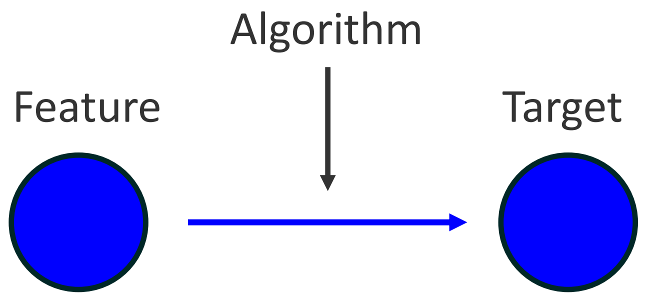
THE UNIVERSITY of TENNESSEE KNOXVILLE

Lecture 09: SVM, kNN, XGBoost, and Materials Discovery

Instructor: Sergei V. Kalinin

Tasks, workflows, and workflow planning

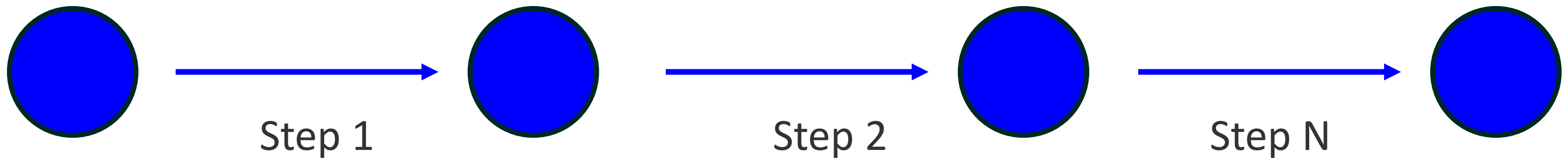
Task:



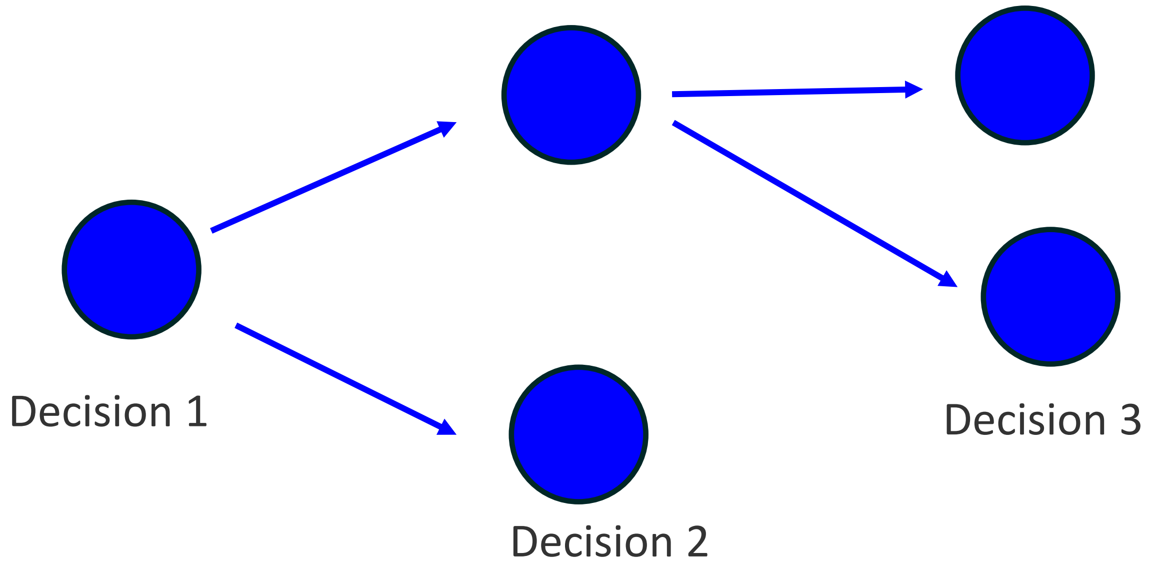
Task can be classification, regression, clustering, optimization.
Algorithm can be kNN, perceptron, DCNN, etc.

We can use complex algorithms for “simple” tasks and simple algorithms for complex tasks.

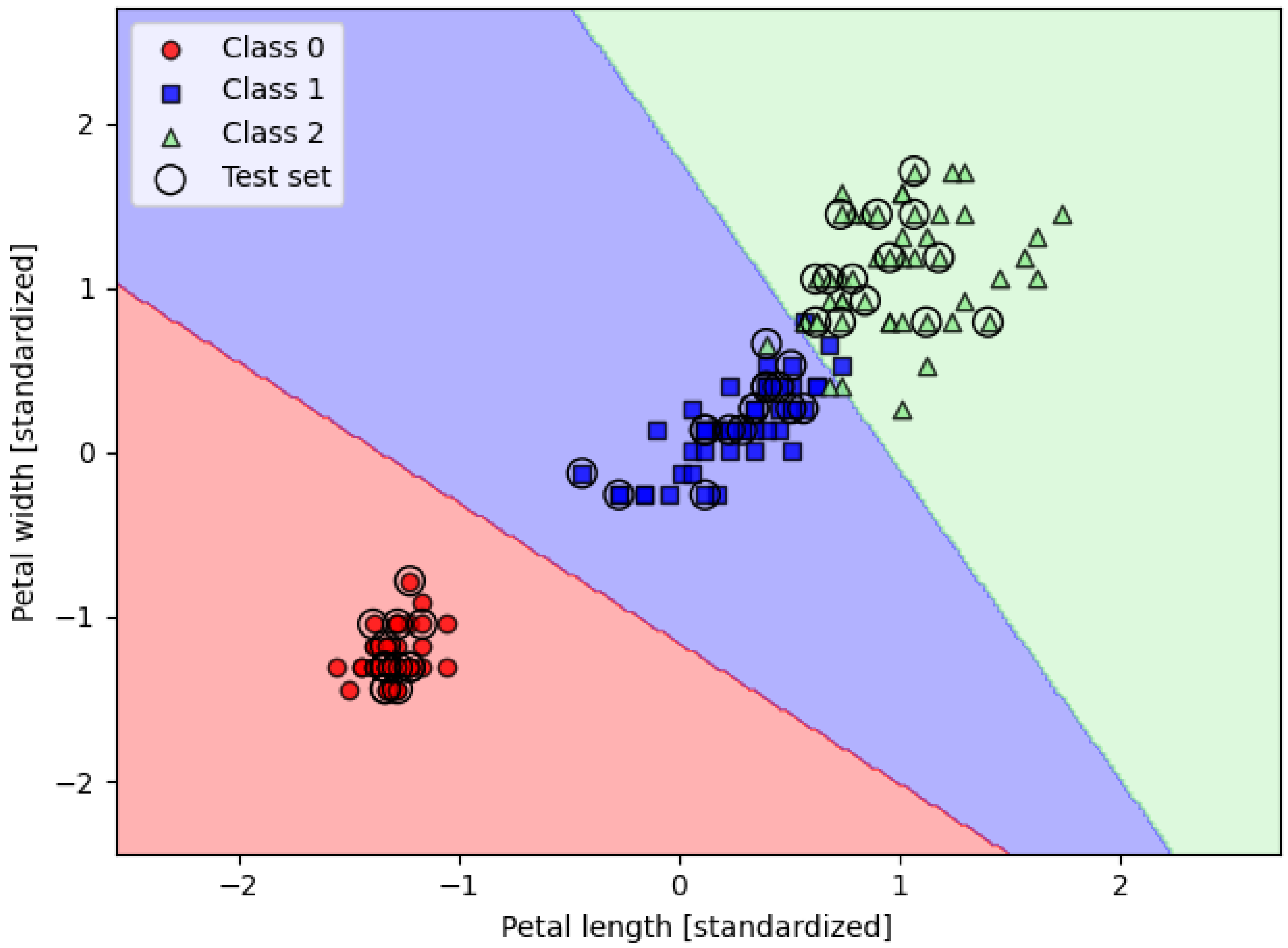
Workflow:



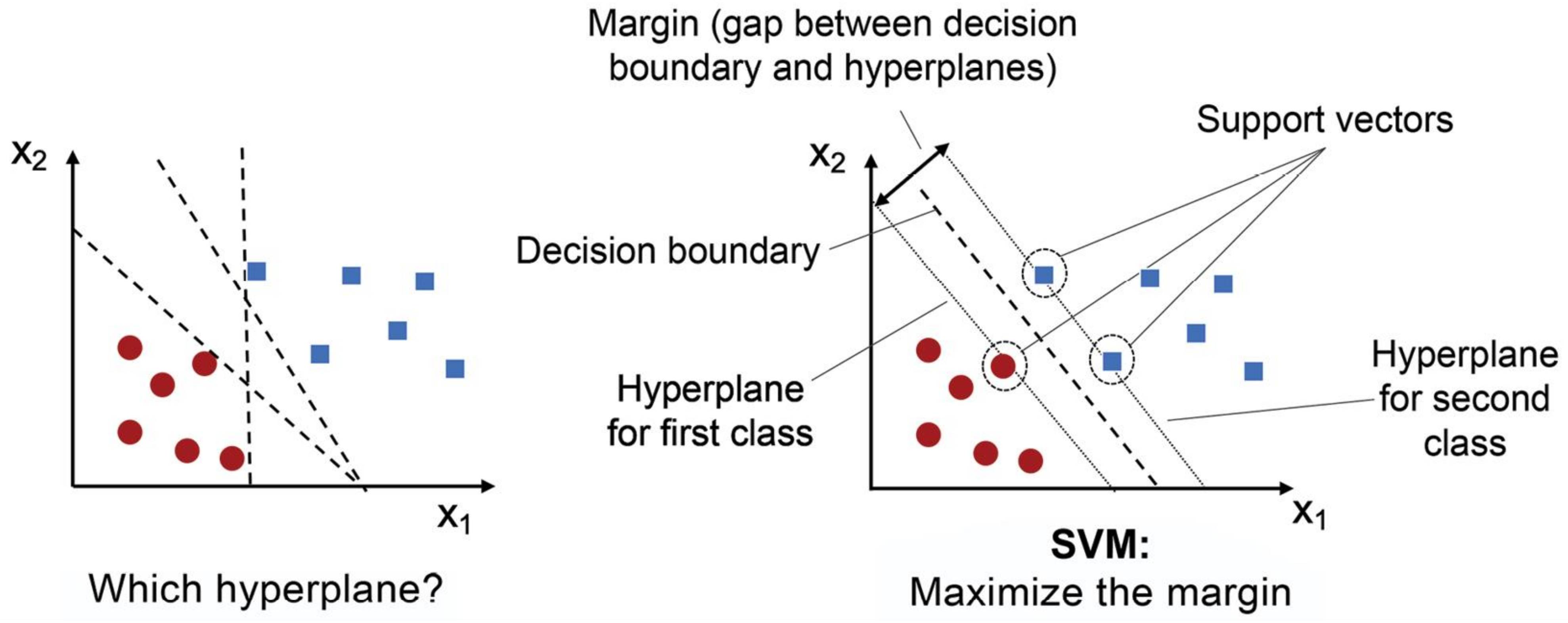
Workflow Design:



Classification problem



Support Vector Machines

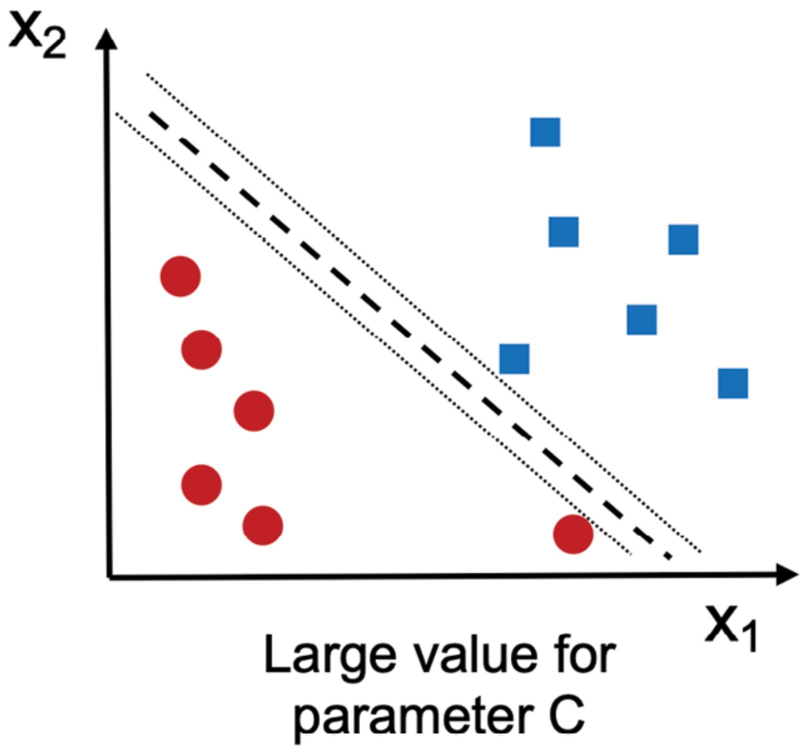

Margin (gap between decision boundary and hyperplanes)
Support vectors
x_2
x_1
Decision boundary
Hyperplane for first class
Hyperplane for second class
Which hyperplane?
SVM:
Maximize the margin

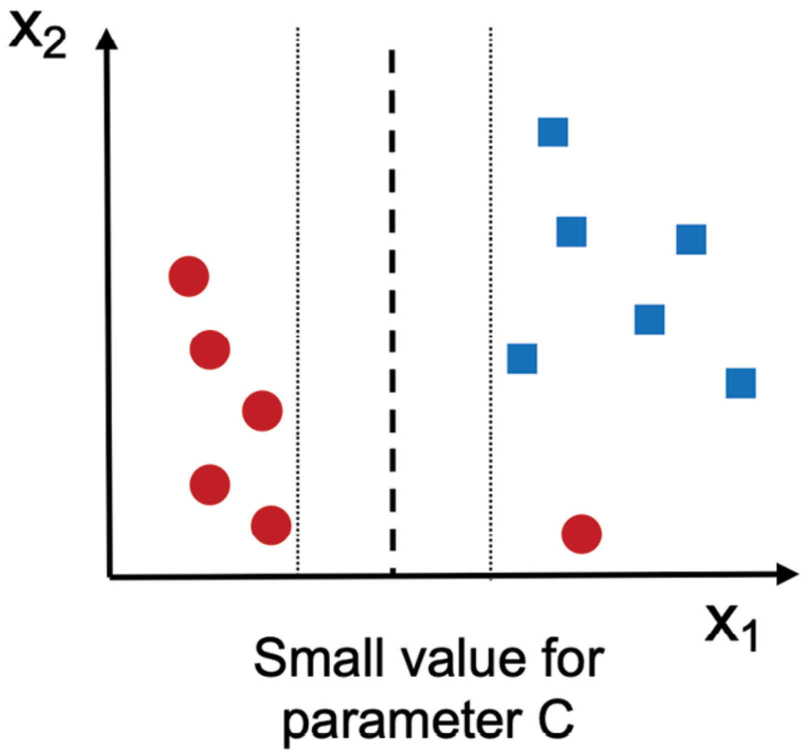
Perceptron: minimize misclassification errors.
SVM: maximize the margin.

The margin is the distance between the separating hyperplane (decision boundary) and the training examples that are closest to this hyperplane, which are called **support vectors**.

From S. Raschka, Machine Learning with PyTorch and Scikit-Learn

Regularization in SVMs

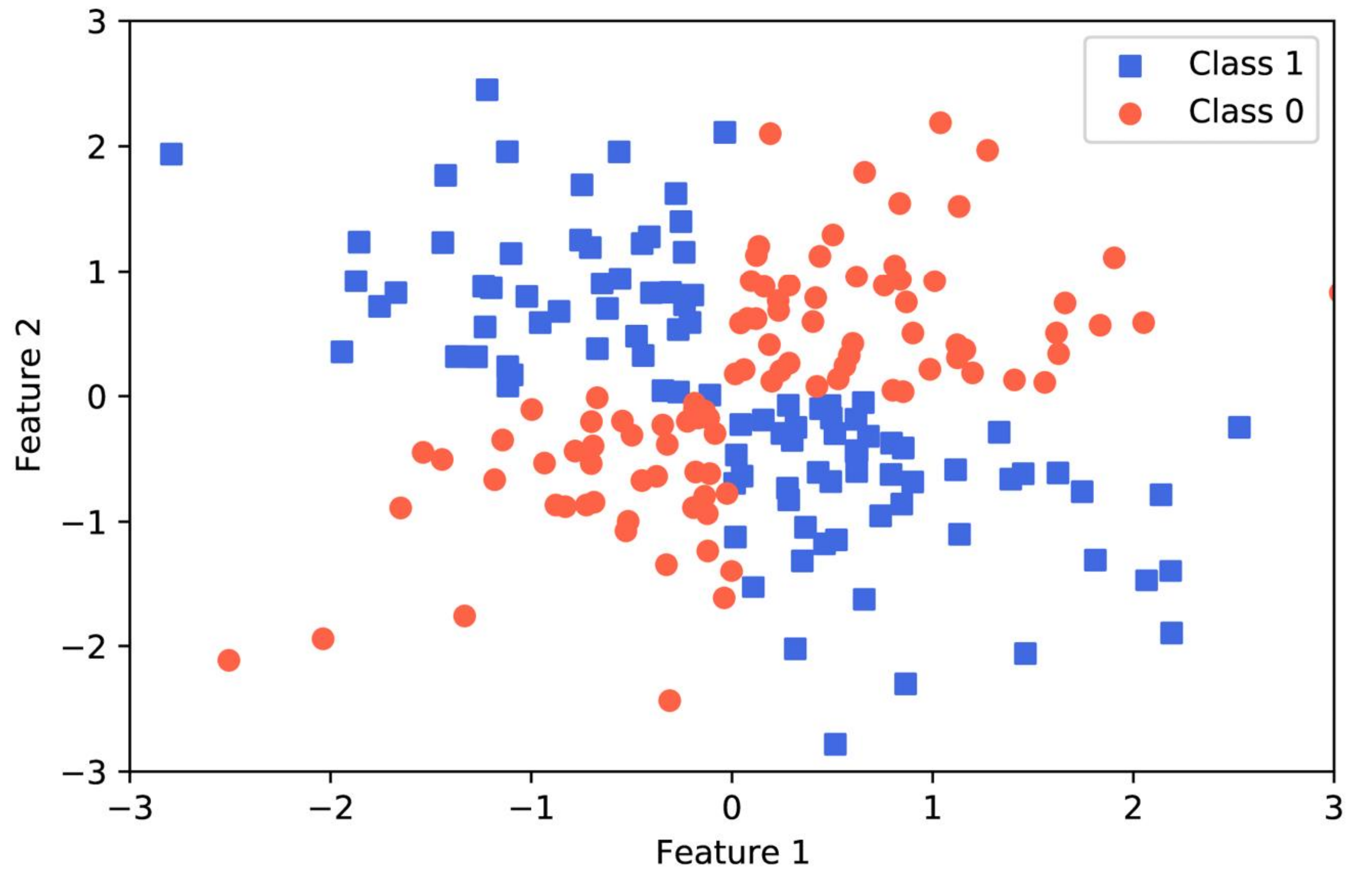




Large values of *C* correspond to large error penalties, whereas we are less strict about misclassification errors if we choose smaller values for *C*.

From S. Raschka, Machine Learning with PyTorch and Scikit-Learn

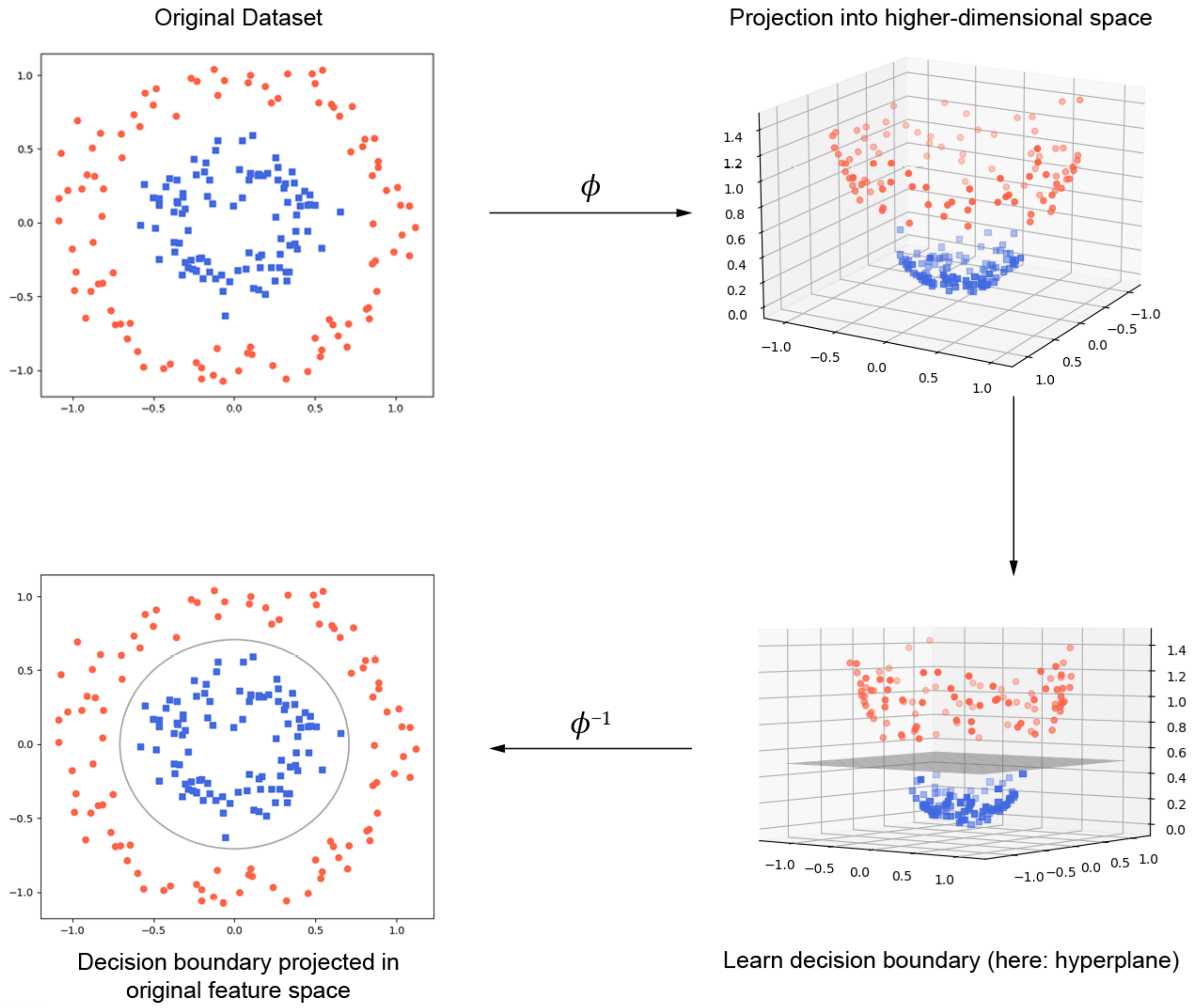
Kernel SVM: Motivation



Non-linearly separable problem

From S. Raschka, Machine Learning with PyTorch and Scikit-Learn

Kernel SVM



The idea behind **kernel methods** for dealing with linearly inseparable data is to create nonlinear combinations of the original features to project them onto a higher-dimensional space via a mapping function, ϕ , where the data becomes linearly separable.

$$\phi(x_1, x_2) = (z_1, z_2, z_3) = (x_1, x_2, x_1^2 + x_2^2)$$

From S. Raschka, Machine Learning with PyTorch and Scikit-Learn

Kernel SVM

- To train an SVM, in practice, we need to replace the dot product $\boldsymbol{x}^{(i)T}\boldsymbol{x}^{(j)}$ by $\phi\left(\boldsymbol{x}^{(i)}\right)^T \phi\left(\boldsymbol{x}^{(j)}\right)$.

- To save the expensive step of calculating this dot product between two points explicitly, we define a **kernel function** $\kappa\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right) = \phi\left(\boldsymbol{x}^{(i)}\right)^T \phi\left(\boldsymbol{x}^{(j)}\right)$
-
- One of the most widely used kernels is the **radial basis function** (**RBF**) kernel, or the **Gaussian kernel**:

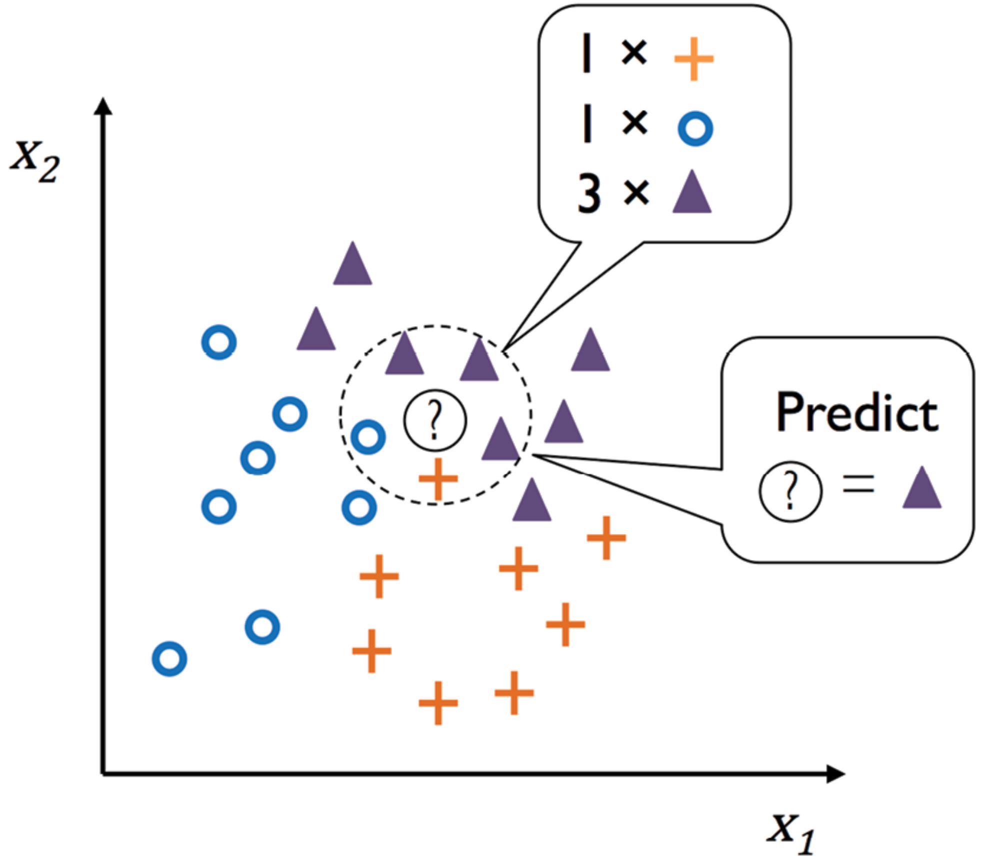
$$\kappa\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right) = \exp\left(-\gamma\left\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\right\|^2\right)$$

γ is a free parameter to be optimized.

- Roughly speaking, the term “kernel” can be interpreted as a **similarity function** between a pair of examples.

From S. Raschka, Machine Learning with PyTorch and Scikit-Learn

k Nearest Neighbours (kNN) Classifier



1. Choose the number of *k* and a distance metric

2. Find the *k*-nearest neighbors of the data record that we want to classify

3. Assign the class label by majority vote

KNN is an example of a **lazy learner**. It is called “lazy” not because of its apparent simplicity, but because it doesn’t learn a discriminative function from the training data but memorizes the training dataset instead.

From S. Raschka, Machine Learning with PyTorch and Scikit-Learn

Parametric vs. Non-parametric methods

Parametric models: we estimate parameters from the training dataset to learn a function that can classify new data points without requiring the original training dataset anymore. Typical examples of parametric models are the perceptron, logistic regression, and the linear SVM.

Non-parametric models can't be characterized by a fixed set of parameters, and the number of parameters changes with the amount of training data. Two examples of non-parametric models are the decision tree classifier/random forest and the kernel (but not linear) SVM.

KNN belongs to a subcategory of non-parametric models described as **instance-based learning**. Models based on instance-based learning are characterized by memorizing the training dataset, and lazy learning is a special case of instance-based learning that is associated with no (zero) cost during the learning process.

From S. Raschka, Machine Learning with PyTorch and Scikit-Learn

Pros and Cons of Memory Based Approaches

- The main advantage of memory-based approach is that the classifier immediately adapts as we collect new training data

- The downside is that the computational complexity for classifying new examples grows linearly with the number of examples in the training dataset in the worst-case scenario—unless the dataset has very few dimensions (features) and the algorithm has been implemented using efficient data structures for querying the training data more effectively.

- Such data structures include k-d tree (https://en.wikipedia.org/wiki/K-d_tree) and ball tree (https://en.wikipedia.org/wiki/Ball_tree), which are both supported in scikit-learn. Furthermore, next to computational costs for querying data, large datasets can also be problematic in terms of limited storage capacities.

- However, in many cases when we are working with relatively small to medium-sized datasets, memory-based methods can provide good predictive and computational performance and are thus a good choice for approaching many real-world problems.

From S. Raschka, Machine Learning with PyTorch and Scikit-Learn

Distances and Neighbours

- We need to choose a distance metric that is appropriate for the features in the dataset.
- Usual choice: Minkowski distance, a generalization of the Euclidean (p = 2) and Manhattan (p = 1) distance:

$$d\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right) = \sqrt[p]{\sum_k \left|x_k^{(i)} - x_k^{(j)}\right|^p}$$

- Many other distance metrics are available in scikit-learn and can be provided to the metric parameter. They are listed at https://scikit-learn.org/stable/modules/generated/sklearn.metrics.DistanceMetric.html.
- KNN is very susceptible to overfitting due to the **curse of dimensionality**, where feature space becomes increasingly sparse for an increasing number of dimensions of a fixed-size training dataset.
- For regression and SVM, we use regularization to avoid this problem.
- We cannot use regularization for decision trees and KNN. Instead, we can use feature selection and dimensionality reduction techniques

From S. Raschka, Machine Learning with PyTorch and Scikit-Learn

How materials are discovered?

Corning Ware glass was accidentally discovered via a furnace mishap

“The temperature gauge was stuck on 900 degrees,
and I thought I had ruined the furnace …

I grabbed some tongs to get it out as fast as I could,
but the glass slipped out of the tongs and fell to the floor.

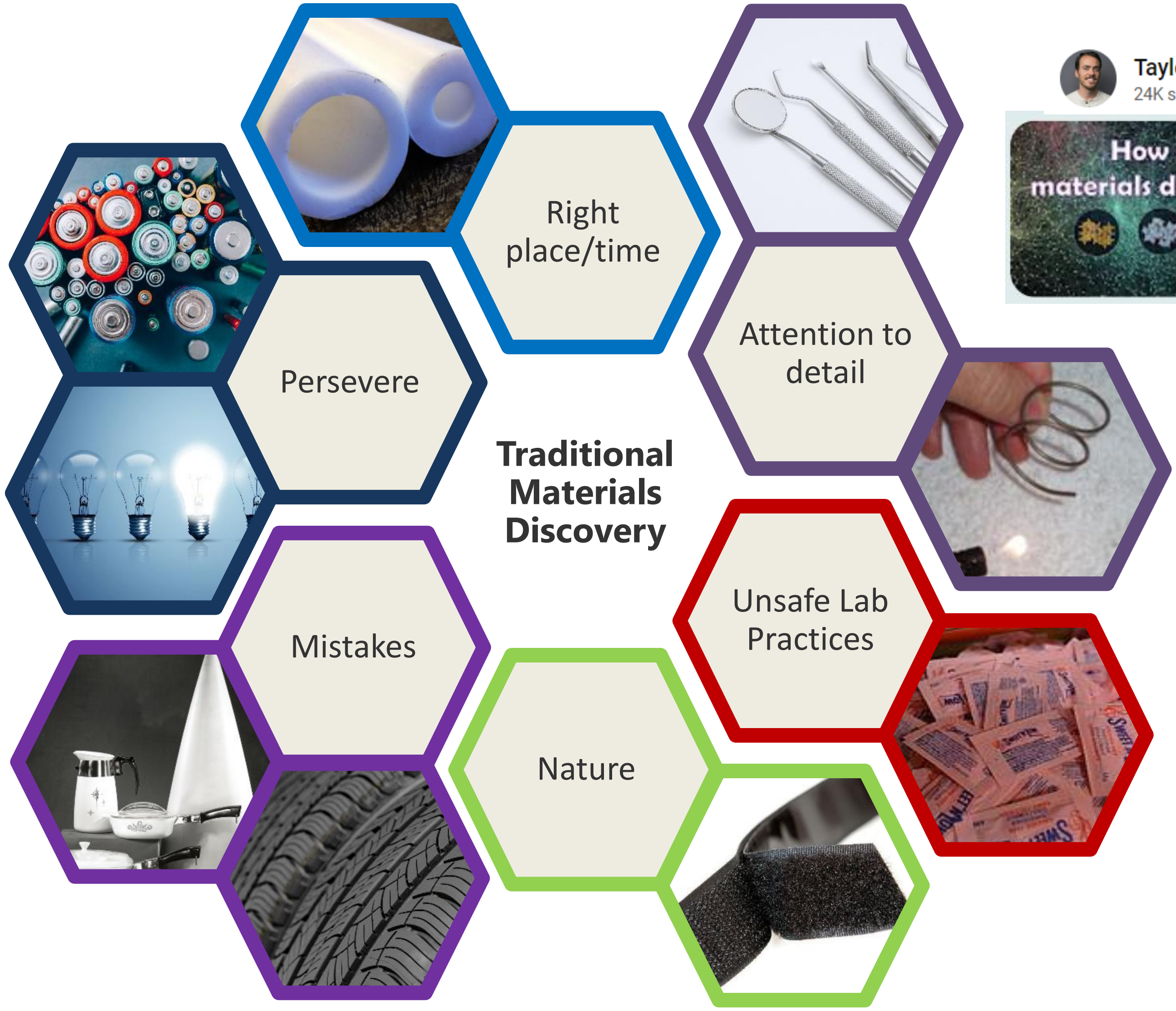
The thing bounced and didn’t break.”

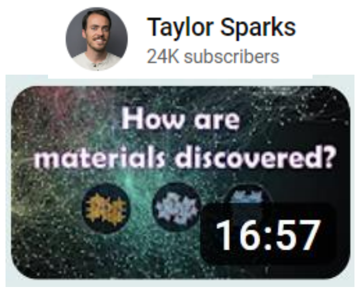
Donald Stookey (1915-2014)

https://cen.acs.org/articles/92/web/2014/12/Donald-StookeyGuy-Gave-Us-Corning.html
https://www.nytimes.com/2014/11/07/business/s-donald-stookey-inventor-of-corningware-dies-at-99.html

Slide by Sterling Baird

How materials are discovered?

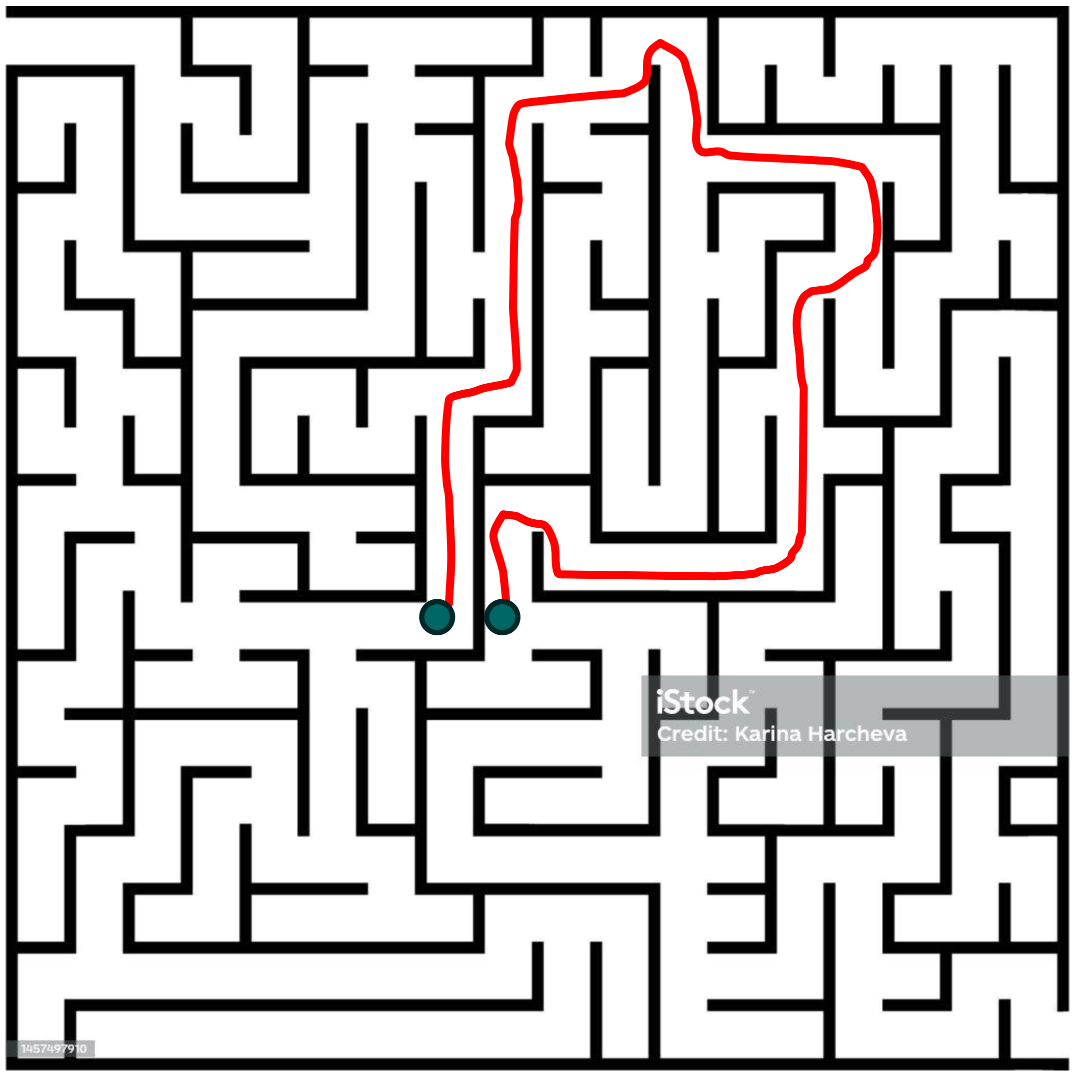




Velcro photo, shape memory alloy licensed under CC BY| Sweet N Low, Teflon licensed under CC BY-SA
“How are materials discovered?” by Taylor Sparks: https://www.youtube.com/watch?v=RRNcqJSJ6vc

Slide by Sterling Baird

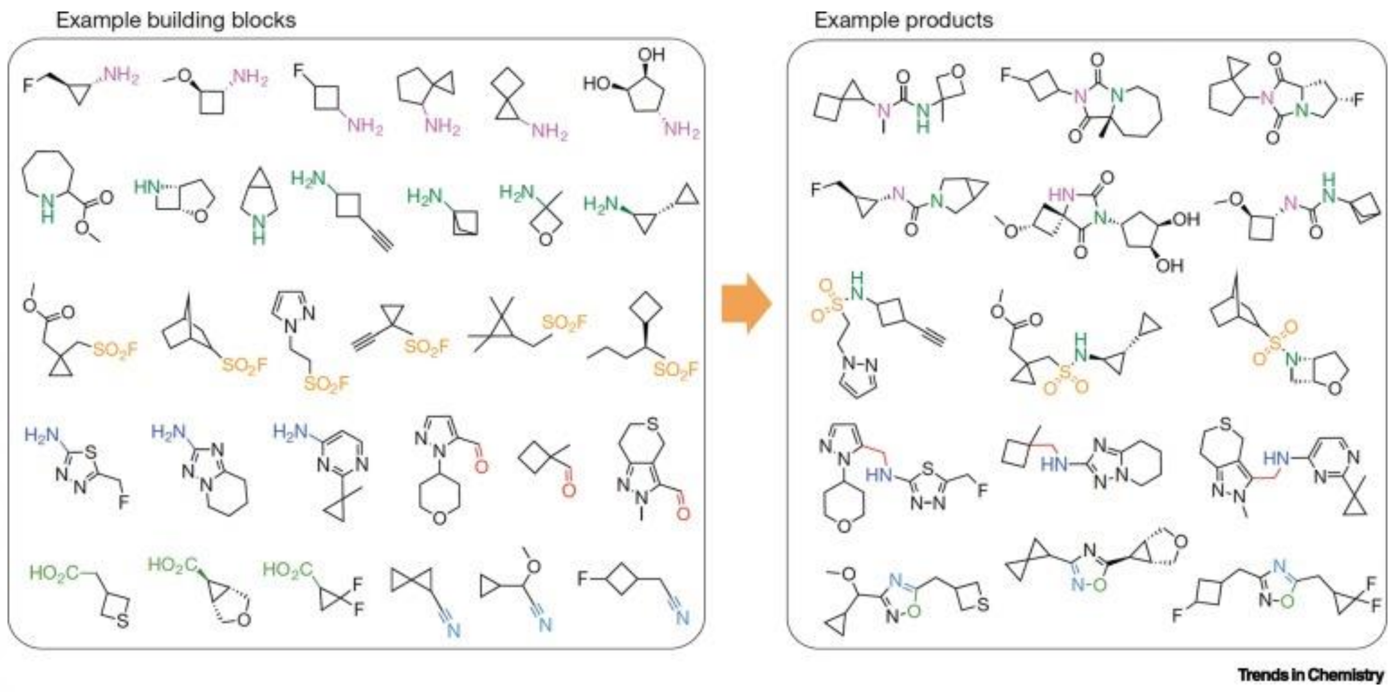
How materials are discovered?

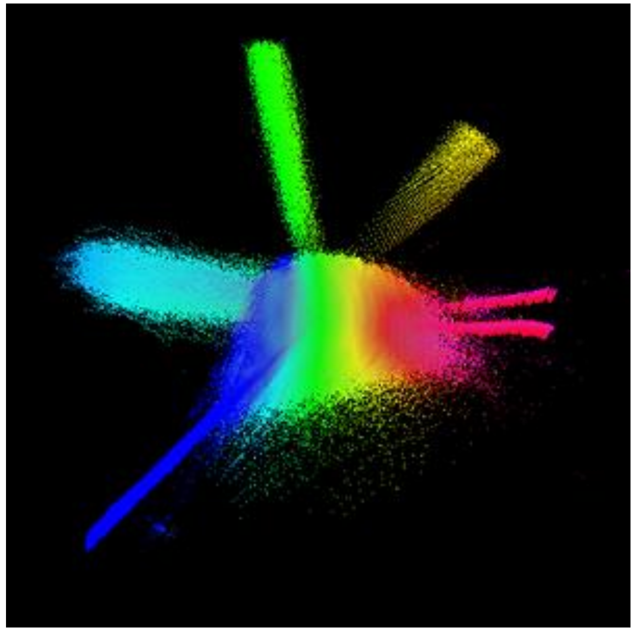


https://en.wikipedia.org/wiki/LK-99

1986	–	$YBa_2Cu_3O_7$. Gave rise to multiple families of Cu and Hg superconductors
2001	–	MgB_2. Point compound
2006	-	Layered iron pnictides. Gave rise to multiple families of superconductors

How many molecules are there?



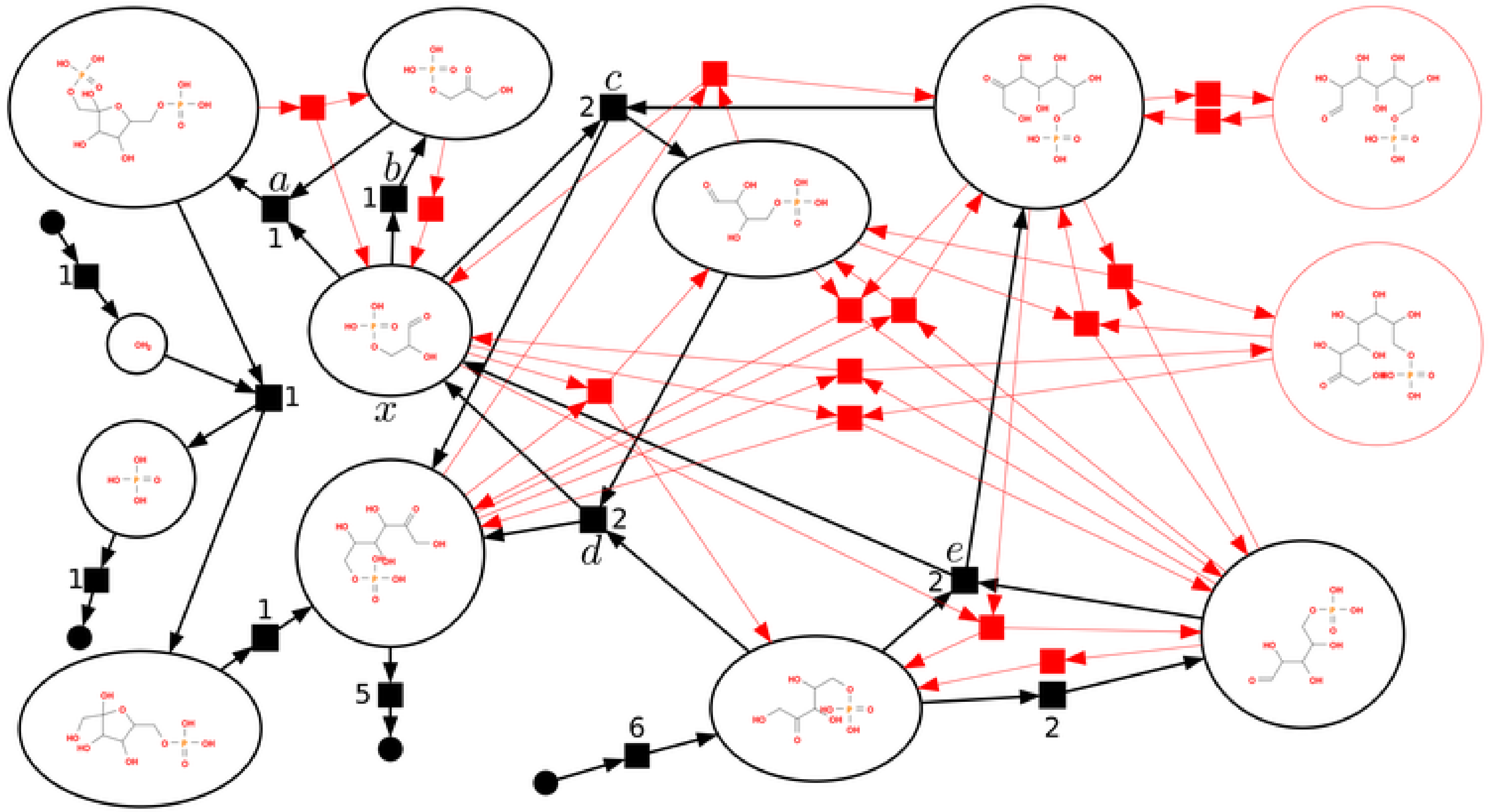


A chemical space often referred to in cheminformatics is that of potential biologically active molecules. Its size is estimated to be in the order of 10^{60} molecules. The estimate restricts the chemical elements used to be C, H, O, N and S. It further makes the assumption of a maximum of 30 atoms to stay below 500 Daltons, allows for branching and a maximum of 4 rings and arrives at an estimate of 10^{63}.

https://www.cell.com/trends/chemistry/fulltext/S2589-5974%2820%2930288-4

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

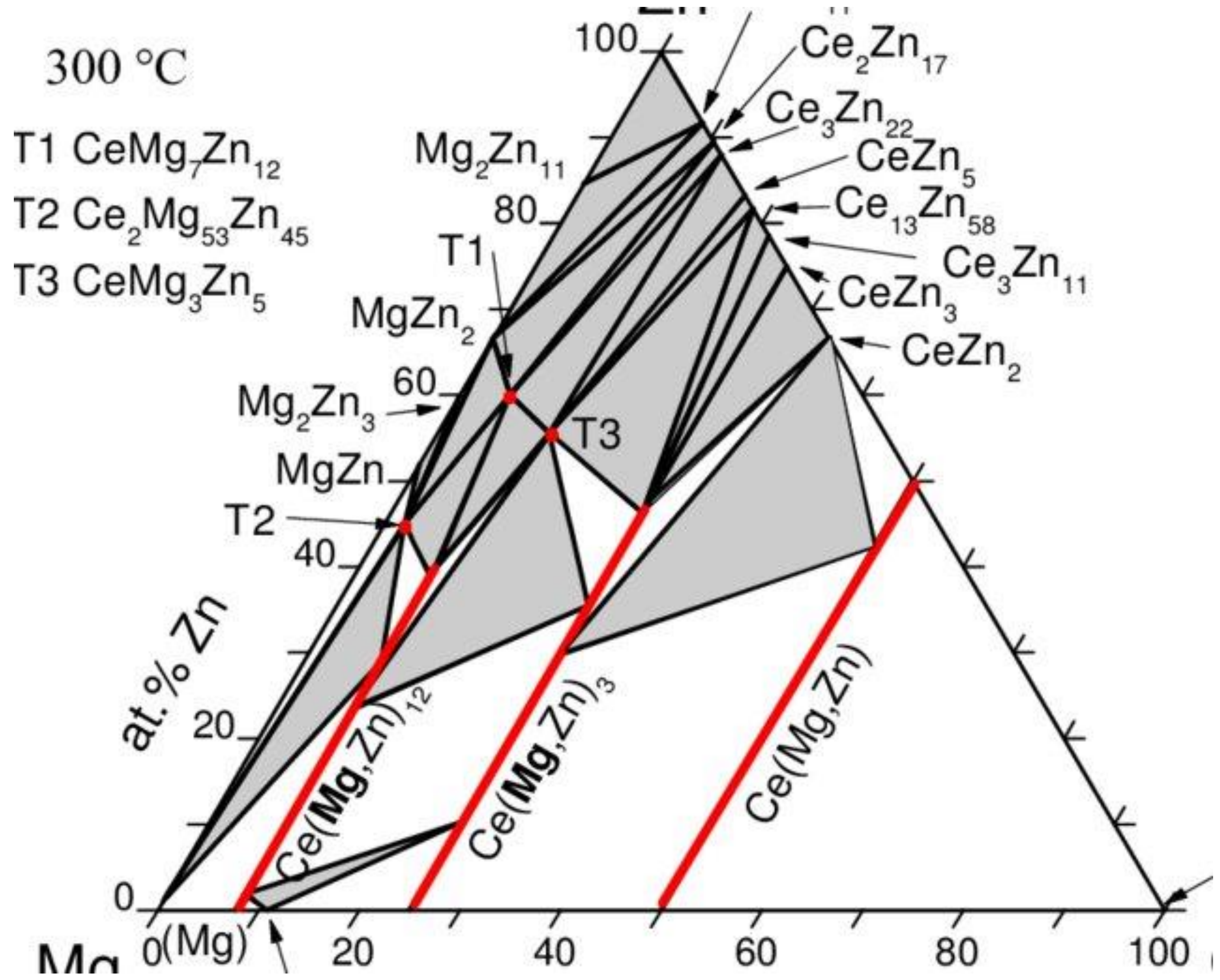
Chemical reactions networks:



- Molecular property predictions: are they **likely** to be useful?
- Synthesizability scores: what would it **probably** take to make them
- Reaction network mining and retrosynthesis: can we identify **possible** synthetic pathways?
- Optimization of specific reaction conditions and pathways: myopic and non-myopic

https://www.mis.mpg.de/stadler/research/chemical-reaction-networks.html

Why dimensionality is a problem?



Let’s think about it as a search problem:

- **Alloying:** need maintain composition ~1%
- **Doping:** need maintain composition ~ 10^{-6}
- Grid search is out for D > 3 (experiment)

Why dimensionality is a problem?

- Suppose that we have data for 1000 students' performance (discretized scores of 0; 25; 50; 75; 100)% in 2 courses c1 and c2. Then in total there are 5 x 5 = 25 different grade combinations.

- If the 1000 students are randomly distributed among each grade combination, then on average there are 40 students with each possible grade combination, which is a good enough sample to draw conclusions such as if, for a student, grade(c1) 50 and grade(c2) 75, then that student is likely to be a Math major.

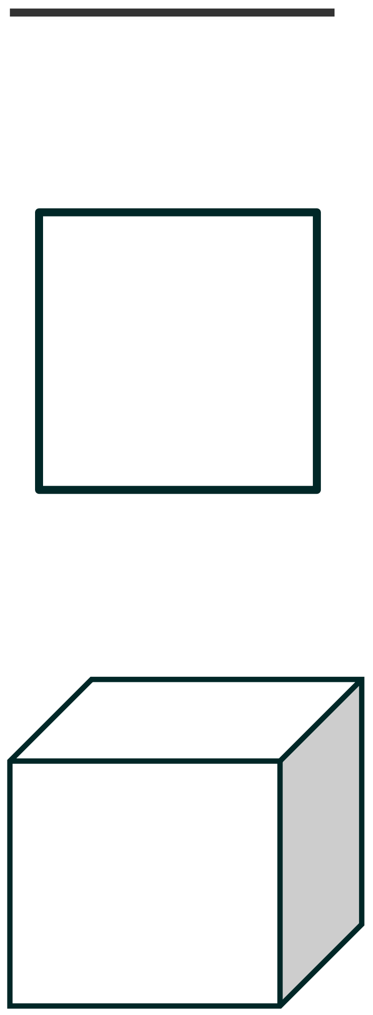
- Now suppose there are 4 courses, then the number of possible grades combination is 54 = 625, and an average number of students per combination is 1:6. For 10 courses, this number reduces to 0:0001024. This means that almost all possible combinations are never observed.

From "Curse of Dimensionality", Lecture Notes for Big Data Analytics, Faizad Ullah, February 2019

Why dimensionality is a problem?

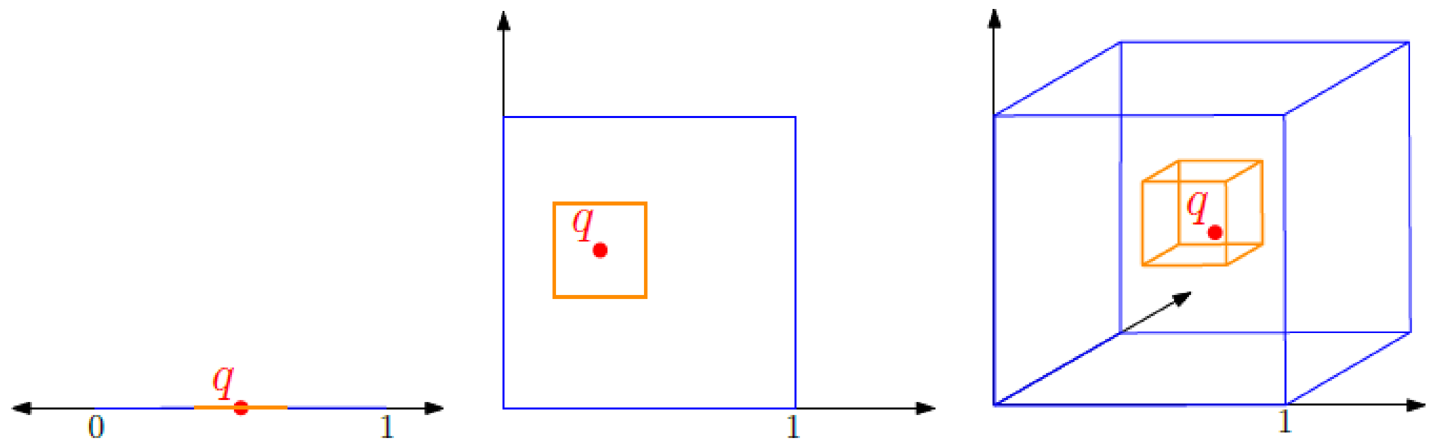
- Suppose *n* points in X are chosen uniformly at random from $[0; 1]^m$ (m-cube). For the query point q grow a hypercube around *q* to contain *f* fraction of points (k = f n) in X. This cube (the search space for q) grows very large (covering almost the whole input space) in large dimension.

- The expected length of the edge of the search cube $E_m(f) = f^{1/m}$, i.e. in 10d to get 10% points around q need cube with edge length 0.8 (which is 80% of the whole cube, the input space). Similarly, to get only 1% points one needs to extend the search cube by 0:63 units along each dimension



From "Curse of Dimensionality", Lecture Notes for Big Data Analytics, Faizad Ullah, February 2019

Why dimensionality is a problem?

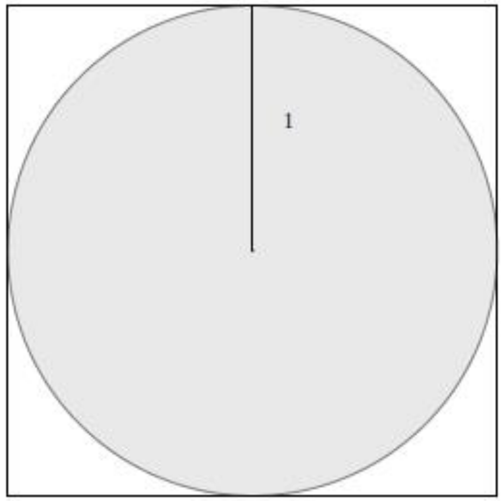


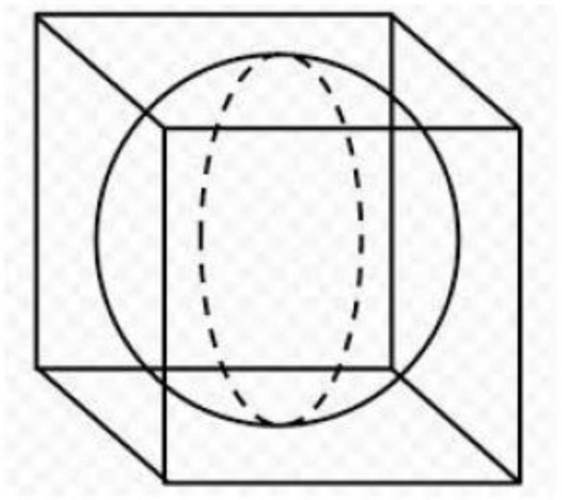
Suppose we have 5000 points:

- In 1d we have to explore 0.001 on average to capture 5 NN
- In 2d, on average we must explore 0:031 units along both dimensions to get 5 nearest neighbors points (about 3% of the whole cube).
- In 3d, on average we must go 10% of the total (unit) length in each of the 3 dimensions
- In 4d, we must explore 17:7% of unit length
- In 10d, we must go 50.1% of unit length along each dimension

From "Curse of Dimensionality", Lecture Notes for Big Data Analytics, Faizad Ullah, February 2019

Why dimensionality is a problem?

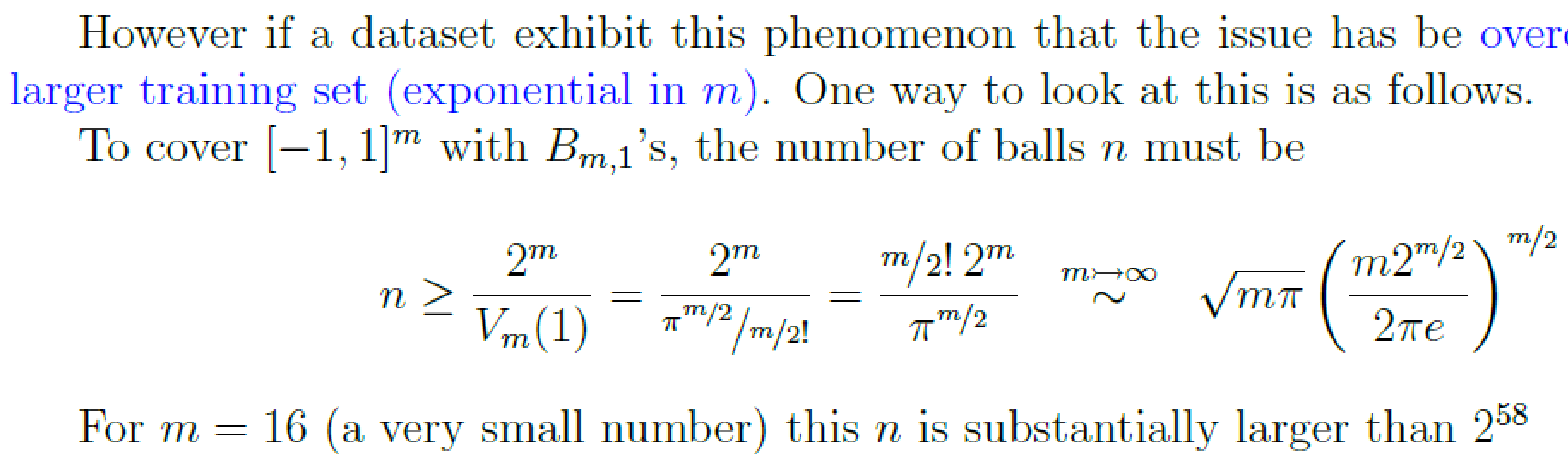




dim m	volume of m-ball	volume of m-cube	ratio
2	π	2^2	~ 0.785
3	${}^4\!/_3\pi$	2^3	~ 0.523
4	${}^{\pi^2}\!/_2$	2^4	~ 0.308
6	${}^{\pi^3}\!/_6$	2^6	~ 0.080
m	$\frac{\pi^{m/2}}{m/2!}$	2^m	$\to 0$

From "Curse of Dimensionality", Lecture Notes for Big Data Analytics, Faizad Ullah, February 2019

Why dimensionality is a problem?



However if a dataset exhibit this phenomenon that the issue has be overcome by getting a larger training set (exponential in m). One way to look at this is as follows.
To cover $[-1,1]^m$ with $B_{m,1}$'s, the number of balls n must be

$$n \geq \frac{2^m}{V_m(1)} = \frac{2^m}{\pi^{m/2}/m/2!} = \frac{m/2!\,2^m}{\pi^{m/2}} \overset{m\to\infty}{\sim} \sqrt{m\pi}\left(\frac{m2^{m/2}}{2\pi e}\right)^{m/2}$$

For $m = 16$ (a very small number) this n is substantially larger than 2^{58}

- In higher dimensions all the volume is in `corners'
- Points in high dimensional spaces are isolated (empty surrounding)
- The probability that a randomly generated point is within r radius of q approaches 0 as dimensionality increases
- The probability of a close nearest neighbor in a data set is very small

From “Curse of Dimensionality”, Lecture Notes for Big Data Analytics, Faizad Ullah, February 2019

THE UNIVERSITY of TENNESSEE KNOXVILLE