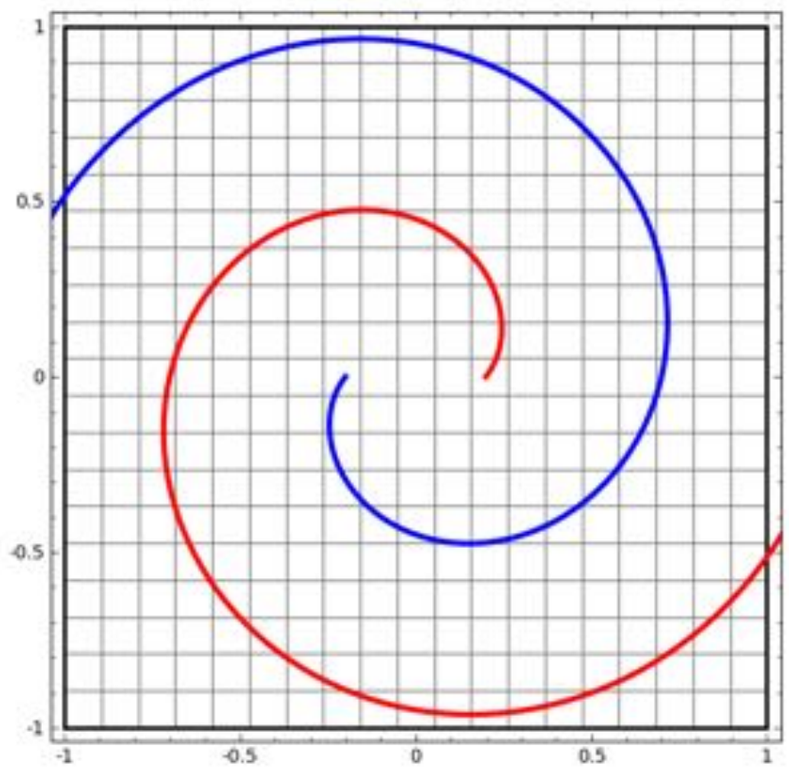
Deep learning try to learn better "representation"



try to learn any "(linearly) separable" representation

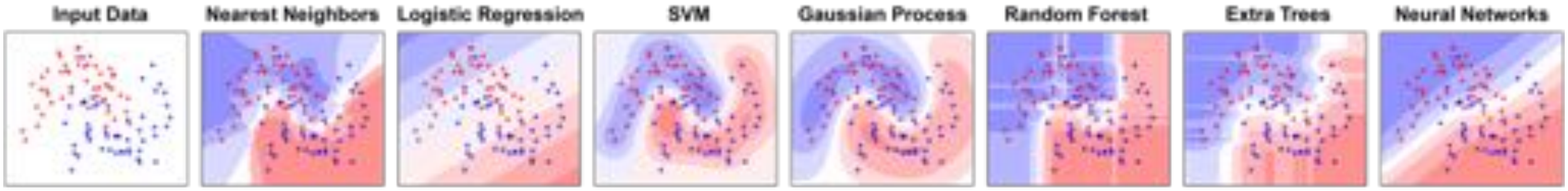


but not always successful... (this task is very difficult)

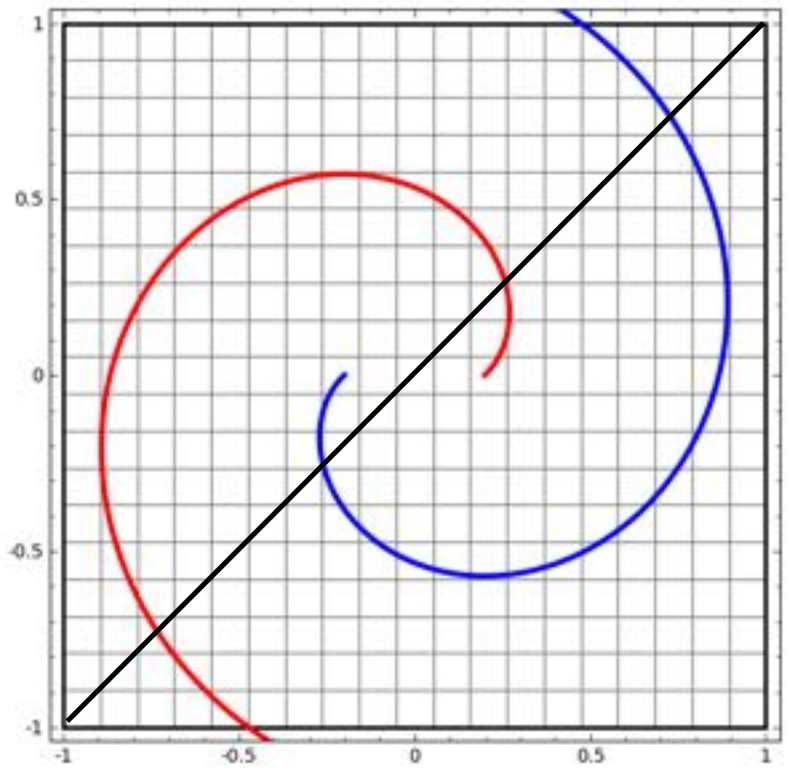


https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

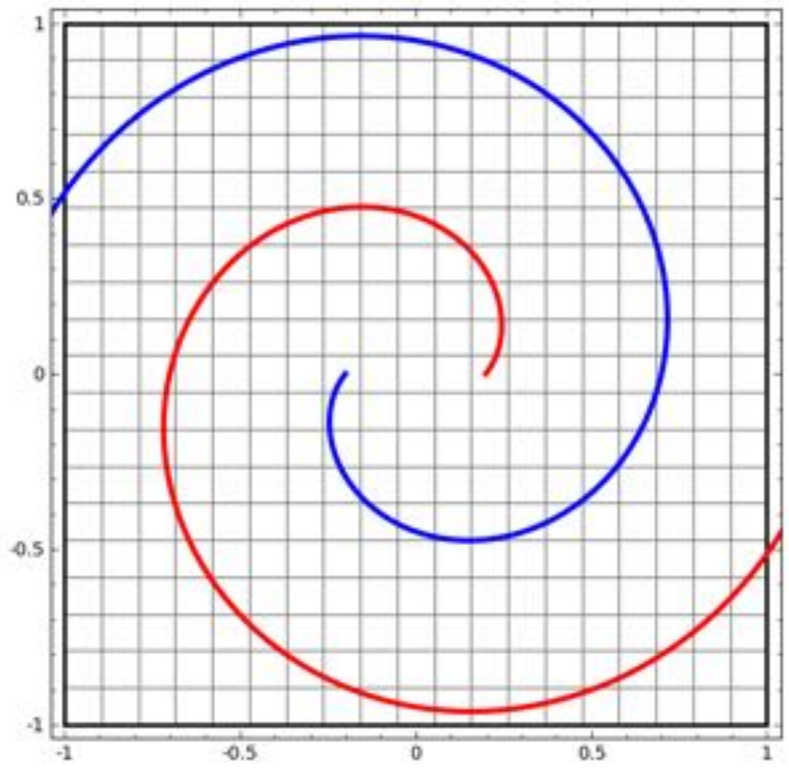
Deep learning try to learn better "representation"



try to learn any "(linearly) separable" representation

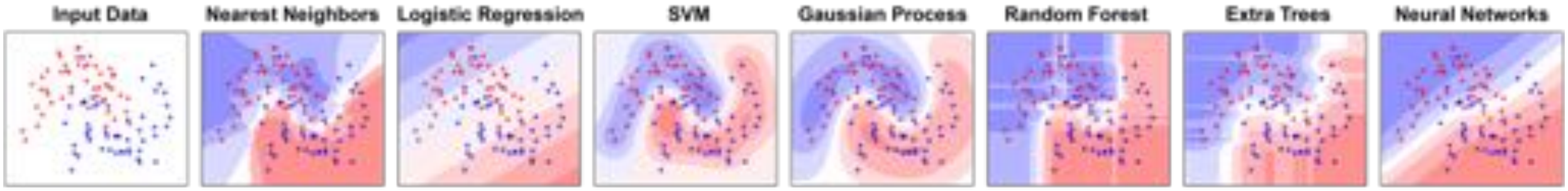


but not always successful... (this task is very difficult)

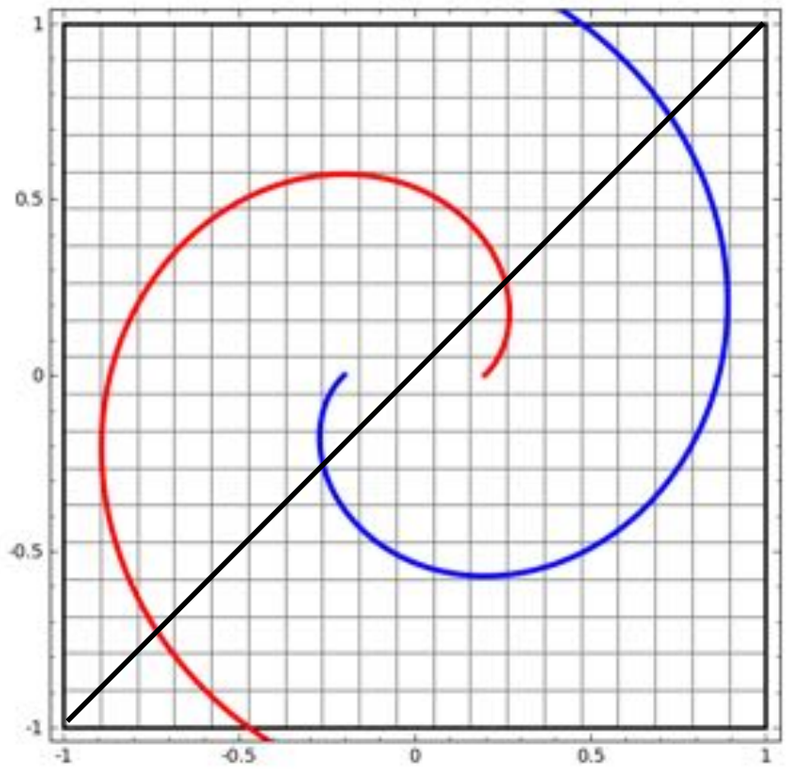


https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

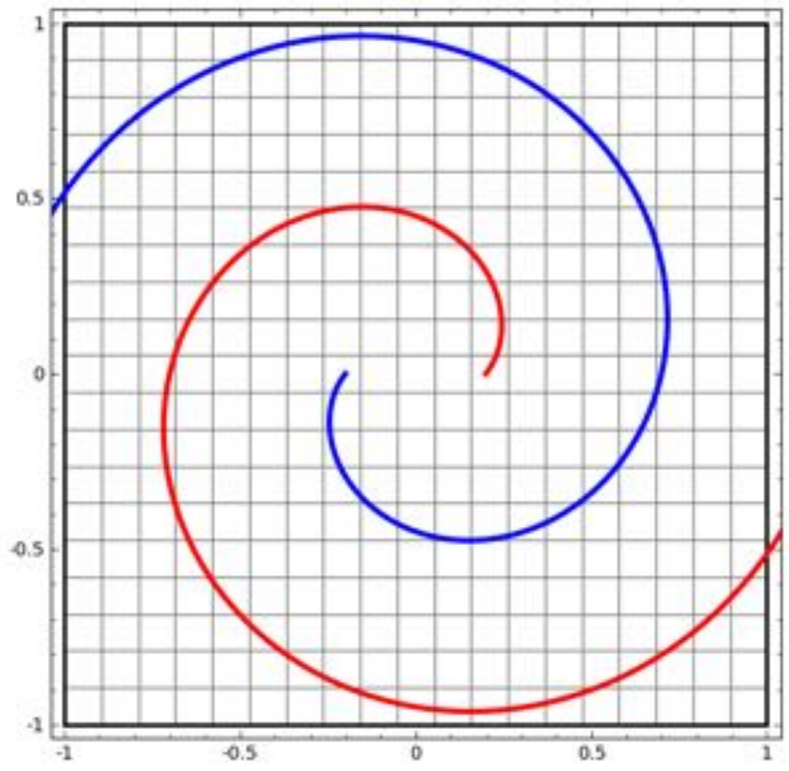
Deep learning try to learn better "representation"



try to learn any "(linearly) separable" representation



but not always successful... (this task is very difficult)



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

Example

Inputs

$(x_1, x_2, \ldots, x_{784})$

(784 real numbers)

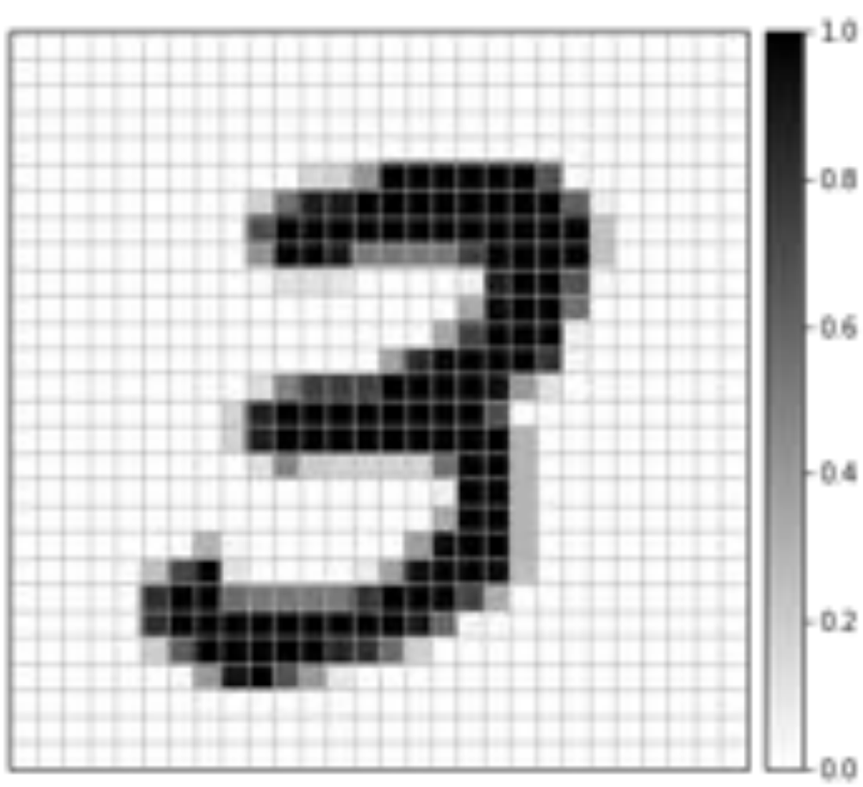
ML model (surface)

Outputs

y

(10 real numbers)

An 28 x 28 (=784) pixel image



Probabilities for 0,1,2,...,9

0 1 2 3 4
[0.0, 0.0, 0.0, **0.9**, 0.0,
5 6 7 8 9
0.0, 0.0, 0.0, **0.1**, 0.0]

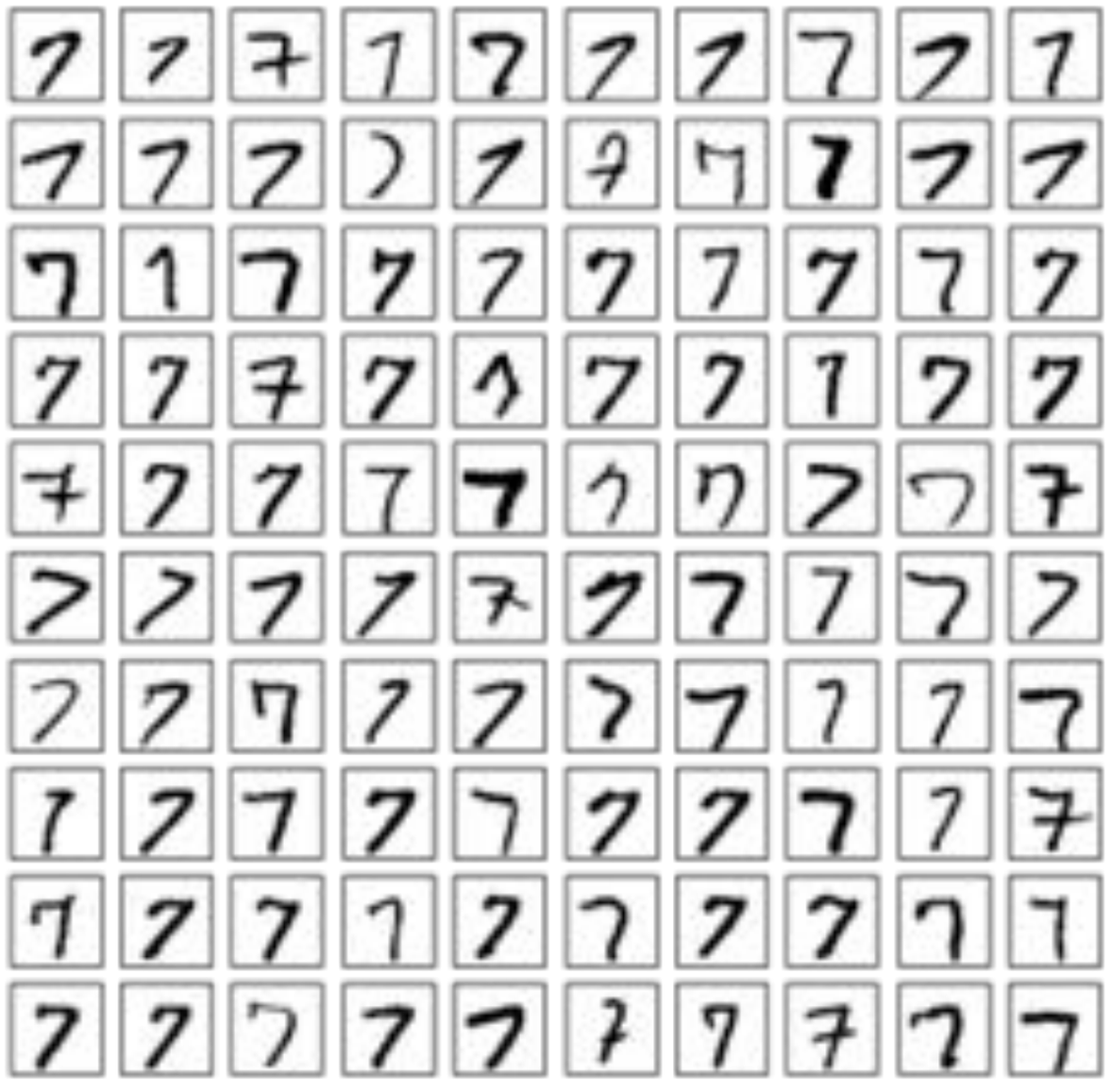
Just fit a 10-dimensional-valued function in 784-dimensional space!

Find a nice mapping $f : \mathbb{R}^{784} \to \mathbb{R}^{10}$

Example: MNIST handwritten digits classification

```
from tensorflow import keras
from tensorflow.keras import layers
```

```
(X_train, y_train), (X_test, y_test) = keras.datasets.mnist.load_data()
```



tensorflow.keras has a scikit-like API

```
input_shape = X_train.shape[1:]

model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(10, activation="softmax"),
    ]
)

model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
```

```
model.fit(X_train, y_train, \
          batch_size=128, epochs=15, \
          validation_split=0.1)
```

```
y_pred = model.predict(X_test)
```

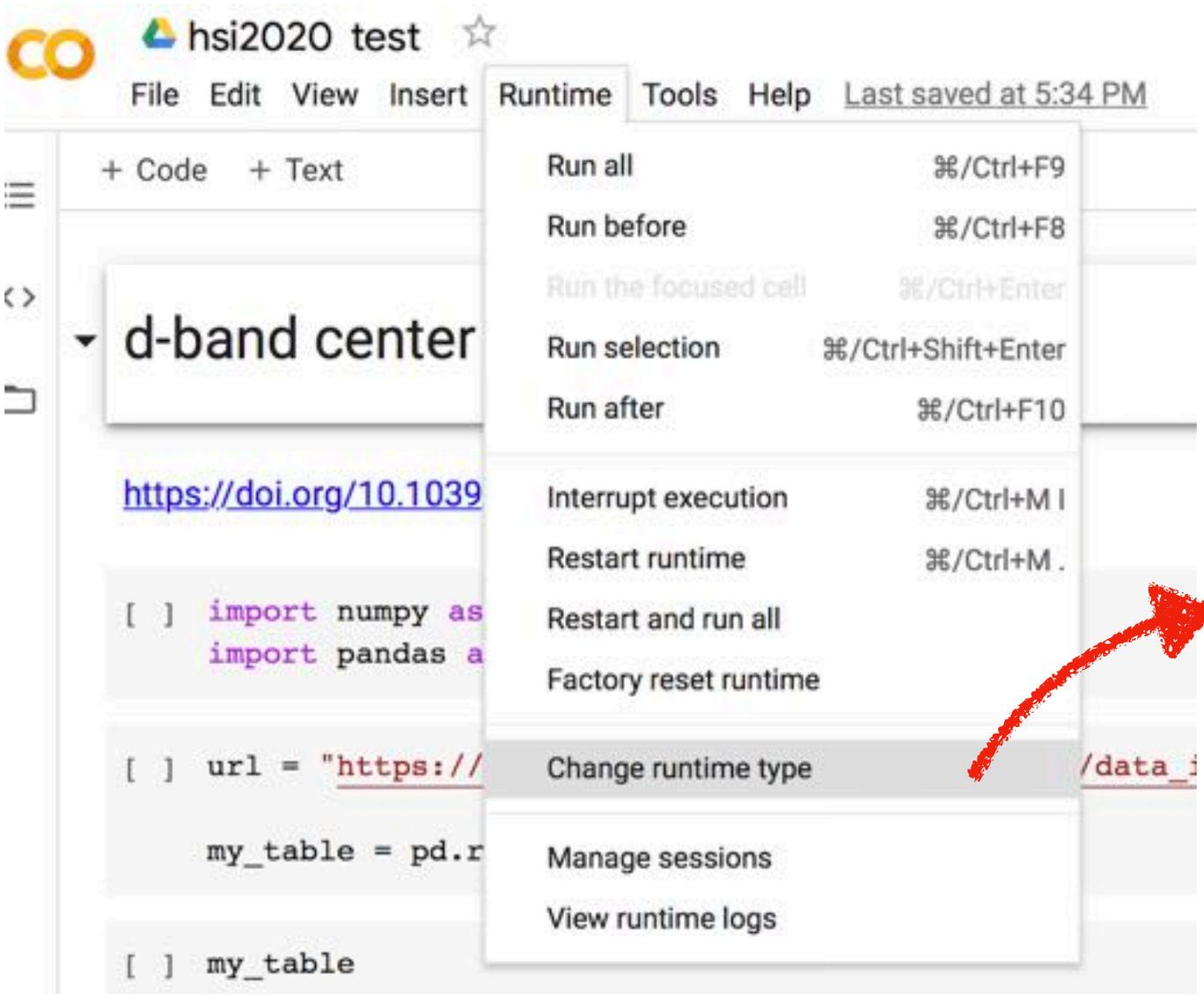
```
accuracy_score(
    np.argmax(y_test, axis=1),
    np.argmax(y_pred, axis=1))
```

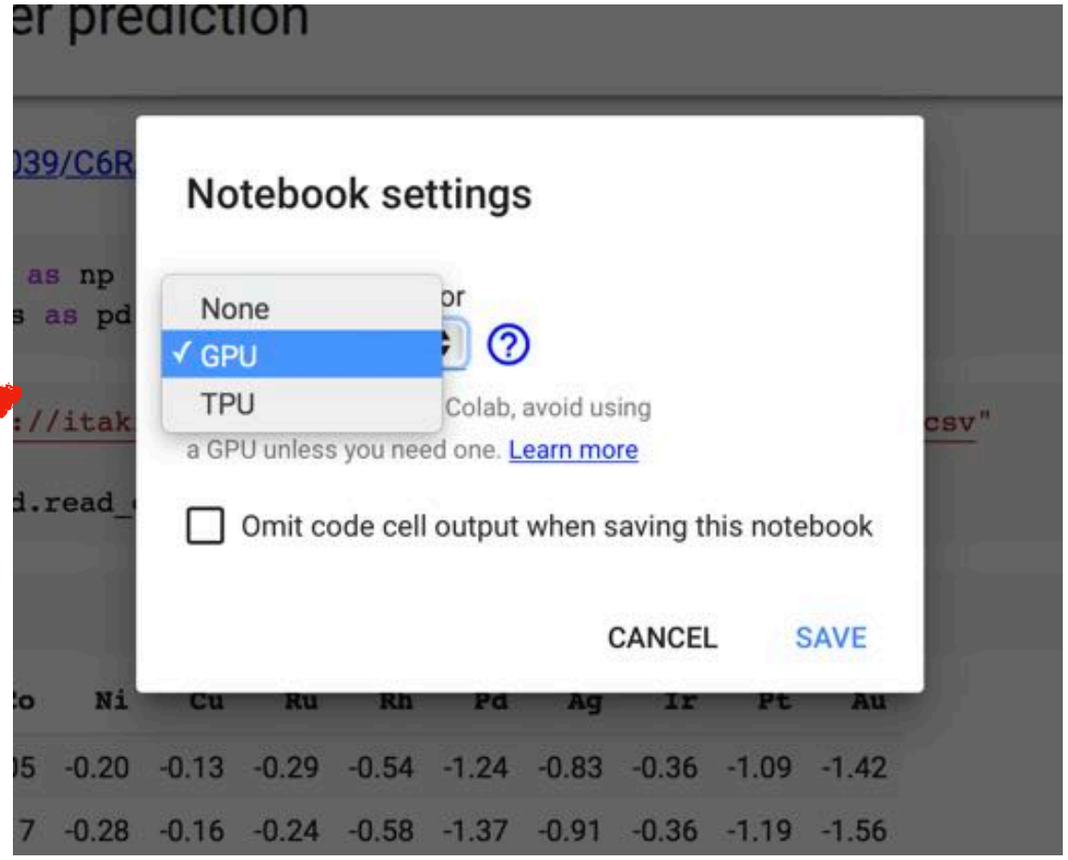
```
0.9925
```

For the full example, try the colab code

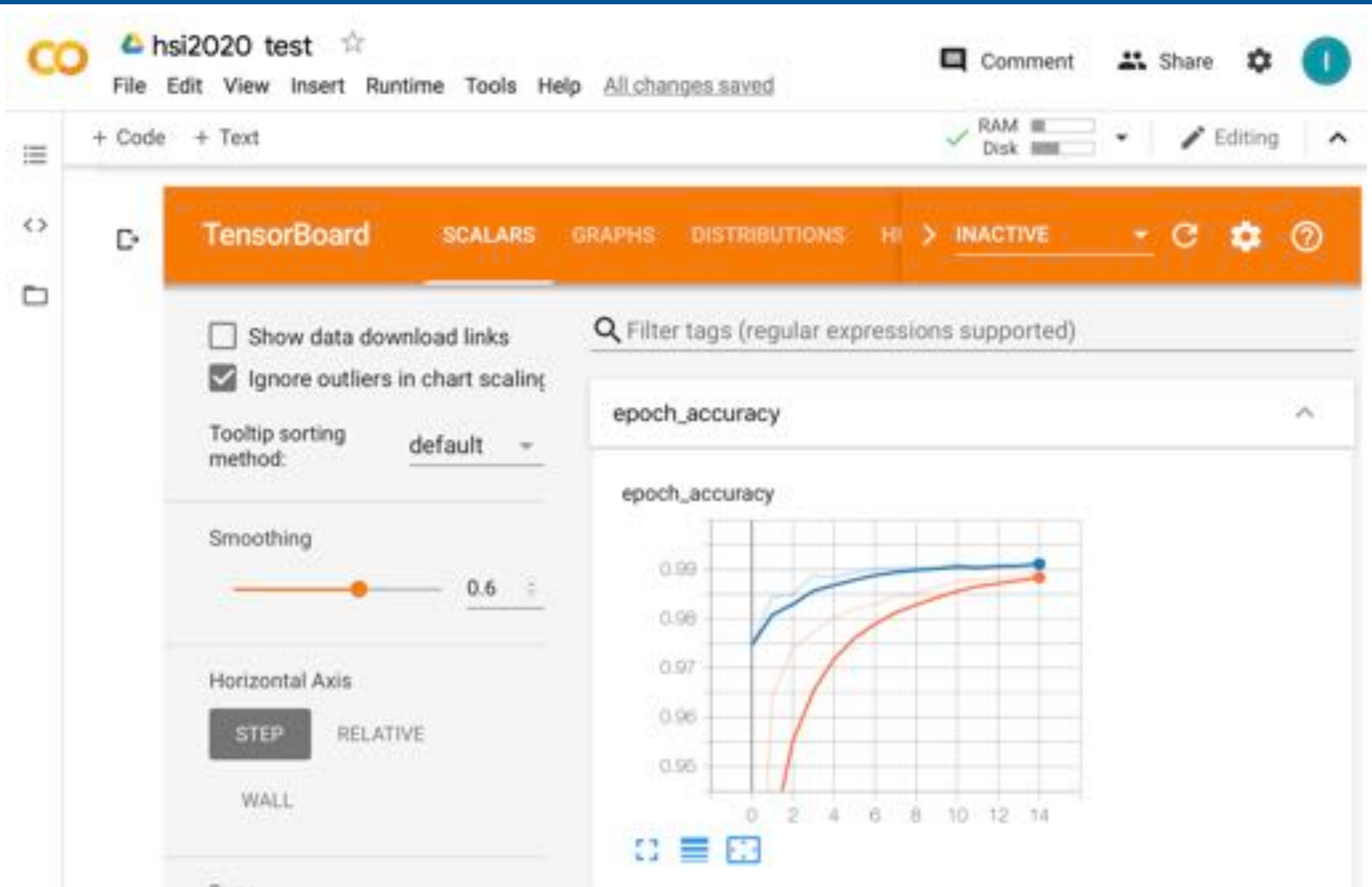
https://colab.research.google.com/drive/1PxRVbn9JxYPdwwzmJvxXMDgZOa_Gjx_V?usp=sharing

Notice: specify "GPU" (or "TPU") for this example though CPUs also work.





Tensorboard



Also note that the standard pipeline also works

Deep learning accuracy

```
accuracy_score(
    np.argmax(y_test, axis=1),
    np.argmax(y_pred, axis=1))
```

```
0.9925
```

```
CPU times: user 30.1 s, sys: 5.77 s, total: 35.9 s
Wall time: 36.2 s
```

time on a GPU

```
%%time
from sklearn.dummy import DummyClassifier
model = DummyClassifier(strategy='stratified')
validate_model(model)
```

```
test accuracy: 0.1039
CPU times: user 11.7 ms, sys: 854 µs, total: 12.5 ms
Wall time: 15.8 ms
```

```
%%time
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(penalty='l2', C= 1.0)
validate_model(model)
```

```
test accuracy: 0.9255
CPU times: user 52.4 s, sys: 6.01 s, total: 58.4 s
Wall time: 30 s
```

```
%%time
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
validate_model(model)
```

```
test accuracy: 0.9688
CPU times: user 14min 17s, sys: 176 ms, total: 14min 17s
Wall time: 14min 17s
```

```
%%time
from sklearn.ensemble import ExtraTreesClassifier
model = ExtraTreesClassifier()
validate_model(model)
```

```
test accuracy: 0.9726
CPU times: user 28 s, sys: 88.3 ms, total: 28.1 s
Wall time: 28.1 s
```

```
from lightgbm import LGBMClassifier
```

```
%%time
model = LGBMClassifier(n_estimators=200)
validate_model(model)
```

```
test accuracy: 0.9792
CPU times: user 11min 45s, sys: 907 ms, total: 11min 46s
Wall time: 5min 59s
```

time on a CPU

Aug 26: 10:30~12:00 (90min)

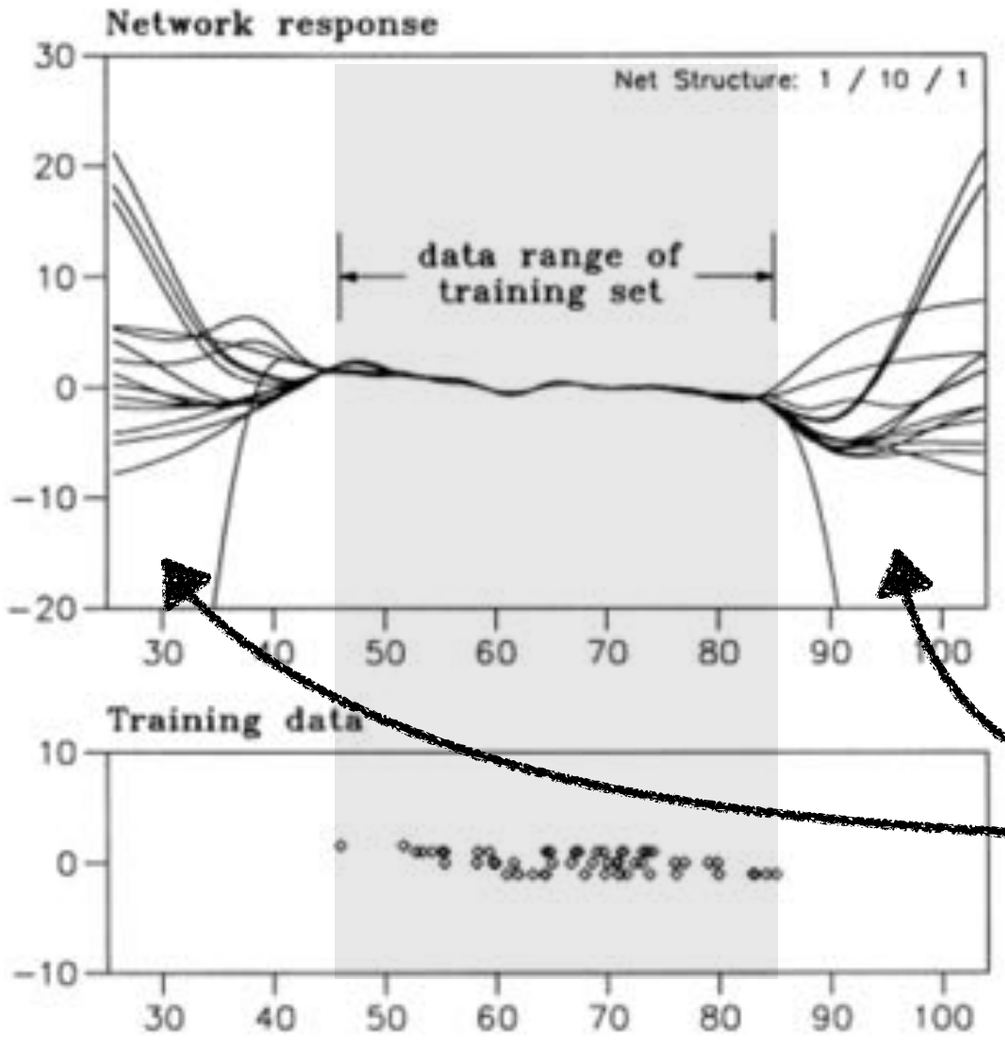
1. What is "machine learning"?
2. Why does it matter to chemists?
3. Let's try it in your browser (with no setup!)

Aug 26: 13:00~14:30 (90min)

4. Five things all beginners should know
    - "The quality of your inputs decide the quality of your output"
    - Training / validation / test data
    - Tuning hyperparameters
    - Identification and design of input variables (or "descriptors")
    - "Correlation does not imply causation"
5. Standard pipeline and deep learning
6. Current efforts and future directions

ML is just representative of the training data

Highly Inaccurate Model Predictions from Extrapolation (Lohninger 1999)



"Beware of the perils of extrapolation, and understand that ML algorithms build models that are representative of the available training samples."

No reliable prediction would be made for the outside of the training data region (i.e. "extrapolation")

☺️**Best Practices for Machine Learning Applications**
https://support.sas.com/resources/papers/proceedings16/SAS2360-2016.pdf

Expect something not in the training data?

In science, we often seek for something not in the currently observed data. Simply put, it should be a 'scientific discovery'.

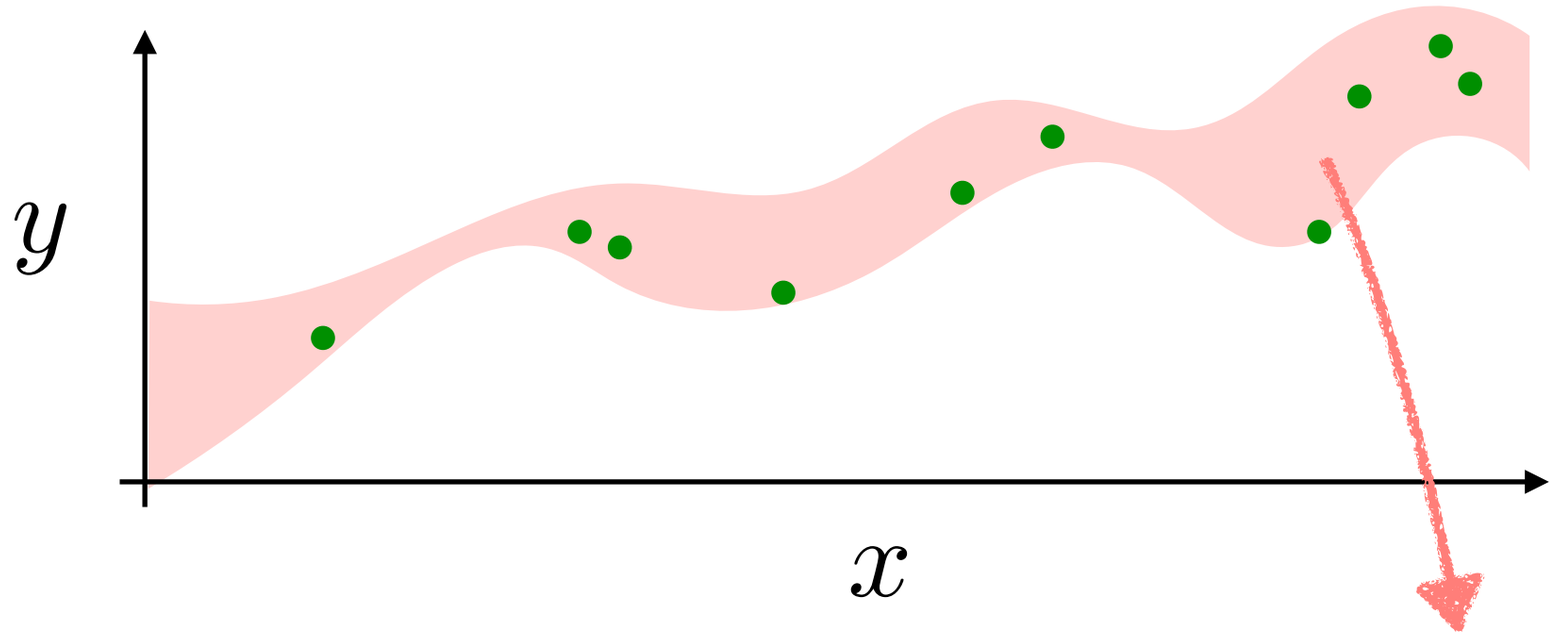
- Revolutionary materials better than any materials we have now
- Surprisingly effective cancer drugs that no one has ever seen before
- Innovative business strategy that has never tried before
- Astonishing scientific law or theory that no one has ever discovered
- Game playing strategy that is stronger than anyone existed before

But ML is just representative of the training data, which obviously sounds quite unlikely to discover something as-yet-unidentified.

Then, what should we do!? 😫

The exploration-exploitation trade-off

To get better y, we need to balance both actively collecting more data and narrowing the plausible region using the current data.

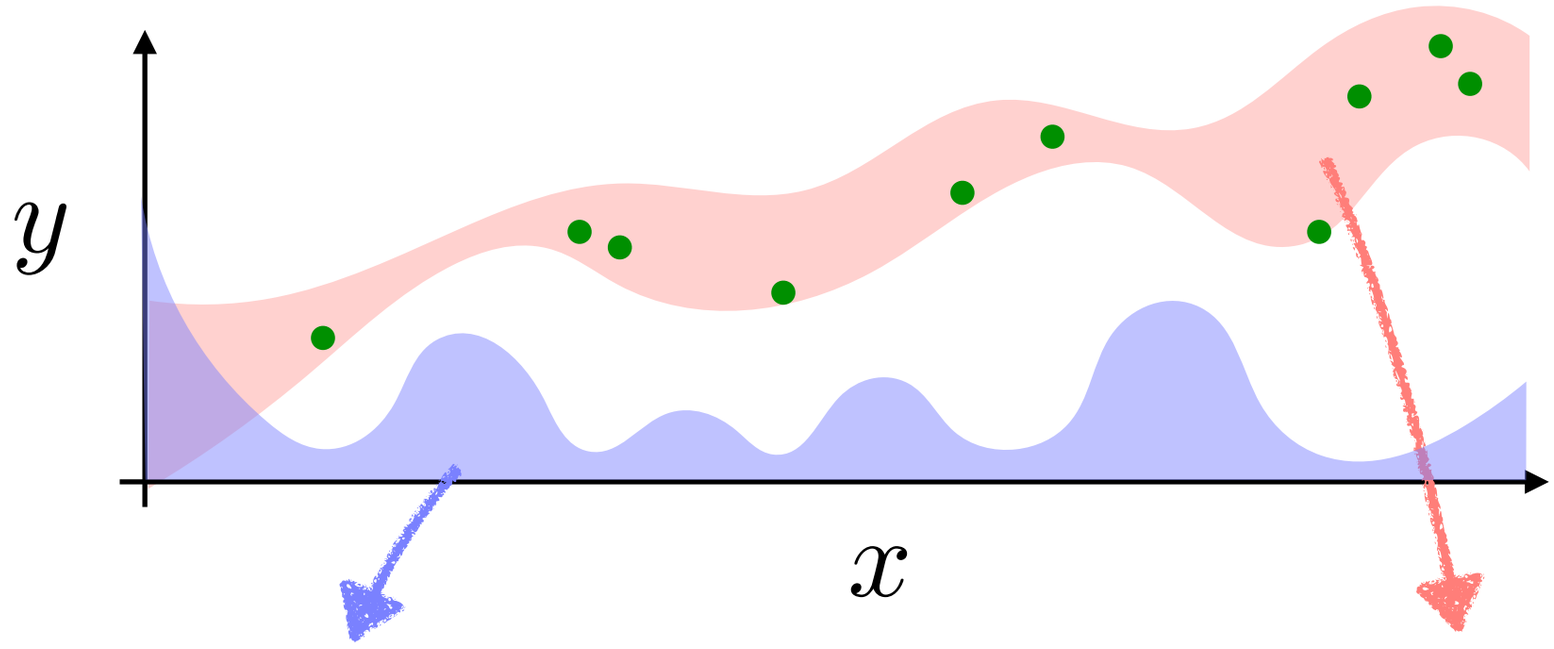


“Exploitation”

We exploit the current data to spot the plausible region having better y

The exploration-exploitation trade-off

To get better y, we need to balance both actively collecting more data and narrowing the plausible region using the current data.



"Exploration"

We explore regions potentially having better y but not yet covered by the current data.

"Exploitation"

We exploit the current data to spot the plausible region having better y

Inverse design

- molecule
- material
- reaction
- process

:

x

relevant representation

“machine learning”

$$\hat{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x})$$

$\hat{y}$

desirable properties

“machine teaching”

Bayesian optimization / Black-box optimization / Sequential design of experiments / Model-based optimization

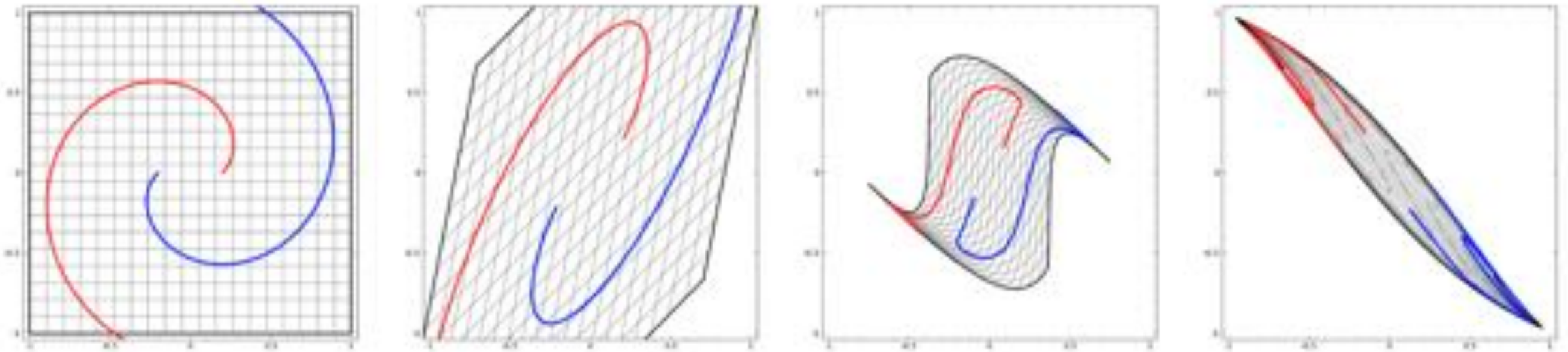
“television”

??

Beware: Inverse is not unique

Representatin learning

Yes, machine learning is interpolative, but interporative under the given representation



input raw representation | transformed representation 1 | transformed representation 2 | transformed representation 3

- **interpolation in transformed representation can be somewhat "extrapolative" in the original input representation**
- **there might exist nice latent representation for generalization**
- **this is also why "feature engineering" is so influential in practice**

"Autonomous Discovery in the Chemical Sciences"

Angew. Chem. Int. Ed. **2020**, *59*, 2 – 38

Machine Learning

How to cite:
International Edition: doi.org/10.1002/anie.201909987
German Edition: doi.org/10.1002/ange.201909987

Autonomous Discovery in the Chemical Sciences Part I: Progress

Connor W. Coley, Natalie S. Eyke, and Klavs F. Jensen**

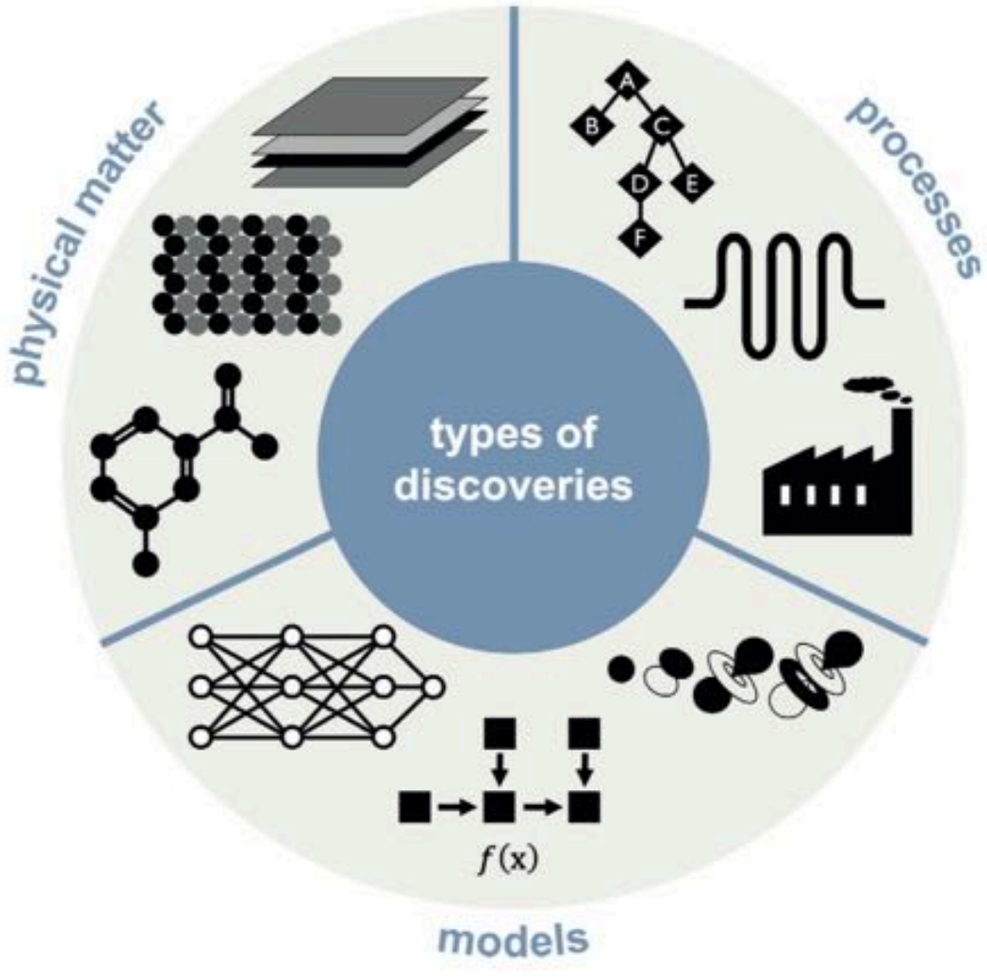


Figure 1. The three broad categories of discovery described in this Review: physical matter, processes, and models.

Angew. Chem. Int. Ed. **2020**, *59*, 2 – 25

Computer Chemistry

How to cite:
International Edition: doi.org/10.1002/anie.201909989
German Edition: doi.org/10.1002/ange.201909989

Autonomous Discovery in the Chemical Sciences Part II: Outlook

Connor W. Coley, Natalie S. Eyke, and Klavs F. Jensen**

Angewandte International Edition Chemie

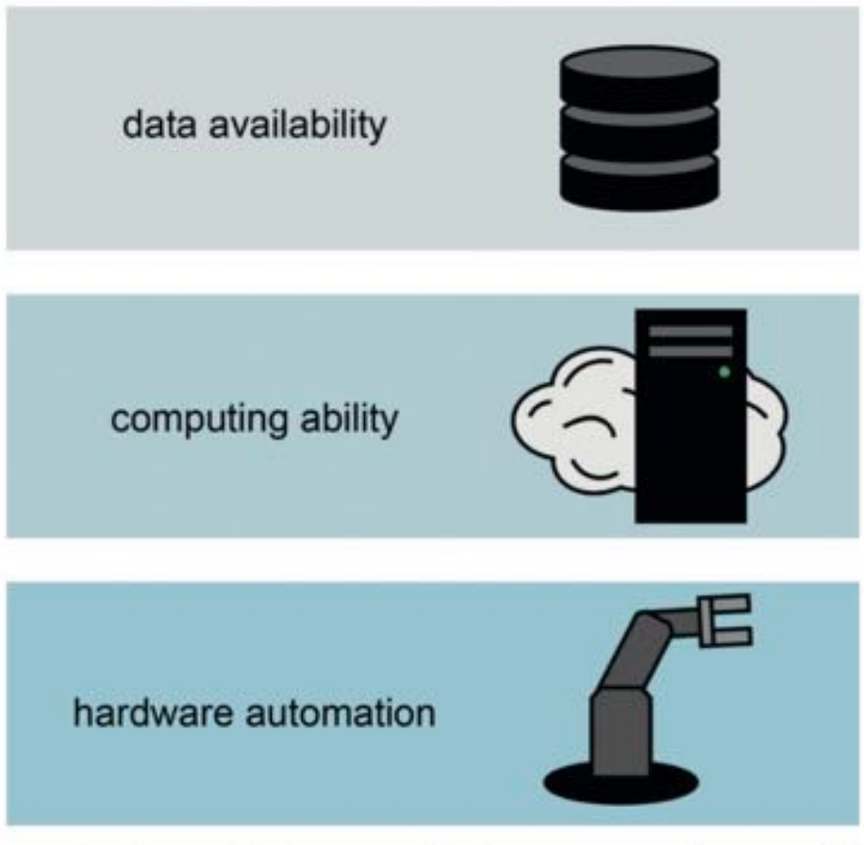


Figure 4. The factors that have enabled autonomous discovery fall into one of three main categories.

"Autonomous Discovery in the Chemical Sciences"

Automation gives advantages not only on efficiency but also reporoducibility.

NATURE REVIEWS | DRUG DISCOVERY
VOLUME 17 | FEBRUARY 2018 | 97

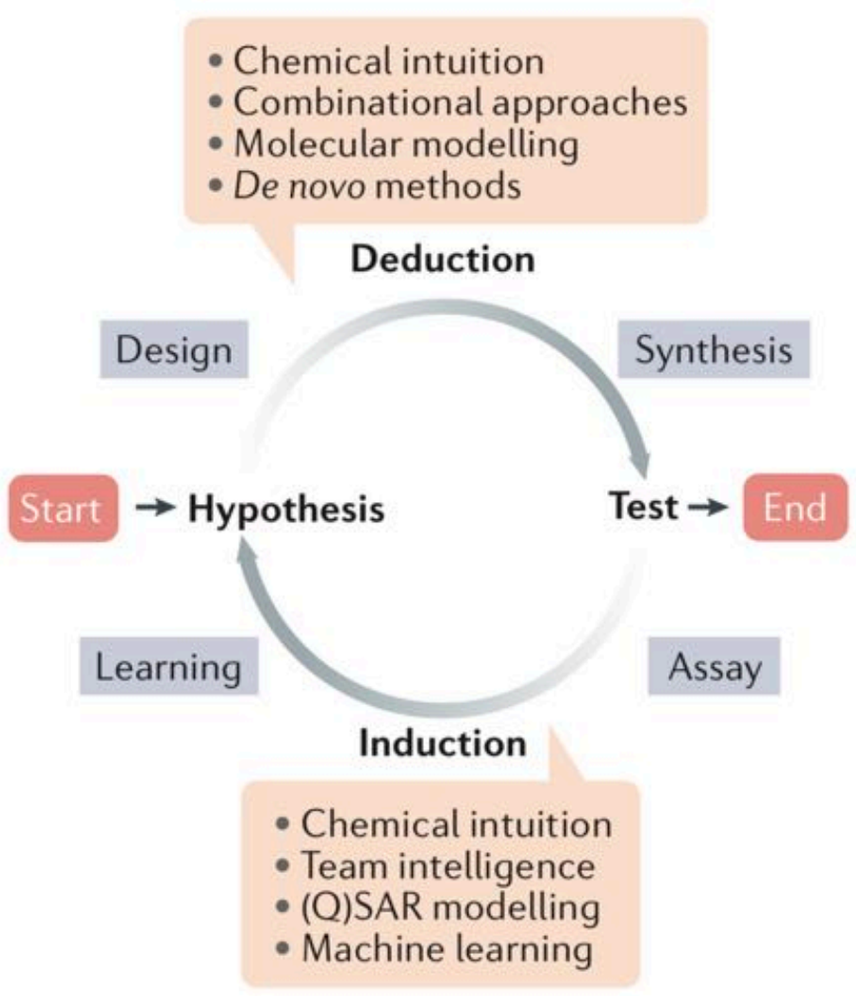
PERSPECTIVES

INNOVATION

Automating drug discovery

Gisbert Schneider

Figure 2 | **Automated drug discovery facilities. a** | Millions of compound samples are stored in compact high-capacity facilities and handled by robots. **b** | Robot systems perform both high-throughput and medium-throughput screening of up to ten thousand samples per day to determine the activity against the biological target of interest. Multiple arms and flexible workstations enable fully automated liquid dispensing, compound preparation and testing. These storage and screening systems have become cornerstones of contemporary drug discovery. **c** | A prototype of a novel miniaturized design–synthesize–test–analyse facility for rapid automated drug discovery at AstraZeneca is shown. Images **a** and **b** courtesy of Jan Kriegl, Boehringer–Ingelheim Pharma; image **c** courtesy of Michael Kossenjans, AstraZeneca.



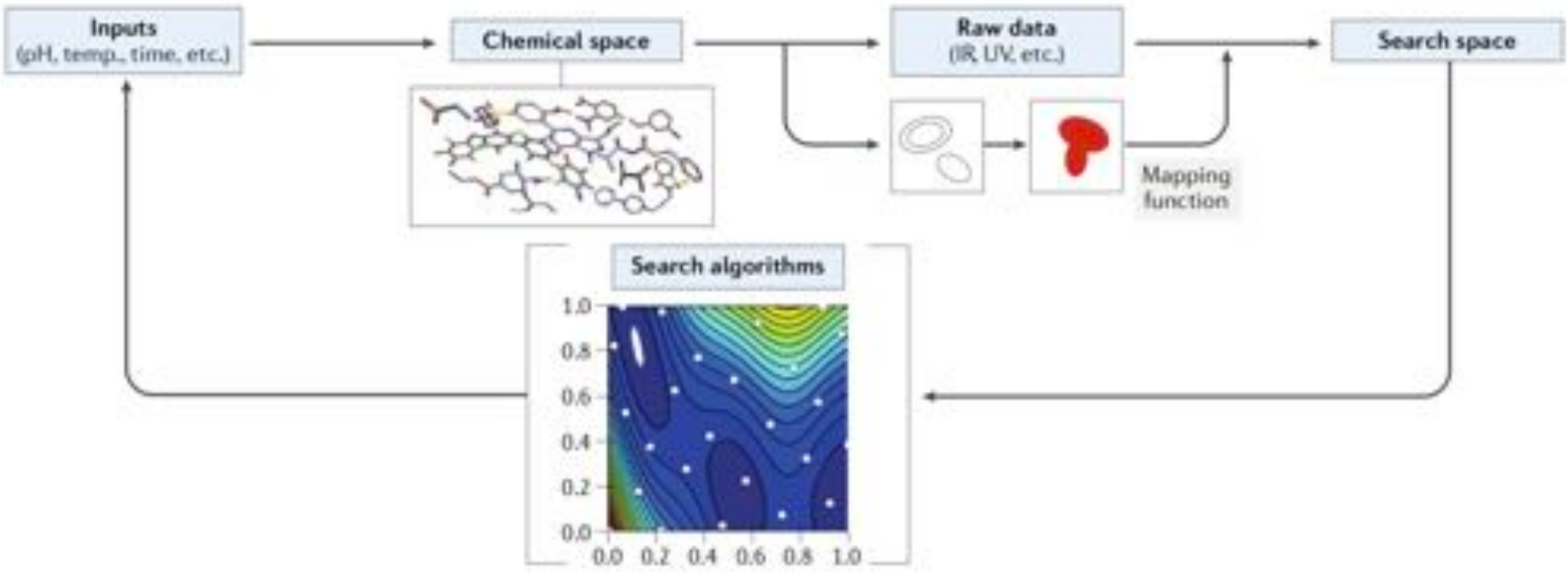
"Autonomous Discovery in the Chemical Sciences"

NATURE REVIEWS | CHEMISTRY

How to explore chemical space using algorithms and automation

PERSPECTIVES

Piotr S. Gromski, Alon B. Henson, Jarosław M. Granda and Leroy Cronin



"Chemputer" (If we set up a good system)

- Science 29 Nov (2018) 10.1126/science.aav2211
- Nature 559, pp 377–381 (2018) 10.1038/s41586-018-0307-8
- Science 359, pp. 314-319 (2018) 10.1126/science.aao3466
- Nat Comm 9, 3406 (2018) 10.1038/s41467-018-05828-8
- ACS Cent Sci 4, pp 793–804 (2018) 10.1021/acscentsci.8b00176

Prof. Leroy Cronin

Controlling an organic synthesis robot with machine learning to search for new reactivity

nature
International journal of science

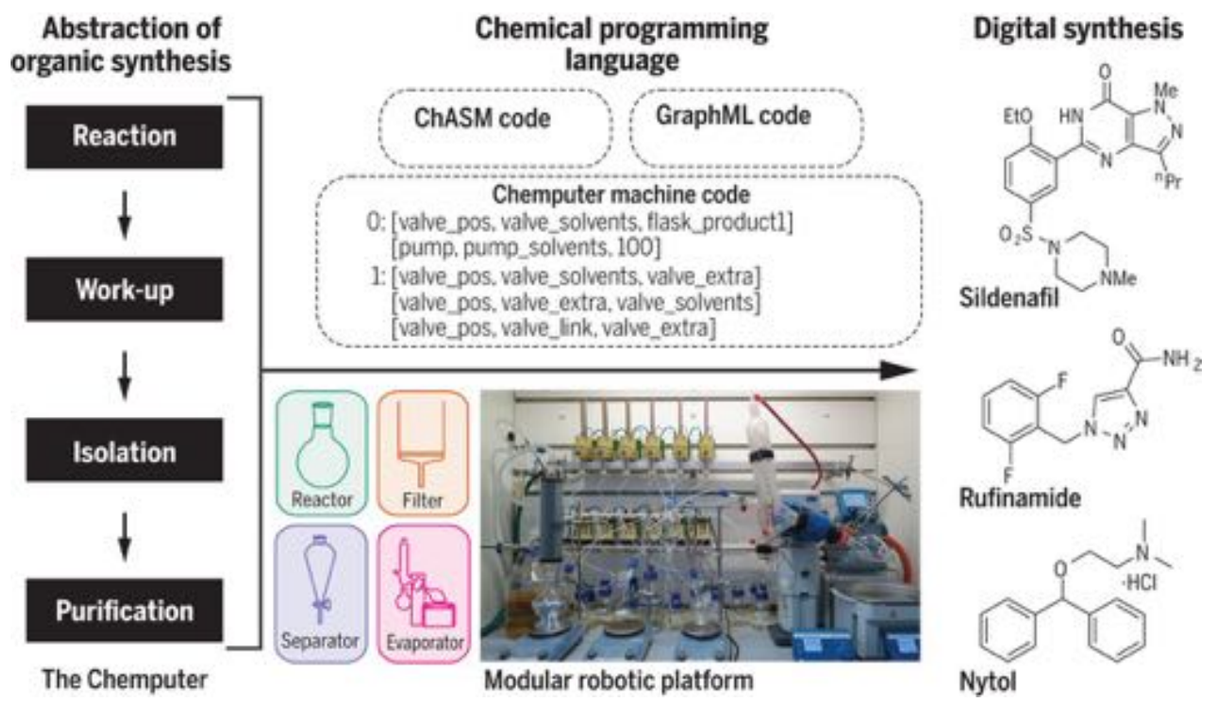
Jarosław M. Granda[1], Liva Donina[1], Vincenza Dragone[1], De-Liang Long[1] & Leroy Cronin[1]*

CHEMISTRY

Organic synthesis in a modular robotic system driven by a chemical programming language

Sebastian Steiner[1]*, Jakob Wolf[1], Stefan Glatzel[1], Anna Andreou[1], Jarosław M. Granda[1], Graham Keenan[1], Trevor Hinkley[1], Gerardo Aragon-Camarasa[1,2], Philip J. Kitson[1], Davide Angelone[1], Leroy Cronin[1]†

Science



"Mobile robotic chemists"?

"The authors estimate that a human scientist would have taken 1,000 times longer to produce similar results."

Article

Nature | Vol 583 | 9 July 2020 | 237

nature

A mobile robotic chemist

https://doi.org/10.1038/s41586-020-2442-2

Received: 1 November 2019

Accepted: 25 March 2020

Published online: 8 July 2020

Check for updates

Benjamin Burger[1], Phillip M. Maffettone[1], Vladimir V. Gusev[1], Catherine M. Aitchison[1], Yang Bai[1], Xiaoyan Wang[1], Xiaobo Li[1], Ben M. Alston[1], Buyi Li[1], Rob Clowes[1], Nicola Rankin[1], Brandon Harris[1], Reiner Sebastian Sprick[1] & Andrew I. Cooper[1]✉

"Mobile robotic chemists"?

"The authors estimate that a human scientist would have taken 1,000 times longer to produce similar results."

Article

Nature | Vol 583 | 9 July 2020 | 237

nature

A mobile robotic chemist

https://doi.org/10.1038/s41586-020-2442-2

Received: 1 November 2019

Accepted: 25 March 2020

Published online: 8 July 2020

Check for updates

Benjamin Burger[1], Phillip M. Maffettone[1], Vladimir V. Gusev[1], Catherine M. Aitchison[1], Yang Bai[1], Xiaoyan Wang[1], Xiaobo Li[1], Ben M. Alston[1], Buyi Li[1], Rob Clowes[1], Nicola Rankin[1], Brandon Harris[1], Reiner Sebastian Sprick[1] & Andrew I. Cooper[1]✉

"Virtual Chemist"?

NATURE CATALYSIS | VOL 3 | JULY 2020 | 574–584

ARTICLES

https://doi.org/10.1038/s41929-020-0468-3

nature catalysis

Check for updates

From desktop to benchtop with automated computational workflows for computer-aided design in asymmetric catalysis

Mihai Burai Patrascu[1], Joshua Pottel[1,2], Sharon Pinus[1], Michelle Bezanson[1], Per-Ola Norrby[3] and Nicolas Moitessier[1]✉

The organic chemist's toolbox is vast, with technologies to accelerate the synthesis of novel chemical matter. The field of asymmetric catalysis is one approach to accessing new areas of chemical space and computational power is today sufficient to assist in this exploration. Unfortunately, existing techniques generally require computational expertise and are therefore underutilized in synthetic chemistry. Here we present our platform Virtual Chemist, which allows bench chemists to predict outcomes of asymmetric chemical reactions ahead of testing in the laboratory, in just a few clicks. Modular workflows facilitate the simulation of various sets of experiments, including the four realistic scenarios discussed: one-by-one design, library screening, hit optimization and substrate-scope evaluation. Catalyst candidates are screened within hours and the enantioselectivity predictions provide substantial enrichments compared to random testing. The achieved accuracies within ~1kcal mol^{-1} provide opportunities for computational chemistry in the field of asymmetric catalyst design, allowing bench chemists to guide the design and discovery of asymmetric catalysts.

Automation goes through chemistry and chemical industry

https://www.chemistryworld.com/news/are-synthetic-chemists-out-of-a-job-as-ai-meets-automation/3010812.article

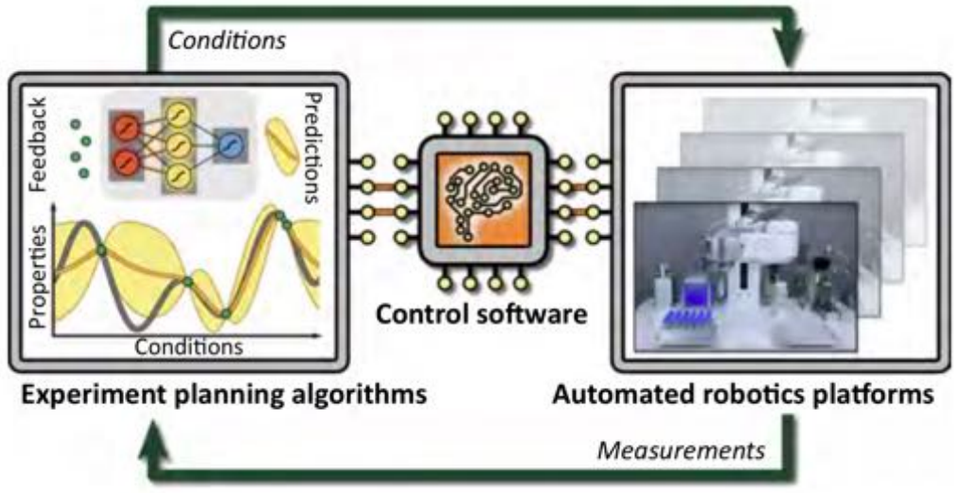
Automation goes through chemistry

Trends in Chemistry, June 2019, Vol. 1, No. 3 10.1016/j.trechm.2019.02.007

Opinion

Next-Generation Experimentation with Self-Driving Laboratories

Florian Häse,[1,2,3,4] Loïc M. Roch,[1,2,3,4] and Alán Aspuru-Guzik[1,2,3,4,5,*]



Computer-Aided Synthetic Planning

International Edition: DOI: 10.1002/anie.201506101
German Edition: DOI: 10.1002/ange.201506101

Computer-Assisted Synthetic Planning: The End of the Beginning

*Sara Szymkuć, Ewa P. Gajewska, Tomasz Klucznik, Karol Molga, Piotr Dittwald, Michał Startek, Michał Bajczyk, and Bartosz A. Grzybowski**

Angew. Chem. Int. Ed. **2016**, *55*, 5904 – 5937

Angewandte International Edition Chemie

Machine-Assisted Chemistry | Special Issue 150 Years of BASF

DOI: 10.1002/anie.201410744

Organic Synthesis: March of the Machines

Steven V. Ley, Daniel E. Fitzpatrick, Richard. J. Ingham, and Rebecca M. Myers*

Angew. Chem. Int. Ed. **2015**, *54*, 3449 – 3464

Automation goes through chemistry and chemical industry

NATURE REVIEWS | DRUG DISCOVERY
VOLUME 17 | FEBRUARY 2018 | 97

INNOVATION

Automating drug discovery

Gisbert Schneider

Figure 2 | **Automated drug discovery facilities. a** | Millions of compound samples are stored in compact high-capacity facilities and handled by robots. **b** | Robot systems perform both high-throughput and medium-throughput screening of up to ten thousand samples per day to determine the activity against the biological target of interest. Multiple arms and flexible workstations enable fully automated liquid dispensing, compound preparation and testing. These storage and screening systems have become cornerstones of contemporary drug discovery. **c** | A prototype of a novel miniaturized design–synthesize–test–analyse facility for rapid automated drug discovery at AstraZeneca is shown. Images **a** and **b** courtesy of Jan Kriegl, Boehringer–Ingelheim Pharma; image **c** courtesy of Michael Kossenjans, AstraZeneca.

Toyota teams with China's CATL and BYD to power electric ambitions

Automaker diversifies battery source and moves up electrification goal by 5 years

YUKIHIRO OMOTO, Nikkei staff writer
JUNE 07, 2019 02:00 JST ● UPDATED ON JUNE 07, 2019 14:39 JST

Little human intervention for highly reproducible large-scale production lines

Automation goes through chemistry and chemical industry

NATURE REVIEWS | DRUG DISCOVERY
VOLUME 17 | FEBRUARY 2018 | 97

INNOVATION

Automating drug discovery

Gisbert Schneider

Figure 2 | **Automated drug discovery facilities. a** | Millions of compound samples are stored in compact high-capacity facilities and handled by robots. **b** | Robot systems perform both high-throughput and medium-throughput screening of up to ten thousand samples per day to determine the activity against the biological target of interest. Multiple arms and flexible workstations enable fully automated liquid dispensing, compound preparation and testing. These storage and screening systems have become cornerstones of contemporary drug discovery. **c** | A prototype of a novel miniaturized design–synthesize–test–analyse facility for rapid automated drug discovery at AstraZeneca is shown. Images **a** and **b** courtesy of Jan Kriegl, Boehringer–Ingelheim Pharma; image **c** courtesy of Michael Kossenjans, AstraZeneca.

Toyota teams with China's CATL and BYD to power electric ambitions

Automaker diversifies battery source and moves up electrification goal by 5 years

YUKIHIRO OMOTO, Nikkei staff writer
JUNE 07, 2019 02:00 JST ● UPDATED ON JUNE 07, 2019 14:39 JST

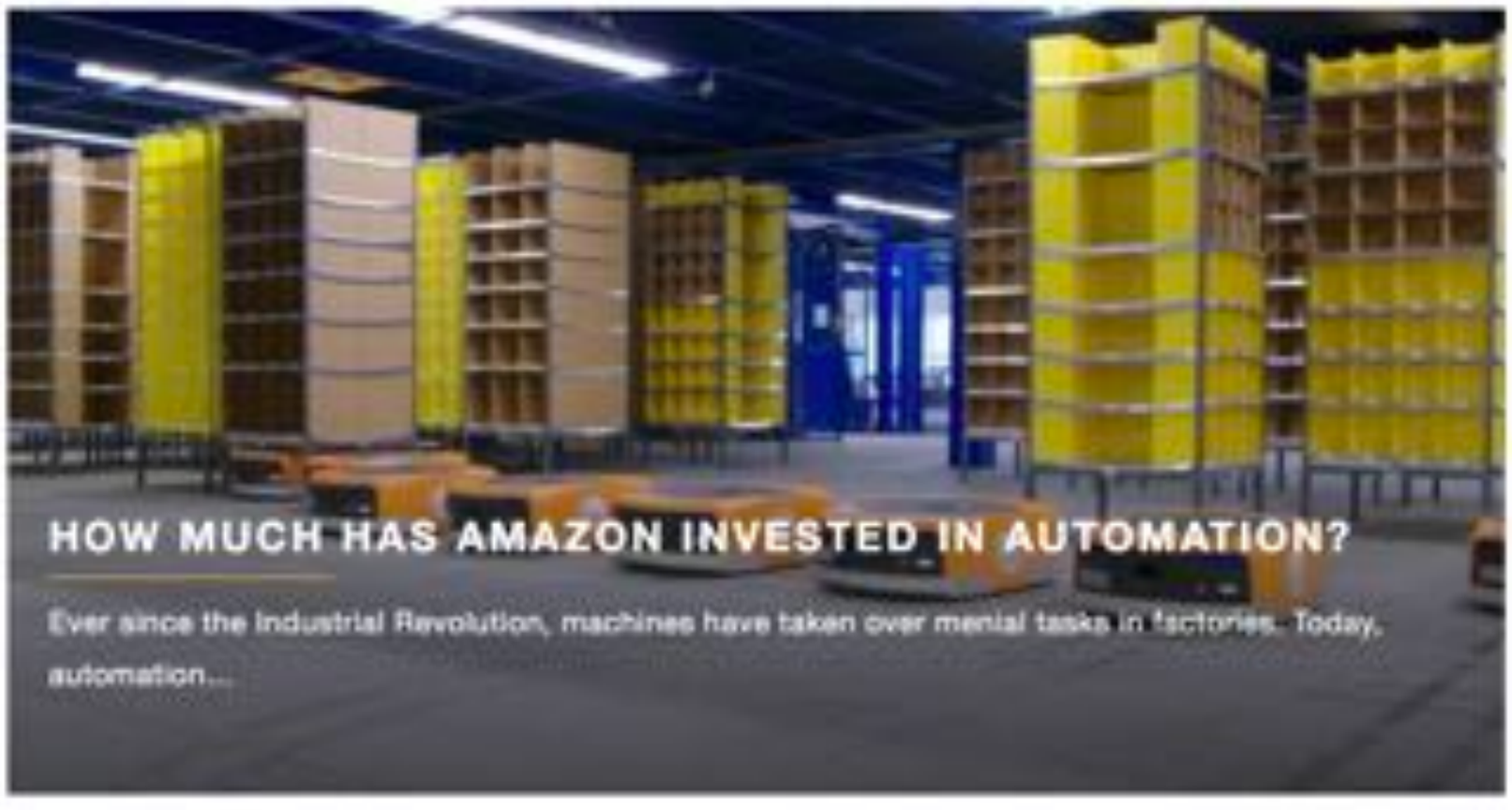
Little human intervention for highly reproducible large-scale production lines

.. and it is also strongly related to machine learning!

AMAZON
PICKING CHALLENGE

AMAZON ROBOT RESEARCH PROJECT

FUTURE OF ROBOTS | ROBOTICS | ABOUT US | MODERN WAR | FUTURE OF AI | CONTACT US



HOW MUCH HAS AMAZON INVESTED IN AUTOMATION?

Ever since the Industrial Revolution, machines have taken over menial tasks in factories. Today, automation...

Search

ABOUT US

AMAZON
PICKING CHALLENGE

AMAZON ROBOT RESEARCH PROJECT

As we all know, Amazon is infamous when it comes to automation and scaling large tasks at ease. The Amazon Robotics/Picking Challenge was a competition from early 2017 that allowed individuals to create their own small robotic systems to help with mass production and management.

After entering and coming second place, we

LATEST POSTS

- Today:
  data exploration (eScience)
  unify theory, experiment, and simulation
  - Data captured by instruments
    Or generated by simulator
  - Processed by software
  - Information/Knowledge stored in computer
  - Scientist analyzes database / files
    using data management and statistics

This would be more strongly complemented by Machine learning or AI, or data sciences.

ACS Catalysis

Review

pubs.acs.org/acscatalysis

Machine Learning for Catalysis Informatics: Recent Applications and Prospects

Takashi Toyao,[†,‡] Zen Maeno,[†] Satoru Takakusagi,[†] Takashi Kamachi,[‡,§] Ichigaku Takigawa,[*,∥,⊥] and Ken-ichi Shimizu[*,†,‡]

ACS Catalysis, 2019; 10: 2260-2297. https://doi.org/10.1021/acscatal.9b0418

Chapter 2 is a user's guide to use ML in practice!

Reviewer: 1

I don't usually recommend that papers should be accepted "as is", but in this case I don't see the need for changes. This review should be accepted and published in ACS Catalysis. ... I will certainly recommend it to my group and my students when it is published.

Reviewer: 2

The manuscript gives an excellent over the field of machine learning especially with regard to heterogeneous catalysis and I would highly recommend the article for the publication in ACS Catalysis.

Reviewer: 3

This is one of the best reviews for catalyst informatics that the Reviewer has read. In particular, the chapter 2 delivers a very good tutorial, which is concisely and professionally written.

Aug 26: 10:30~12:00 (90min)

1. What is "machine learning"?
2. Why does it matter to chemists?
3. Let's try it in your browser (with no setup!)

Aug 26: 13:00~14:30 (90min)

4. Five things all beginners should know
    - "The quality of your inputs decide the quality of your output"
    - Training / validation / test data
    - Tuning hyperparameters
    - Identification and design of input variables (or "descriptors")
    - "Correlation does not imply causation"
5. Standard pipeline and deep learning
6. Current efforts and future directions