Tutorial: Machine Learning in Materials Science — From Basic Concepts to Active Learning

Arun Mannodi Kanakkithodi

School of Materials Engineering
Purdue University, West Lafayette, IN

IMRC 2025, Cancun, Mexico
Monday, Aug 18, 2025, 8.30 am – 12.30 pm

Email: amannodi@purdue.edu

PURDUE UNIVERSITY | School of Materials Engineering

Tutorial Outline

No.	Instructor	Topic	Duration
1	Arun Mannodi Kanakkithodi	Introduction to ML for materials science, some high-level examples,	1:30 hrs
2		Break	15 mins
3	Arun Mannodi Kanakkithodi	Gaussian Process Regression and Active Learning	1 hr
4		Break	15 mins
5	Arun Mannodi Kanakkithodi	Overview of neural networks for regression and classification models,	1 hr.

About Me

Arun Kumar Mannodi Kanakkithodi

Assistant Professor of Materials Engineering

Contact Information

Office: DLR 103F
E-mail: amannodi@purdue.edu

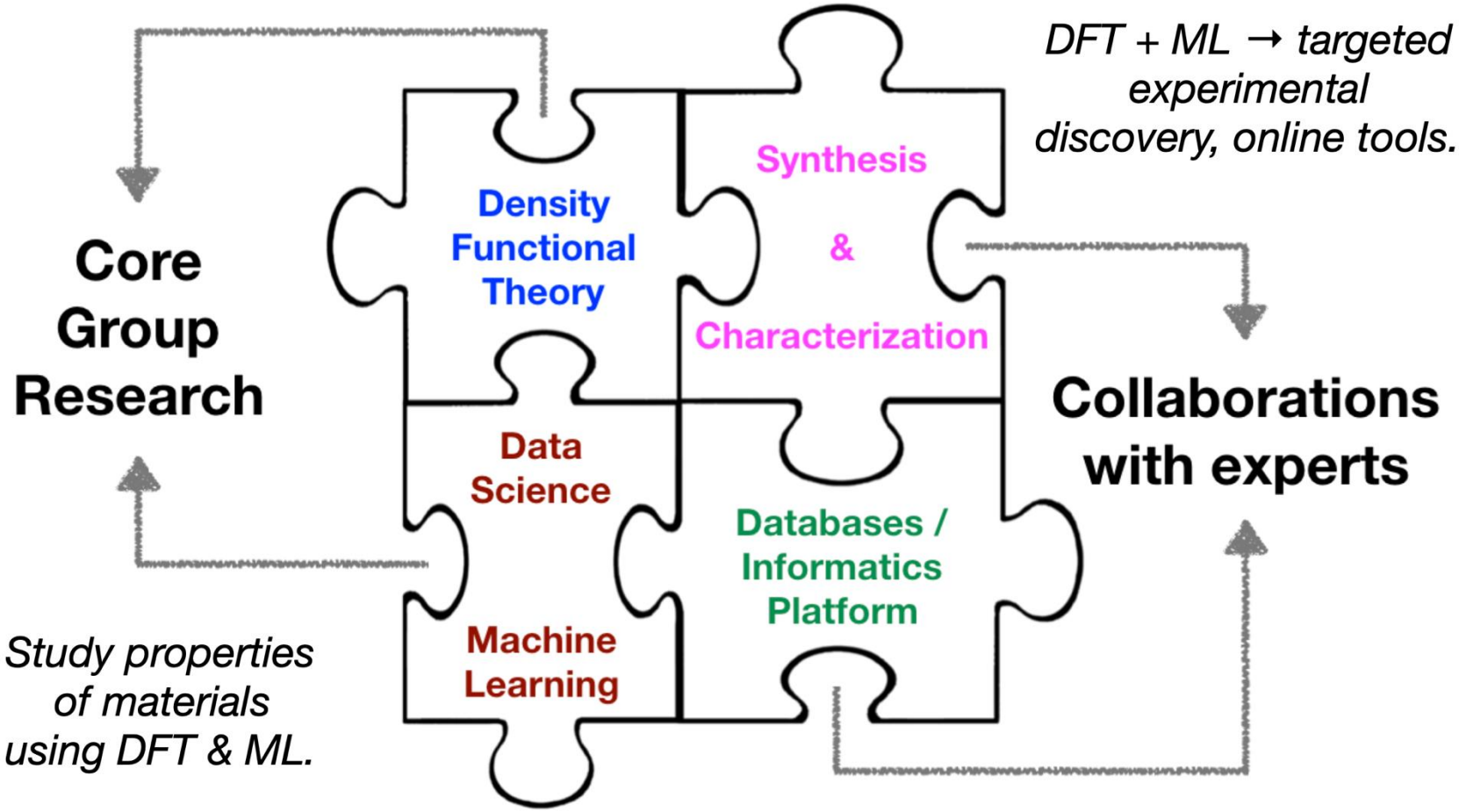
Research Group Web Site

School of Materials Engineering
Neil Armstrong Hall of Engineering
701 West Stadium Avenue
West Lafayette, IN 47907-2045

Research keywords: density functional theory (DFT), high-throughput computations, materials informatics, machine learning, data-driven discovery.

https://www.mannodigroup.com/

My Research Group



https://www.mannodigroup.com/

MRS Bulletin Article

A framework for materials informatics education through workshops

Arun Mannodi-Kanakkithodi,* Austin McDannald, Shijing Sun, Saaketh Desai, Keith A. Brown, and A. Gilad Kusne

The burgeoning field of materials informatics necessitates a focus on educating the next generation of materials scientists in the concepts of data science, artificial intelligence (AI), and machine learning (ML). In addition to incorporating these topics in undergraduate and graduate curricula, regular hands-on workshops present the most effective medium to initiate researchers to informatics and have them start applying the best AI/ML tools to their own research. With the help of the Materials Research Society (MRS), members of the MRS AI Staging Committee, and a dedicated team of instructors, we successfully conducted workshops covering the essential concepts of AI/ML as applied to materials data, at both the Spring and Fall Meetings in 2022, with plans to make this a regular feature in future meetings. In this article, we discuss the importance of materials informatics education via the lens of these workshops, including details such as learning and implementing specific algorithms, the crucial nuts and bolts of ML, and using competitions to increase interest and participation.

doi:10.1557/s43577-023-00531-6 (2023)

PART 1:

Introduction to ML in Materials Science: Data, Descriptors, Regression and Classification Models

Materials Informatics

- *Informatics* is the study of the structure, behavior, and interactions of natural and engineered computational systems. The central notion is the transformation of information - whether by computation or communication.

- *Materials informatics* is a field of study that applies the principles of informatics to materials science and engineering to improve the understanding, use, selection, development, and discovery of materials.

- Machine learning (ML) / artificial intelligence (AI) / data science applied to materials science problems.

Some Definitions

- AI: The branch of computer science that aims to replicate or simulate human intelligence in machines. Origins of AI lie in Alan Turing's "thinking machines".

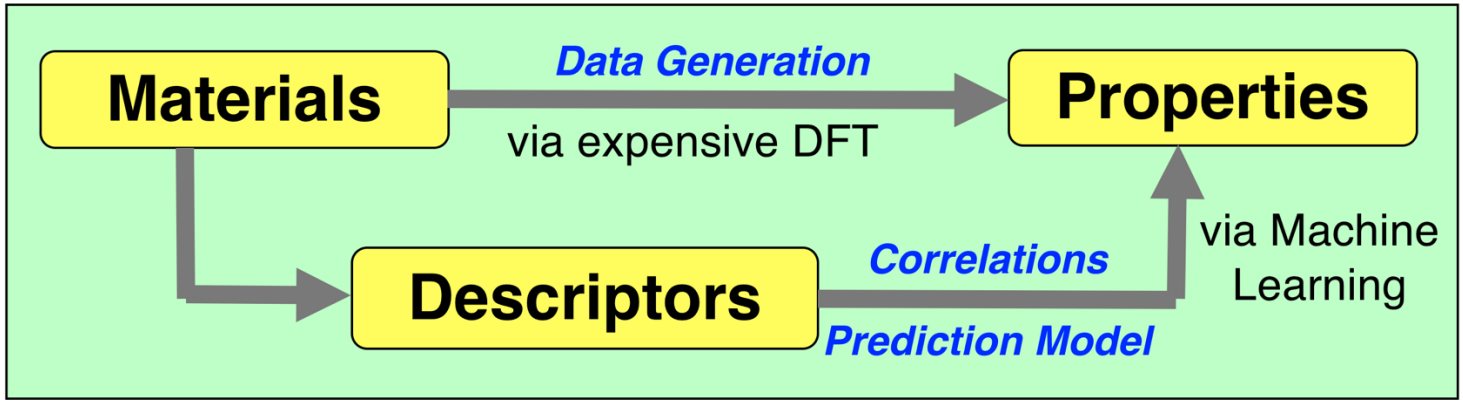
- ML: The study of computer algorithms that improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence.

- Data Science: Extracting knowledge from structured and unstructured data using scientific methods, processes, algorithms and systems. Includes data capture, communication, analysis, processing and maintenance.

When should ML be used?

- When fundamental laws underlying a process don't exist [e.g., social science problems].
- When such fundamental laws may exist, but are enormously complex [e.g., weather prediction].
- When we have a lot of data and we are looking for simple rules and correlations [e.g., Hall-Petch equation].
- In materials science: to uncover trends in the behavior of materials / to predict a new material's behavior / to accelerate computations, experiments and discovery.

Machine Learning in Materials Science



a Example dataset

Material	Property Value
Material 1	P_1
Material 2	P_2
⋮	⋮
Material *N*	P_N

b The learning problem

Material	Property Value
Material *X*	?

c Fingerprinting, learning and prediction

Material	Fingerprint	Property Value
Material 1	$F_{11}, F_{12}, \ldots F_{1M}$	P_1
Material 2	$F_{21}, F_{22}, \ldots F_{2M}$	P_2
⋮	⋮	⋮
Material *N*	$F_{N1}, F_{N2}, \ldots F_{NM}$	P_N

Fingerprinting

Learning

$f(F_{i1}, F_{i2}, \ldots, F_{iN}) = P_i$

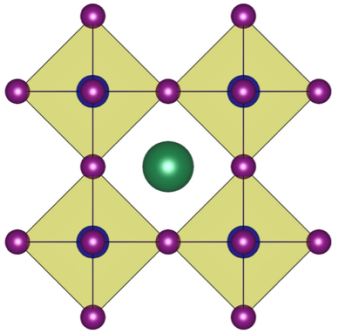
Prediction Model

Key Ingredient of ML: Feature Vectors / Materials Descriptors / Fingerprints

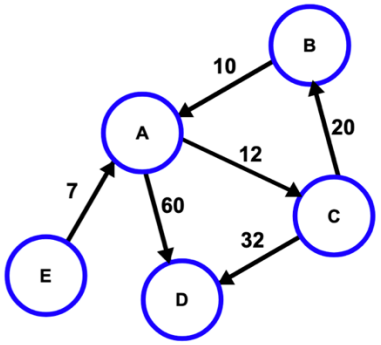
- Numerical representation of materials, input to ML.
- Definition depends on: a) application, b) domain expertise, and c) accuracy desired.
- Requirements: a) intuitive and inexpensive to calculate, b) generalizable to every material in the chemical space, and c) invariant to translation / rotation / permutation of like elements.

Examples of Fingerprints

3D geometry:
Atom i = (Z,x,y,z)



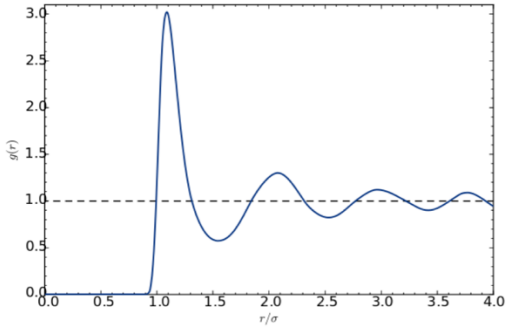
Weighted graph:
atoms & bonds



Coulomb Matrix

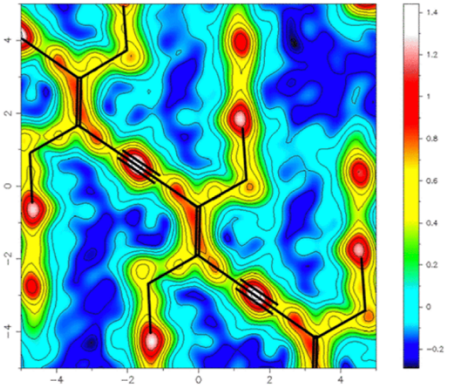
$$M_{IJ} = \begin{cases} 0.5Z_I^{\,2.4} & \text{for } I = J \\ \dfrac{Z_I Z_J}{\left|\boldsymbol{R}_I - \boldsymbol{R}_J\right|} & \text{for } I \neq J \end{cases}$$

Radial Distribution Function

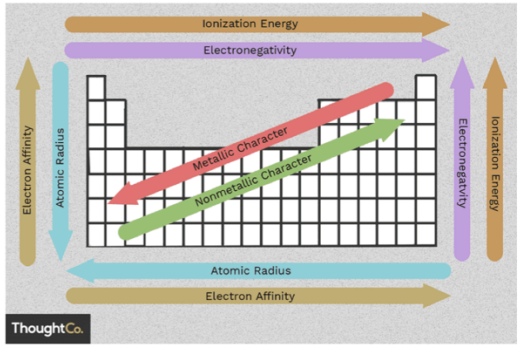


$A_i(\eta) = \exp(-r_{ij}^2/\eta^2)*f(r_{ij})$

Electron Density Distribution



Tabulated elemental properties



Materials Descriptors

	H	Li	Be	B	C	N	O	F	…
LiH	1	1	0	0	0	0	0	0	…
LiF	0	1	0	0	0	0	0	1	…
BeO	0	0	1	0	0	0	1	0	…
BN	0	0	0	1	0	1	0	0	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	…

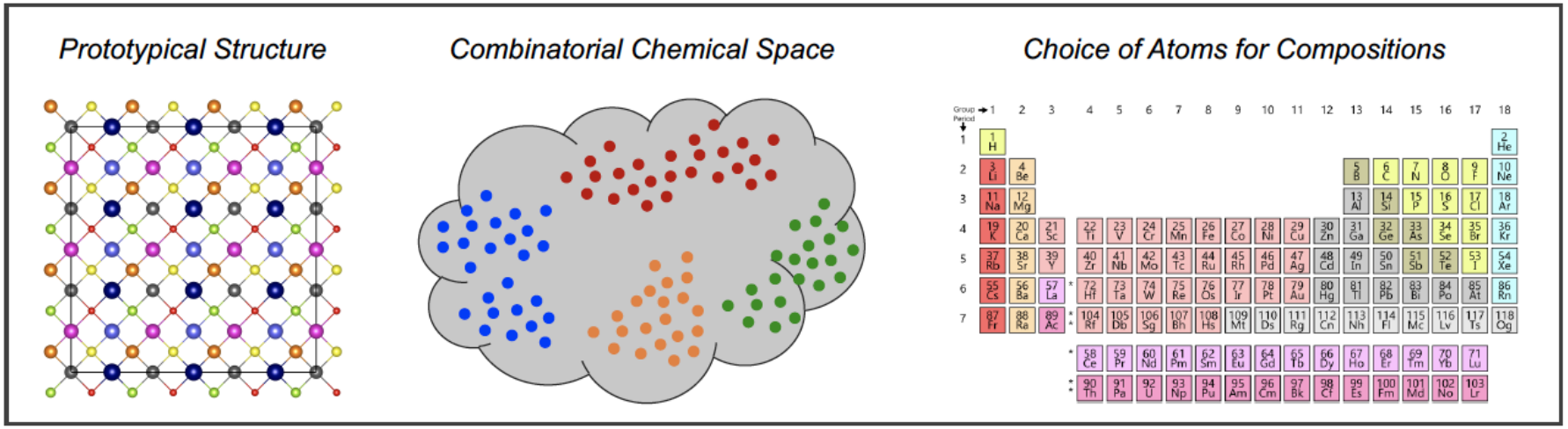
I. Tanaka (ed.), *Nanoinformatics*, https://doi.org/10.1007/978-981-10-7617-6_1

Level 1:
What are the atoms (A) → indices / elemental properties / DFT properties

Level 2:
What is the composition (C) → indices / elemental properties / DFT properties

Level 3:
What is the structure (S) → atomic coordinates / RDF / CM

Materials Descriptors

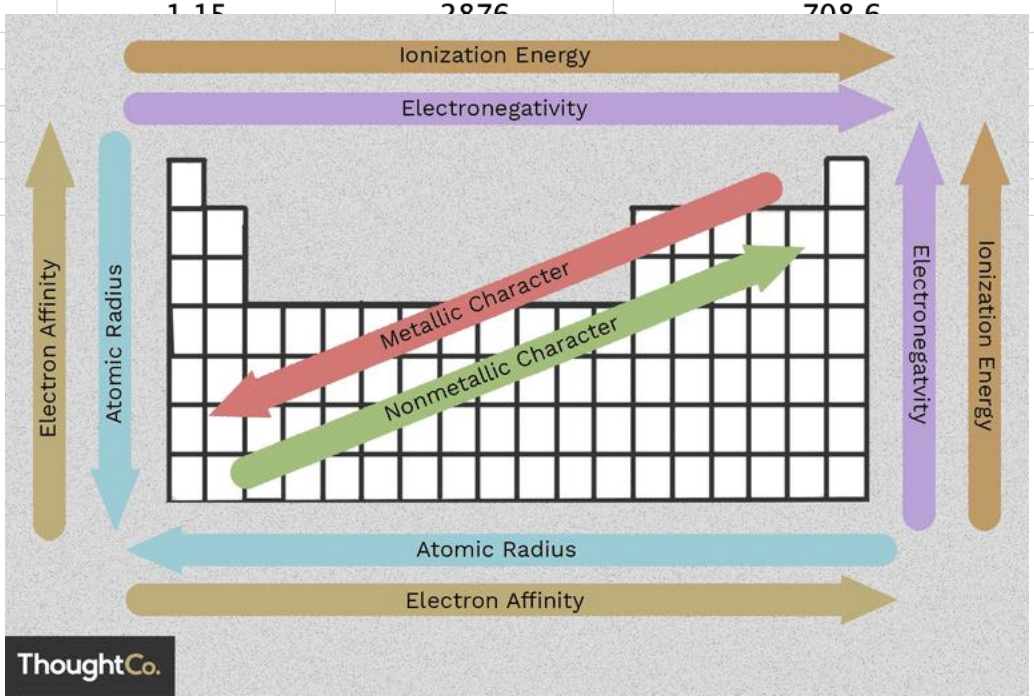


The set of descriptors $X^k = \{ X_1^k, X_2^k, \ldots X_m^k \}$ should:

- account for all necessary A-C-S information,
- apply to every material in the chemical space, and
- be represented equally in the training set, that is, each normalized X_i should add up to similar number.

Elemental Properties

Symbol	Ionic_radius (Å)	Boiling_point (K)	Ionization_energy (kJ/mol)	Electronegativity	Atomic_weight (u)	…
K	1.51	1033	418.8	0.82	39.0983	
Rb	1.61	961	403	0.82	85.4678	
Cs	1.74	944	375.7	0.79	132.9054	
Ca	1	1757	589.8	1	40.078	
Sr	1.26	1655	549.5	0.95	87.62	
Ba	1.42	2078	502.9	0.89	137.33	
Ge	0.87	3107	762.1	2.01	72.61	
Sn				1.96	118.71	
Pb				2.33	207.2	
Cl				3.16	35.4527	
Br				2.96	79.904	
I				2.66	126.9045	



Accounts for atom + composition information, encodes chemical and physical behavior, electronic features, etc.

Structural Representations

Several pairwise and angular-dependent structural representations can be used, such as partial radial distribution function (PRDF), generalized radial distribution function (GRDF), bond-orientational order parameter (BOP), and angular Fourier series (AFS).

$$\mathrm{GRDF}_n^{(i)} = \sum_j f_n(r_{ij}) \qquad f_n(r) = \exp\left[-p_n(r-q_n)^2\right] f_c(r)$$

$$Q_l^{(i)} = \left[\frac{4\pi}{2l+1}\sum_{m=-l}^{l}|Q_{lm}^{(i)}|^2\right]^{1/2} \qquad \mathrm{AFS}_{n,l}^{(i)} = \sum_{j,k} f_n(r_{ij}) f_n(r_{ik}) \cos(l\theta_{ijk})$$

A. Seko et al., Phys. Rev. B. 95, 144110 (2017).

Feature Selection / Dimensionality Reduction

- Filter-based: Define large-dimensional space, eliminate dimensions based on Pearson correlation coefficient.

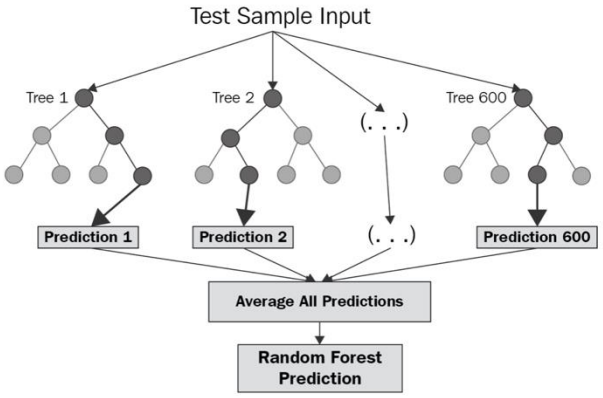
$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

- Wrapper-based: Recursive feature elimination, train ML models on subsets, eliminate ones that don't matter.

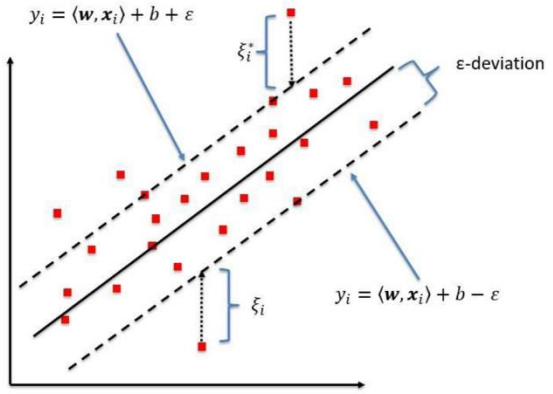
- Embedded: Methods like LASSO, random forest, even neural networks, can apply feature importance scores.

Examples of ML Techniques

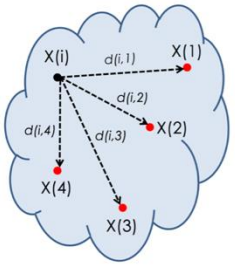
Random Forest Regression



Support Vector Regression



Kernel Ridge Regression



$X(i) = \{x_1, x_2, x_3 \dots x_m\}$

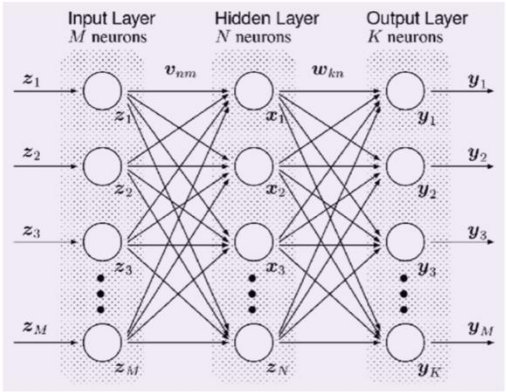
Measure of Similarity: Euclidean Distance

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + \dots + (x_{im} - x_{jm})^2}$$

Property = Weighted sum of Gaussians

$$f(i) = \sum_{k=1}^{N} a_k \cdot \exp\left(-\frac{1}{2\sigma^2} \cdot [d(i, i_k)]^2\right)$$

Neural Networks



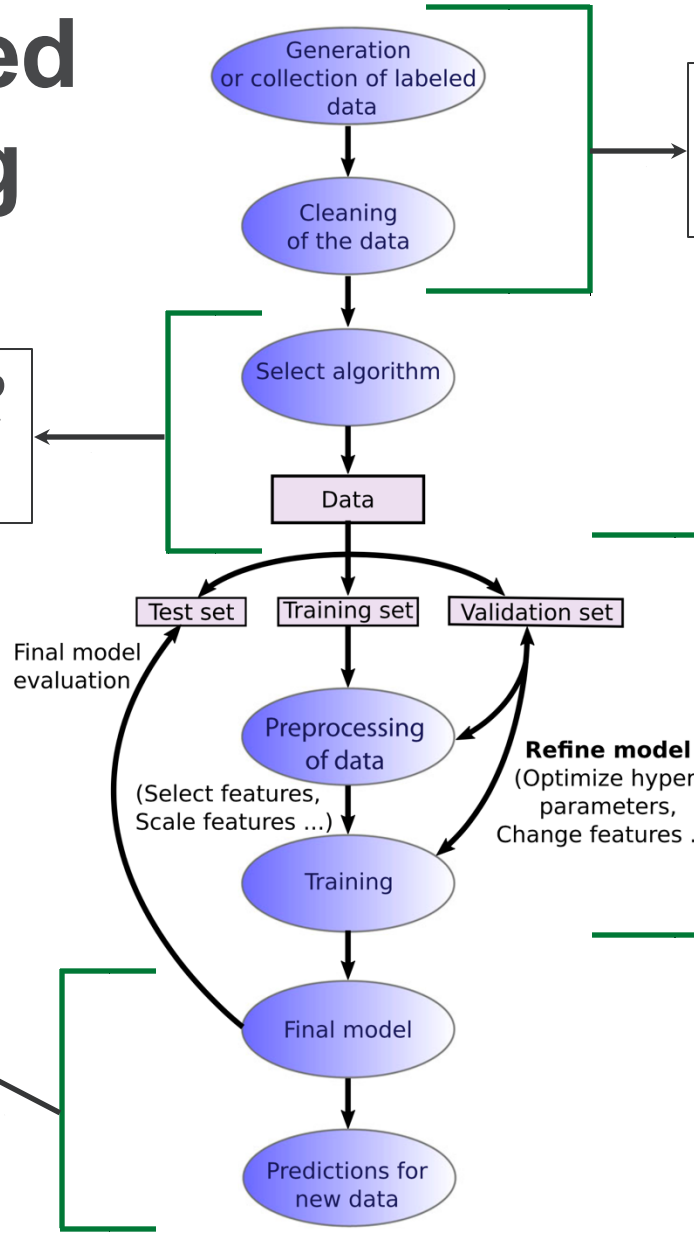
Machine Learning Models

- Three types: supervised, unsupervised, and reinforcement learning.
- Nuts and bolts of training a ML model: inputs and outputs, error quantification, cross-validation and hyperparameter optimization.
- Classification and regression techniques: KNN, linear regression, KRR, random forest, etc.
- Optimization: Genetic Algorithm, active learning, Bayesian optimization.

Types of Machine Learning

- Supervised learning: From labeled training data, find the unknown function connecting known inputs to unknown outputs, based on extrapolation of patterns.
- Unsupervised learning: Find patterns in unlabeled data, leading to clustering of samples.
- Semi-supervised learning: Representations learned from a mix of unlabeled and labeled data.
- Reinforcement learning: Finding optimal or sufficiently good actions for a situation to maximize a reward.

Supervised Learning



Data: Properties + Descriptors

LR / KRR / GPR / RFR / NN

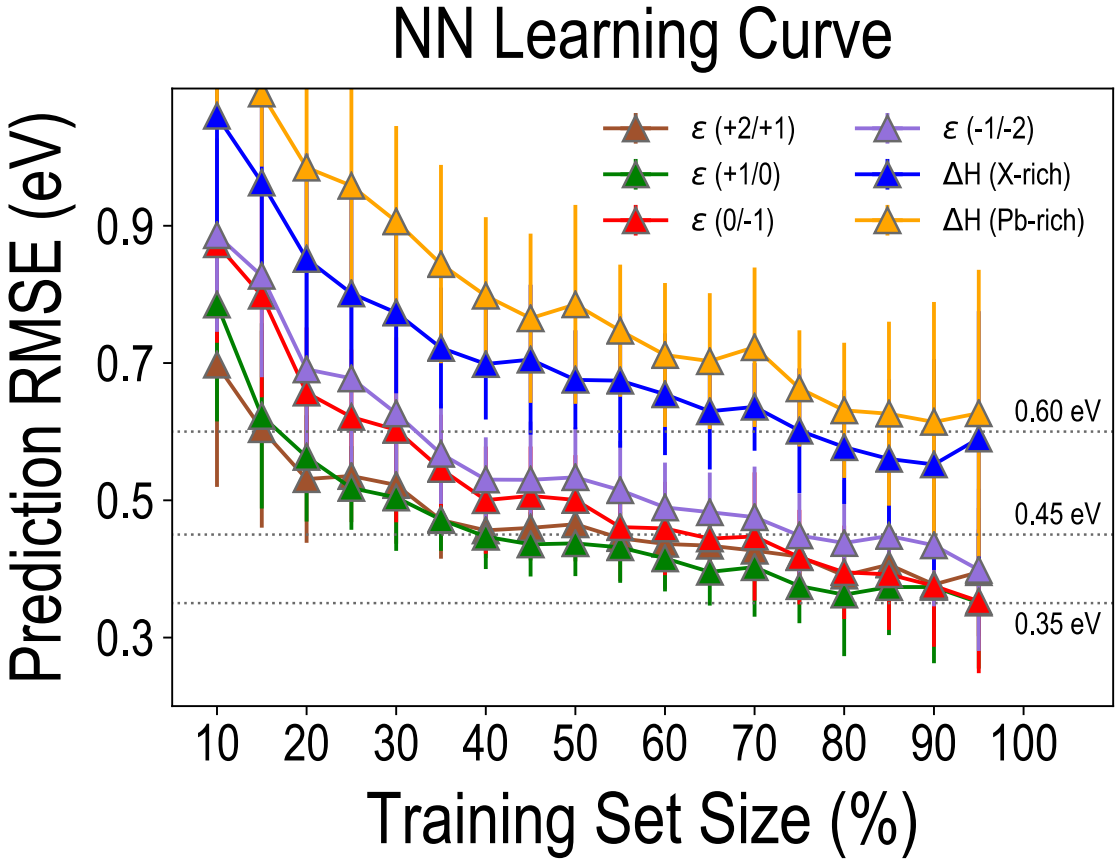
Standard practices: train-test split, cross-validation, hyper-parameter optimization

Best models → predictions → screening

Nuts and Bolts of ML

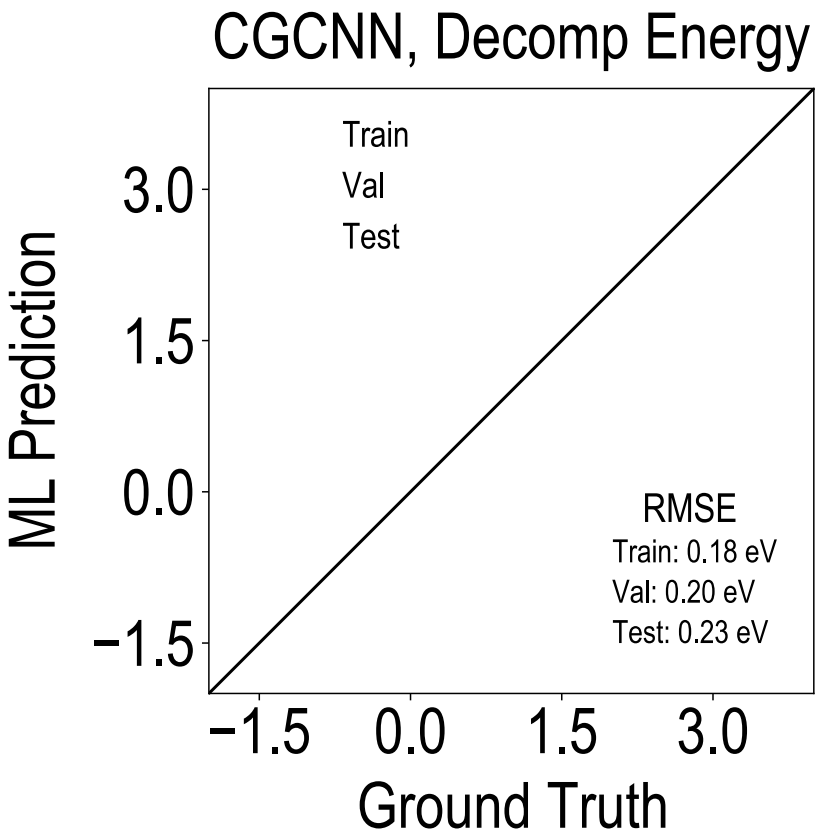
- Data: Divide into training, validation, and test sets.
- Descriptors: enumerate for all data points, perform dimensionality reduction and feature scaling.
- Cross-validation: n-fold, leave-one-out.
- Hyperparameter optimization: grid-search, Bayesian.
- Best model: optimized w.r.t. training data and descriptor size, CV, and HPO; choose error definition.
- Quality and quantity of data and descriptors: prevent underfitting. CV, HPO, regularization: prevent overfitting.

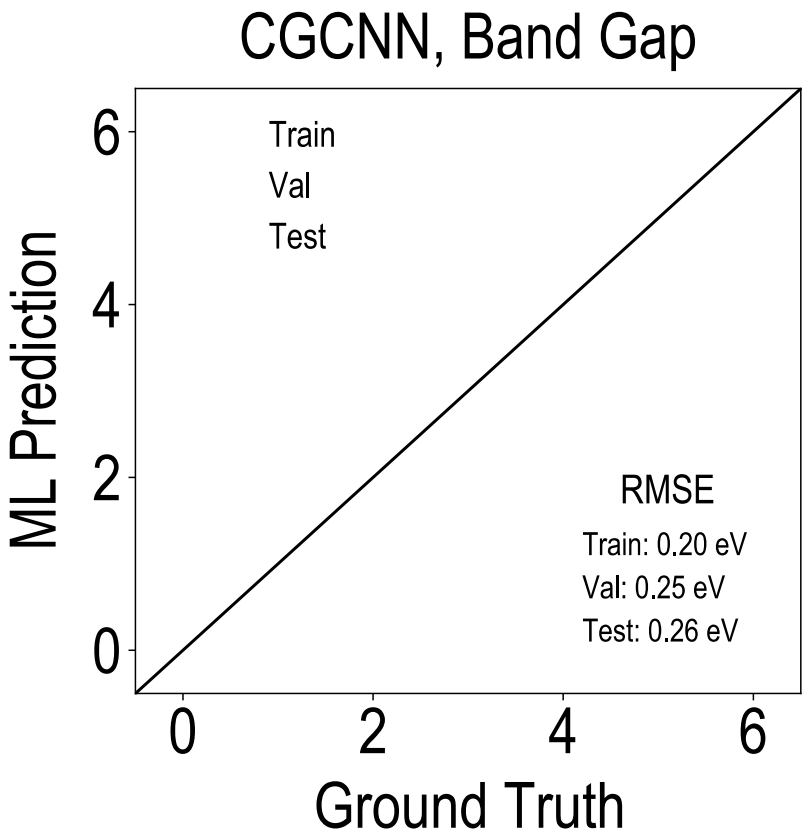
Training Data Size: Learning Curves



Iteratively change training set size (and descriptor dimensions) until test error saturates → learning curve.

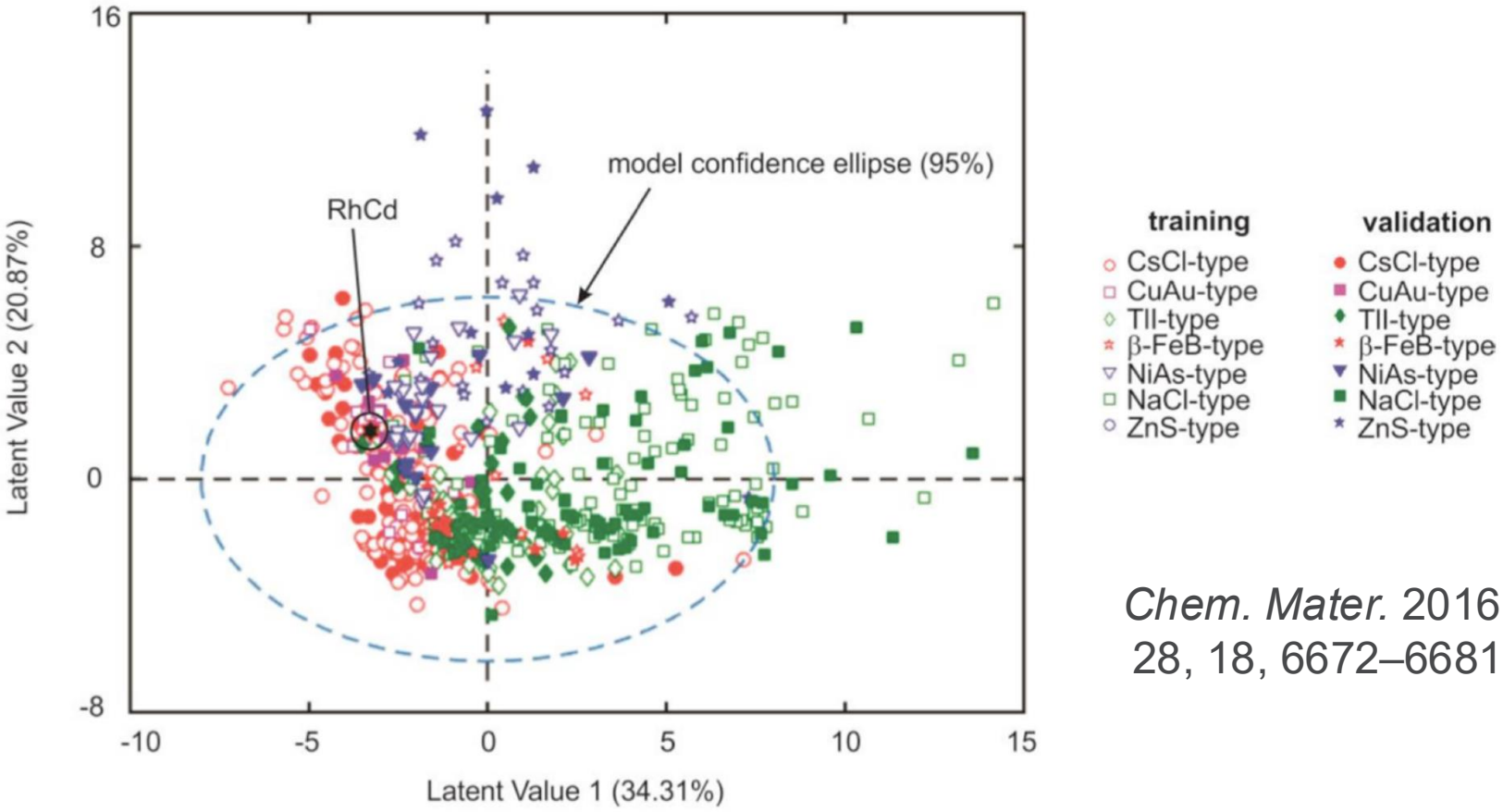
Example: ML Regression





Split data → optimize model → visualize as parity plots.

Example: ML Classification



Chem. Mater. 2016, 28, 18, 6672–6681

Classifying AB compounds into different structure types.

Feature Scaling

- ML algorithms may not perform well when input numerical attributes have very different scales.
- Two common methods: normalization & standardization.
- Normalization: Min-max scaling. All values subtracted by min value and divided by (max – min), so they range from 0 to 1. E.g., *import sklearn.MinMaxScaler*.
- Standardization: subtract mean value and divide by variance, such that mean = 0 and variance = 1. This approach is less affected by outliers.

Error Quantification

- The accuracy of ML predictions are tested using RMSE, MAE, etc. E.g., *sklearn.metrics.mean_squared_error*.

- Coefficient of determination (R^2):
  $R^2 = 1 - (\Sigma_i(Y^{truth}_i - Y^{pred}_i)^2 / (\Sigma_i(Y^{truth}_i - Y_{mean})$

- Root mean square error (RMSE):
  $RMSE = [\Sigma_i(Y^{truth}_i - Y^{pred}_i)^2 / N]^{1/2}$

- Mean absolute error (MAE):
  $MAE = \Sigma_i |Y^{truth}_i - Y^{pred}_i| / N$

- Other measures: accuracy, precision, recall.

Cross Validation & Regularization

- n-fold CV: Divide training set into n subsets, train using (n-1) subsets at a time and validate with the n^{th} set. Averaging over the n subsets provides final parameters, training error and cross-validation error. Allows for more general predictions on new and/or noisy data.

- Leave one out (LOO) CV: For N points, train on (N-1) points at a time and test on N^{th} point. Report models by averaging over all N iterations.

- Regularization: Constrains the model to make it simpler and reduce overfitting. A tuning parameter decides how much we want to penalize the flexibility of our model.

Hyperparameter Optimization

- Rigorously tune values of ML model hyperparameters to yield least errors and most general predictions.

- Random search, grid search, or optimization using Bayesian approach or genetic algorithm.

- E.g., random forest regression:

```
param_grid = {   "n_estimators": [50, 100, 200],
                 "max_depth": [5, 10, 15],
                 "max_features": [m-10, m-5, m],
                 "min_samples_leaf": [5, 10, 20],
                 "min_samples_split": [2, 5, 10]    }
rfreg_opt = GridSearchCV(RandomForestRegressor(),
param_grid=param_grid, cv=5)
```

ML Classification

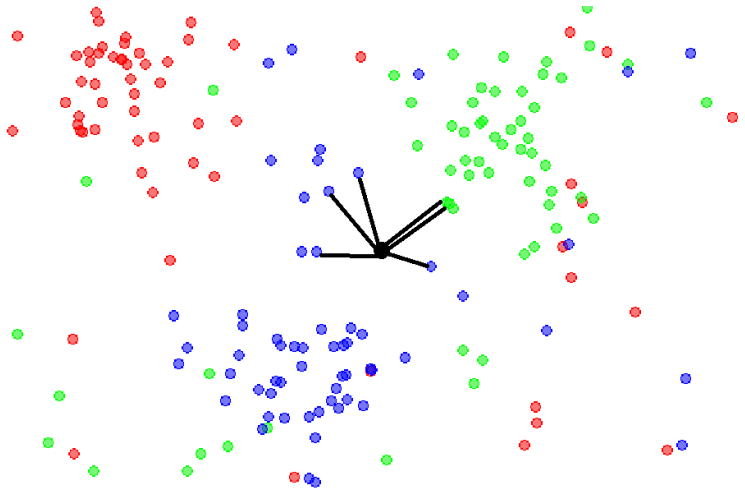
- Divide a labeled dataset of points into categories.

- Example: email spam filter, weather prediction, getting approved for a credit card, etc.

- Techniques:
  - Naïve Bayes Approach
  - K-Nearest Neighbors
  - Logistic Regression
  - Support Vector Machine
  - Decision Trees / Random Forest
  - Neural Networks

K-Nearest Neighbors

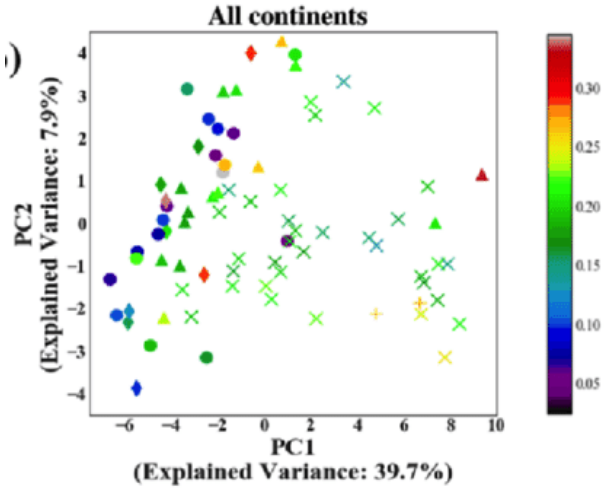
- Estimate the classification of an unseen instance using the labels of the instances it is closest to.
- Find K training points closest to new point, take the most commonly occurring classification of the K points.



- Closeness of points: Euclidean distance or other distance measures.
- Normalizing data and weighting distance may be important.

Principal Component Analysis

- Projecting data into a hyperplane that lies closest to the data. Used for dimensionality reduction and clustering.
- Convert m-dimensional data into m orthogonal principal components that capture largest to lowest amount of variance in the training data.



- Ideal scenario: PC1 and PC2 together capture all the variance. This allows for real 2D projection and visualization.
- import sklearn.decomposition.PCA

ML Regression

- Train a model to make a quantitative estimate of a desired value, that is, $Y^k = f(X_i^k)$.

- Ways to best optimize a regression model:
  - Use a closed-form equation that directly computes the model parameters that best fit the training data.
  - Use a gradient descent approach to gradually tweak model parameters to minimize cost function.

- From Scikit-learn, call desired regressor (and related packages): from sklearn.linear_model import LinearRegression, from sklearn.ensemble import RandomForestRegressor, etc.

Linear Regression

- Linear regression model prediction form:
  $y = \boldsymbol{\theta}_0 + \boldsymbol{\theta}_1 x_1 + \boldsymbol{\theta}_2 x_2 + \ldots + \boldsymbol{\theta}_m x_m$; or, $y = h_{\boldsymbol{\theta}}(x) = \boldsymbol{\theta}^T.\mathbf{x}$

- MSE cost function for a linear regression model:

$$\mathrm{MSE}(\mathbf{X}, h_\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(\theta^T \cdot \mathbf{x}^{(i)} - y^{(i)}\right)^2$$

- Direct estimation of $\boldsymbol{\theta}$ using the formula below (indirect method uses GD):

$$\hat{\theta} = \left(\mathbf{X}^T \cdot \mathbf{X}\right)^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y}$$

- Easy extension to Polynomial regression.

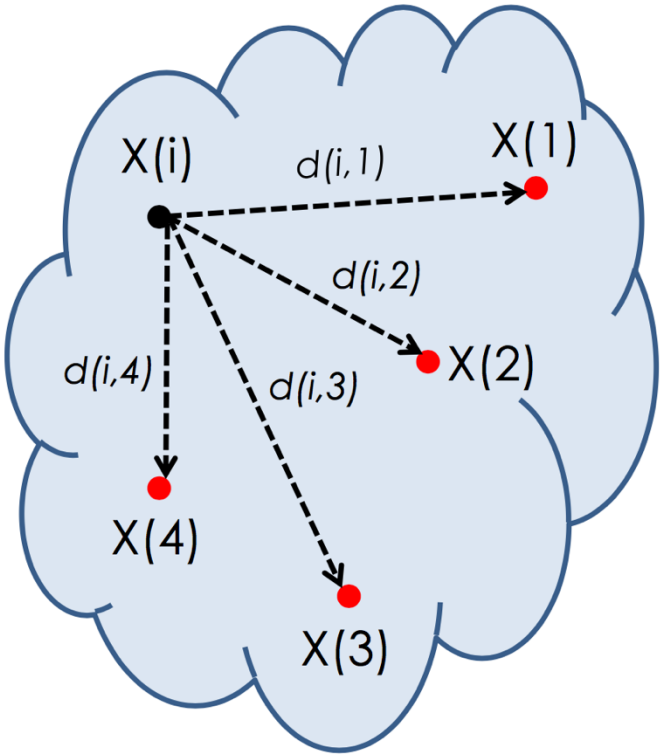
Ridge & LASSO Regression

- Regularized version of linear regression where a regularization term = $\alpha\Sigma_i\boldsymbol{\theta}_i^2$ is added to the cost function to keep the model weights small.
- Ridge regression cost function $J(\boldsymbol{\theta}) = MSE(\boldsymbol{\theta}) + \alpha\Sigma_i\boldsymbol{\theta}_i^2$
- LASSO regression cost function $J(\boldsymbol{\theta}) = MSE(\boldsymbol{\theta}) + \alpha\Sigma_i|\boldsymbol{\theta}_i|$
- L_2 norm vs L_1 norm: $|\mathbf{x}| = \sqrt{x_1^2 + x_2^2 + x_3^2}\,.$ $\quad |\mathbf{x}|_1 = \sum_{r=1}^{n} |x_r|.$
- Ridge regression closed-form solution (GD-based optimization also common):

$$\hat{\theta} = \left(\mathbf{X}^T \cdot \mathbf{X} + \alpha\mathbf{A}\right)^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y}$$

Kernel Ridge Regression

Chemical Space



$X(i) = \{x_1, x_2, x_3 \ldots x_m\}$

KERNEL RIDGE REGRESSION (KRR)

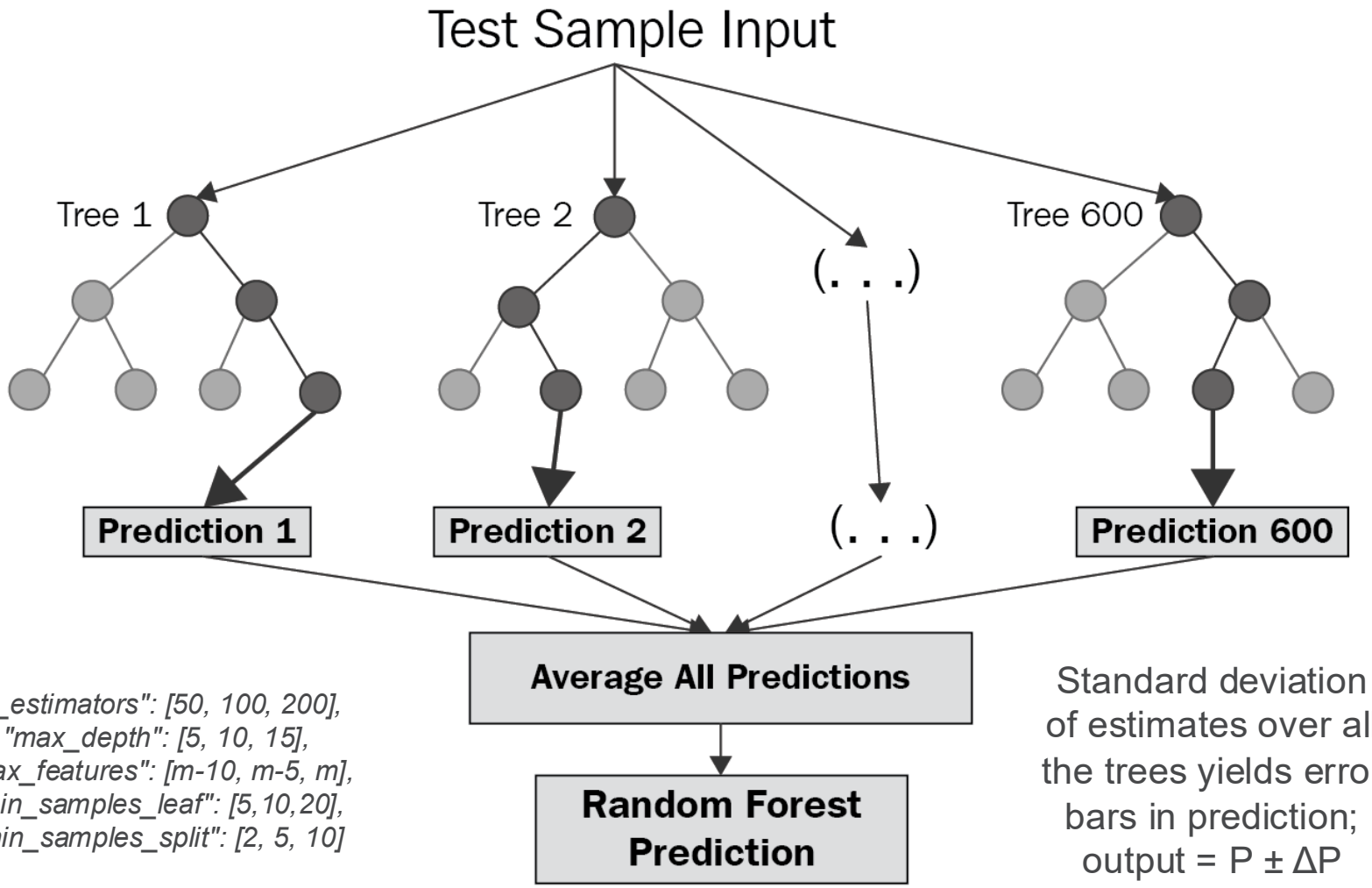
Measure of Similarity: Euclidean Distance

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + \ldots + (x_{im} - x_{jm})^2}$$

Property = Weighted sum of Gaussians

$$f(i) = \sum_{k=1}^{N} a_k \,.\, \exp\left(-\frac{1}{2\sigma^2} \,.\, [d(i, i_k)]^2\right)$$

Random Forest Regression



"n_estimators": [50, 100, 200],
"max_depth": [5, 10, 15],
"max_features": [m-10, m-5, m],
"min_samples_leaf": [5,10,20],
"min_samples_split": [2, 5, 10]

Standard deviation of estimates over all the trees yields error bars in prediction; output = P ± ΔP

CASE STUDY

- Designing novel ABX_3 perovskites for solar absorption.
- Check out data and Jupyter notebook here: **https://github.com/mannodiarun/IMRC-2025-ML-Tutorial**
- Directly open notebook on Google Colab: **https://colab.research.google.com/github/mannodiarun/IMRC-2025-ML-Tutorial/blob/main/ml-tutorial.ipynb**
- Shortened url: **https://tinyurl.com/3nn8fruc**

PART 2:

Gaussian Process Regression and Active Learning

Gaussian Process Regression

- GP: collection of Gaussian distributions over some domain as the input.
- GP relates similarity (co-variance) between inputs and outputs as defined by a Kernel function.
- In the same way that a Gaussian distribution is defined by the mean and variance, a Gaussian Process is defined by the mean function and co-variance function. Mean function is some f(X), where X is the input, while co-variance is the similarity between any X and X'.
- Predicted value = mean over predictive distribution, uncertainty = standard deviation over distribution.

Gaussian Process Regression

Why use a Gaussian Process?

Based on the measurements we've taken, we can infer the value at the function at new locations. Because of the kernel fuction these predictions are based on how similar the new locations are to the locations where we've taken measurements.

Since the GP is a collection of Gaussian distributions, we can calculate how likely data is given the model. We can use that information to find how likely the model is given the data.

This is a form of Bayesian Inference:

$$P(model|data) = \frac{P(data|model)P(model)}{P(data)}$$

Credit: Dr. Austin McDannald, NIST

https://github.com/mannodiarun/mrs_spring_tutorial/tree/GP_and_AL/GP_and_AL

Gaussian Process Regression

We can use GP's to interpolate and extrapolate from the measurements we've taken - with uncertainty!

We can encode physics into the kernel to make accurate predictions.

We can propagate uncertainties from measurements to the predictions.

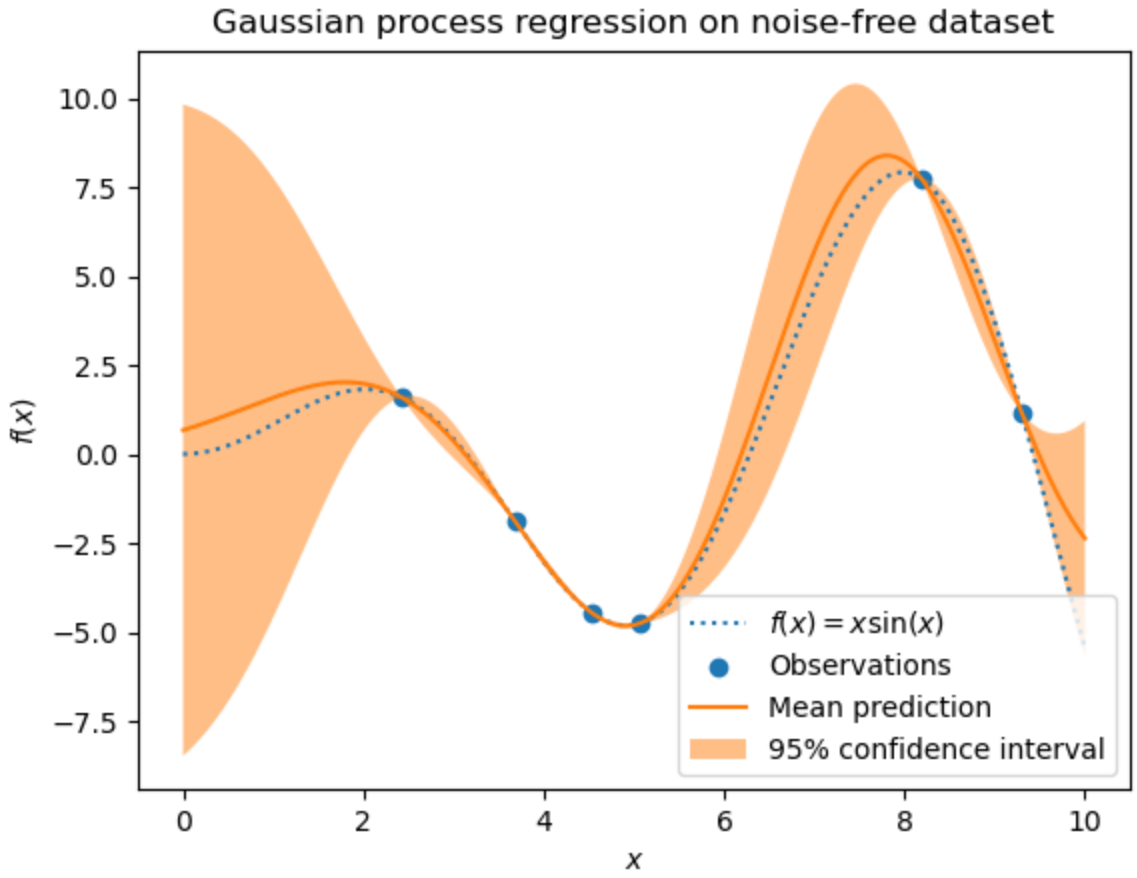
This lets us answer questions like:

- What can we conclude from these measurements?
- Where are we likely to find the optimum of a fuction?
- What regions of the input space are we uncertain about?

Credit: Dr. Austin McDannald, NIST

https://github.com/mannodiarun/mrs_spring_tutorial/tree/GP_and_AL/GP_and_AL

Gaussian Process Regression



https://scikit-learn.org/stable/modules/gaussian_process.html

GP Kernels

- Kernels or covariance functions determine the shape of prior and posterior of the GP. They define similarity.
- Many kernel functions should be tested and optimized for the given dataset.

ConstantKernel

$$k(x_i, x_j) = constant\_value \;\forall\; x_i, x_j$$

Radial Basis Function (RBF)

$$k(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j)^2}{2l^2}\right)$$

l = length-scale

WhiteKernel

$$k(x_i, x_j) = noise\_level \text{ if } x_i == x_j \text{ else } 0$$

Dot-Product Kernel

$$k(x_i, x_j) = \sigma_0^2 + x_i \cdot x_j$$

GP Kernels

- Kernels or covariance functions determine the shape of prior and posterior of the GP. They define similarity.
- Many kernel functions should be tested and optimized for the given dataset.

Rational Quadratic Kernel

$$k(x_i, x_j) = \left(1 + \frac{d(x_i, x_j)^2}{2\alpha l^2}\right)^{-\alpha}$$

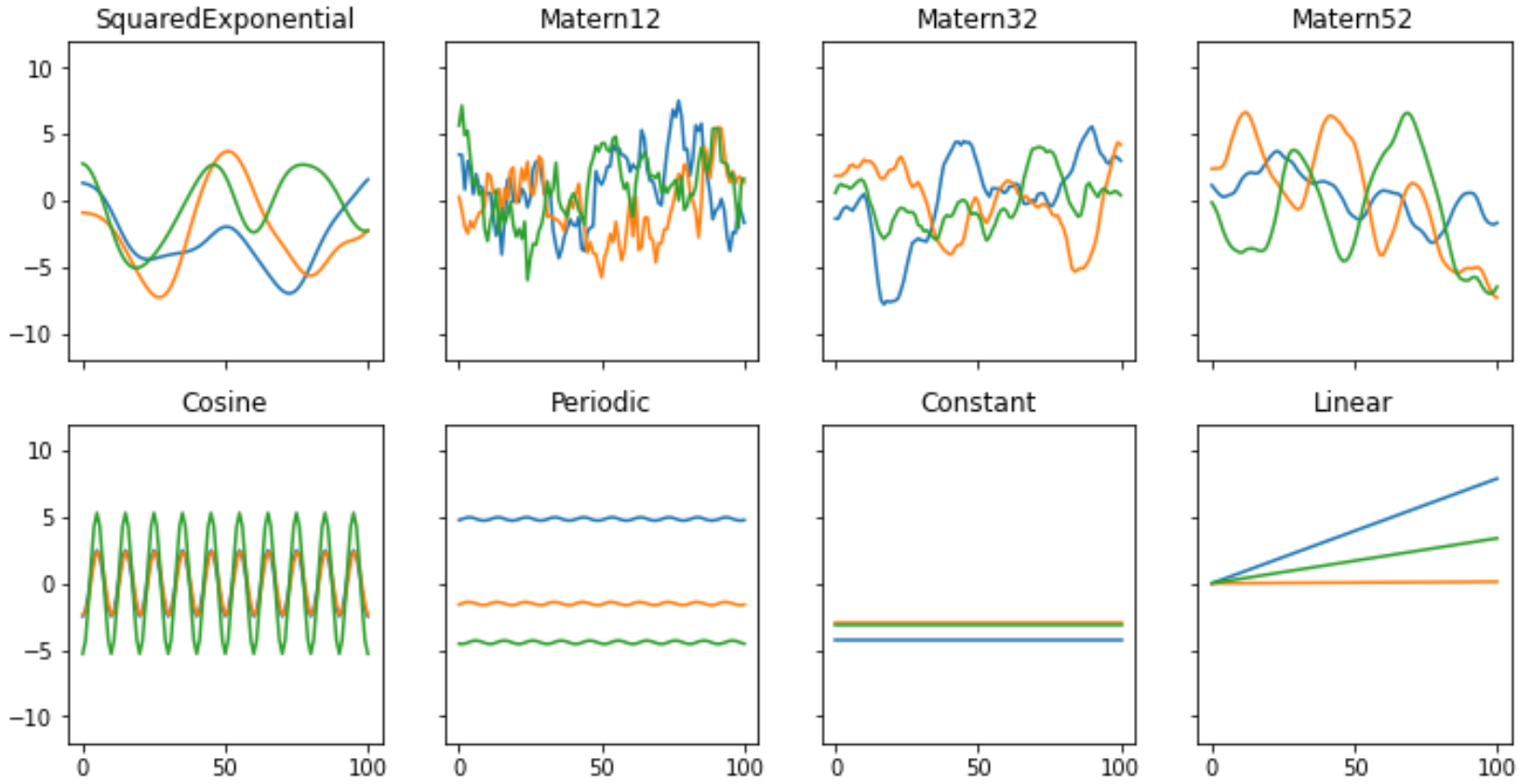
l = length-scale,
α = scale-mixture

Exp-Sine-Squared Kernel

$$k(x_i, x_j) = \exp\left(-\frac{2\sin^2(\pi d(x_i, x_j)/p)}{l^2}\right)$$

l = length-scale,
p = periodicity

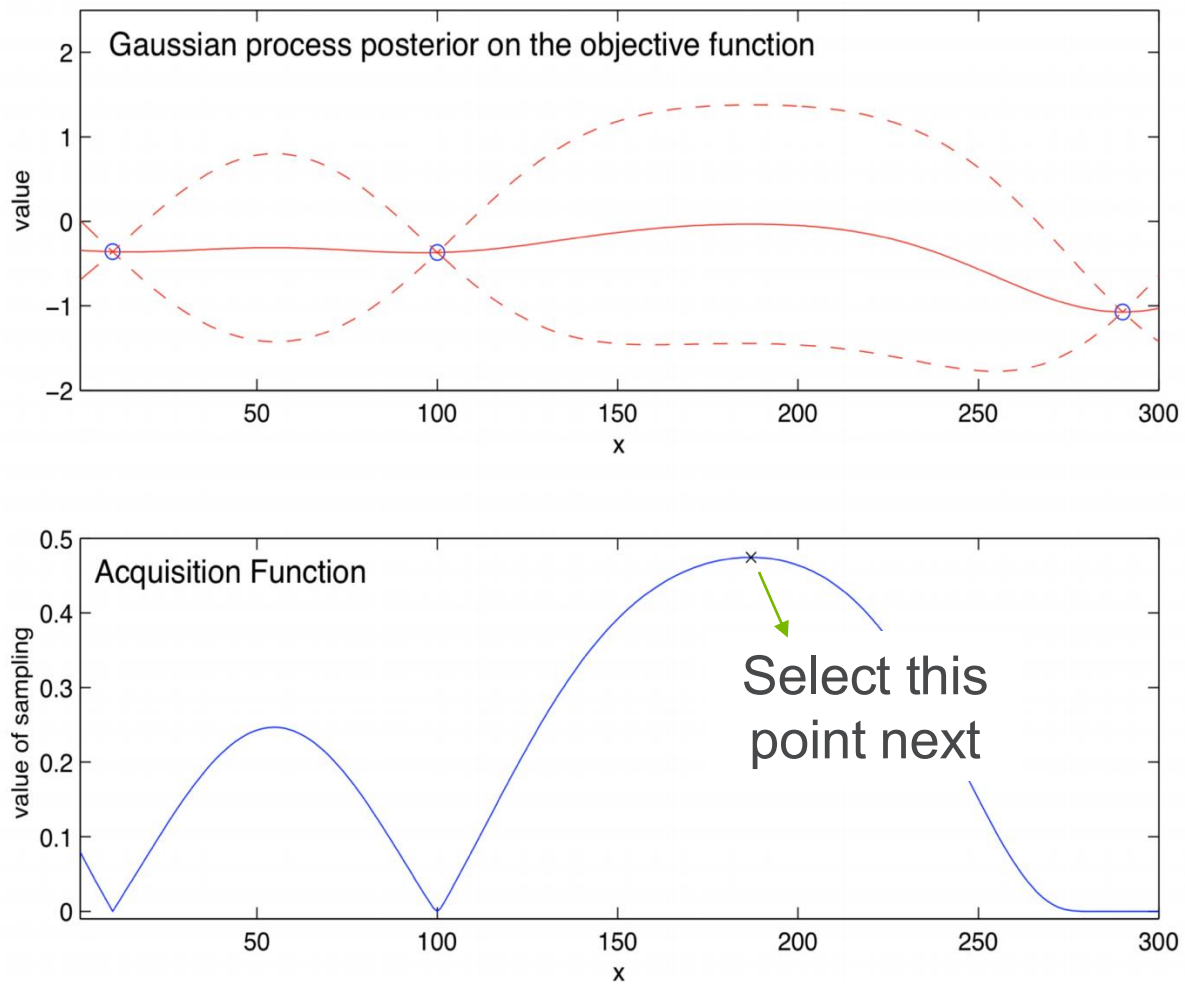
GP Kernels



Credit: Dr. Austin McDannald, NIST

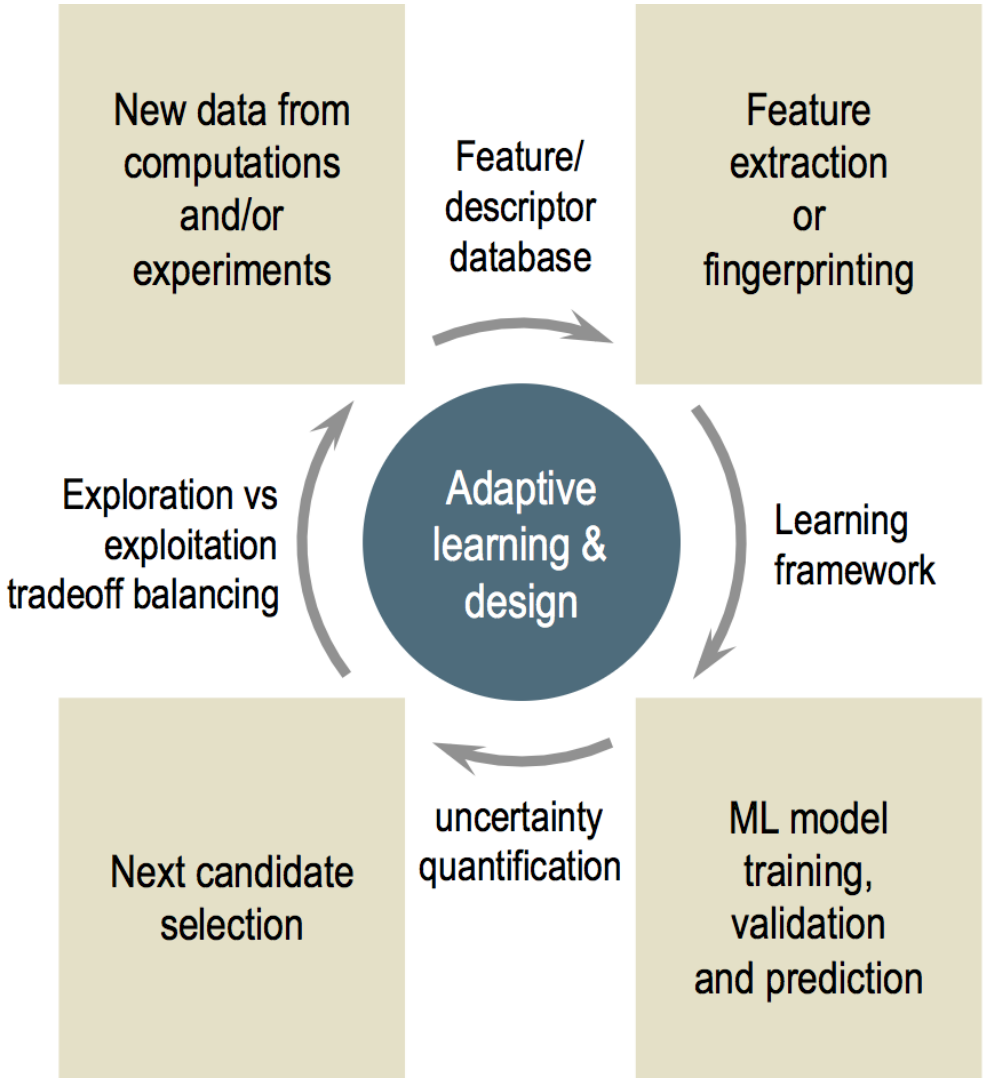
https://github.com/mannodiarun/mrs_spring_tutorial/tree/GP_and_AL/GP_and_AL

Bayesian Optimization



Based on maximizing reward and minimizing uncertainty, pick next data points, iteratively improve ML model predictions.

Active Learning



- Intelligent, sequential data generation using Bayesian optimization.
- At every step, predict output value and uncertainty (e.g., using GPR) for new points.
- Pick next point based on metrics such as highest uncertainty, maximum reward, or a combination.

Acquisition Functions

Upper Confidence Bound

$$UCB(x) = \mu(x) + \beta\sigma(x),$$

Probability of Improvement

$$PI(x) = \psi\left(\frac{\mu(x) - f(x^+) - \xi}{\sigma(x)}\right),$$

Expected Improvement

$$EI(x) = (\mu(x) - f(x^+) - \xi)\psi\left(\frac{\mu(x) - f(x^+) - \xi}{\sigma(x)}\right) + \sigma(x)\phi\left(\frac{\mu(x) - f(x^+) - \xi}{\sigma(x)}\right),$$

- $\mu(x)$ and $\sigma(x)$ are the mean and variance of the regressor at x, f is the function to be optimized.
- Pick next data point where UCB(x), PI(x), or EI(x) is maximum.

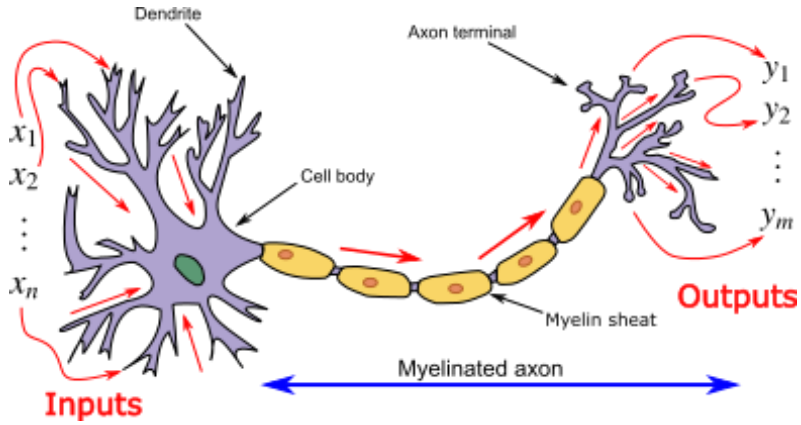
https://modal-python.readthedocs.io/en/latest/content/query_strategies/Acquisition-functions.html

- Open the Jupyter notebook again: **https://colab.research.google.com/github/mannodiarun/IMRC-2025-ML-Tutorial/blob/main/ml-tutorial.ipynb**
- Shortened url: **https://tinyurl.com/3nn8fruc**
- This time we will run the GPR and AL cells.

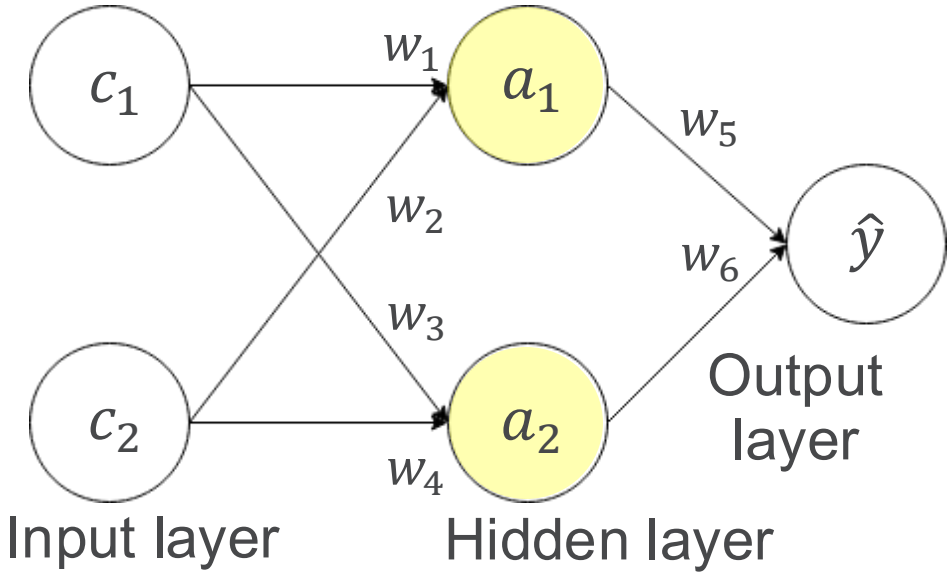
PART 3:

Neural Networks for Regression and Classification

Neural networks 101



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



activation

$$a_1 = f_1(w_1 c_1 + w_2 c_2 + b_1)$$

weight

bias

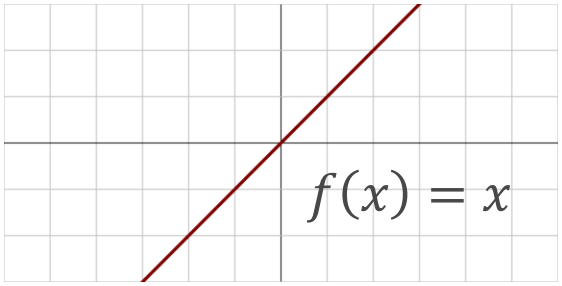
$$a_2 = f_2(w_3 c_1 + w_4 c_2 + b_2)$$

$$\hat{y} = Band\ gap = f_3(w_5 a_1 + w_6 a_2 + b_3)$$

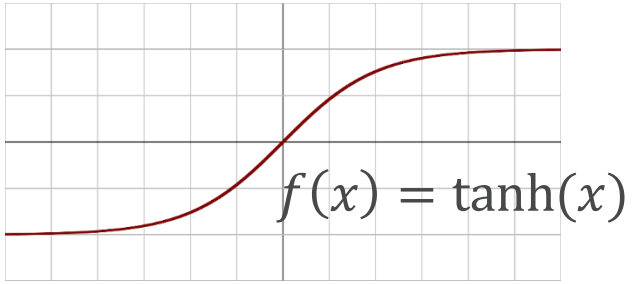
model prediction

Neural networks 101

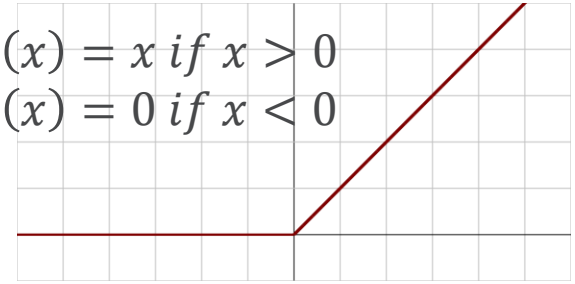
Linear

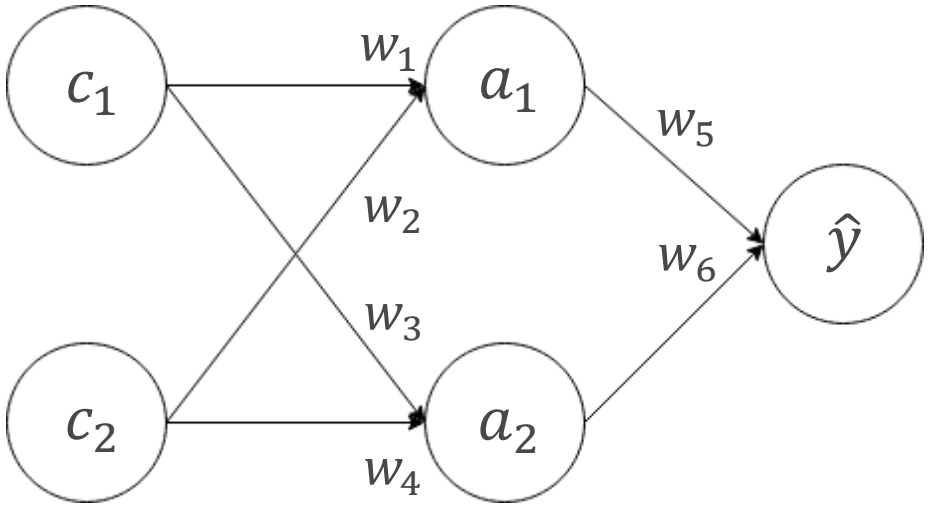


Tanh



Relu





$$a_1 = f_1(w_1 c_1 + w_2 c_2 + b_1)$$

$$Objective = \frac{1}{N_{samples}} \sum_{i=1}^{N_{samples}} (\hat{y}_i - y_i)^2$$

ground truth

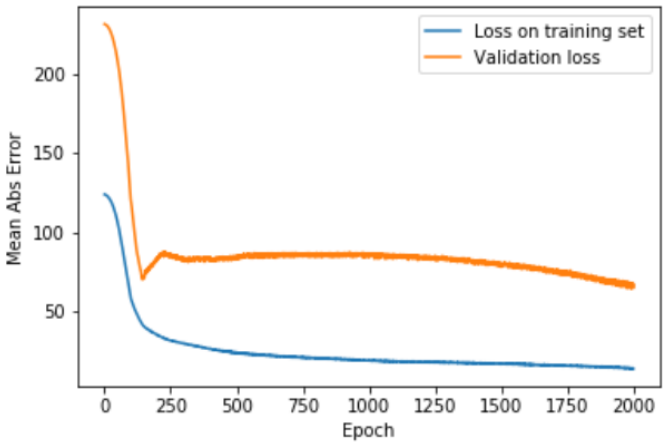
$$w_1 = w_1 - \alpha \frac{\partial(Objective)}{\partial w_1}$$

Backpropagation

Overfitting and underfitting

How do we judge if the model has learnt all that it could?

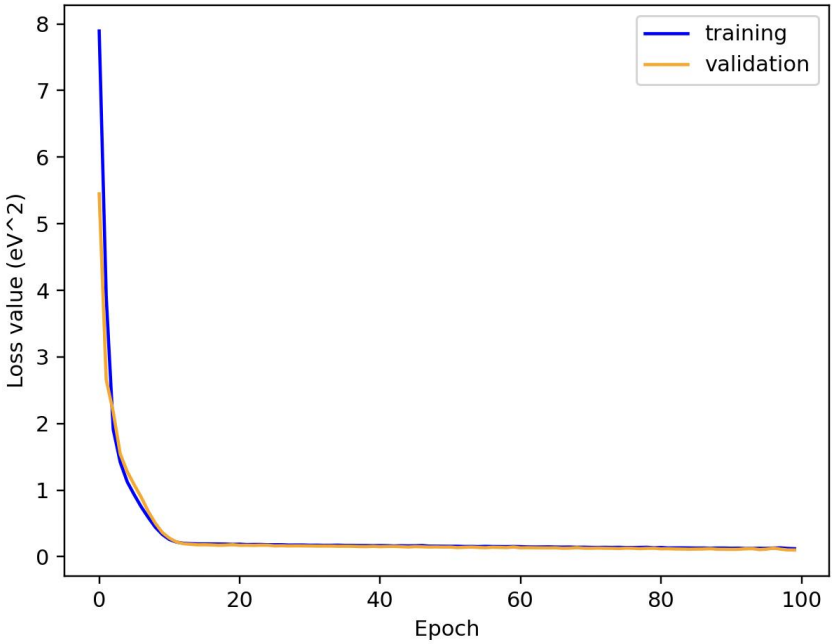
- Underfitting – model hasn't learnt all the trends in the training data
- Overfitting – model has "memorized" data, ignoring the underlying trend



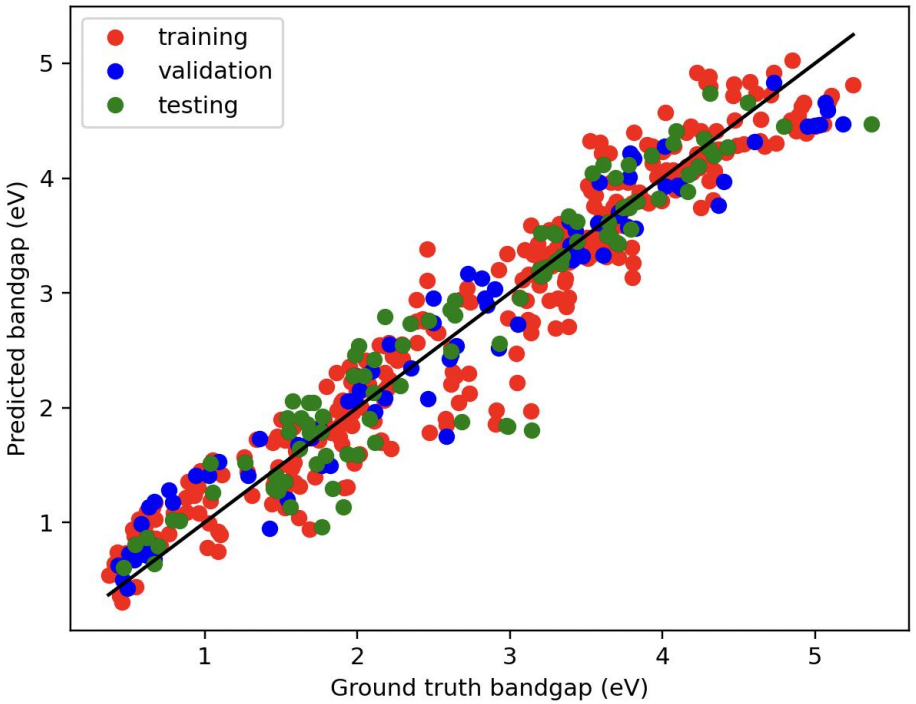
How do we train models that generalize well?

- Use a low enough learning rate
- Monitor error on validation set as a measure of model's ability to generalize

Visualizing the learning



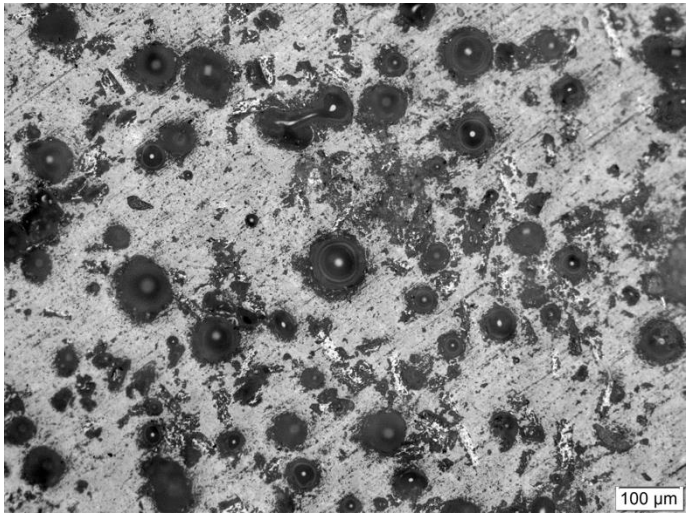
Learning curve showing evolution of training and validation loss

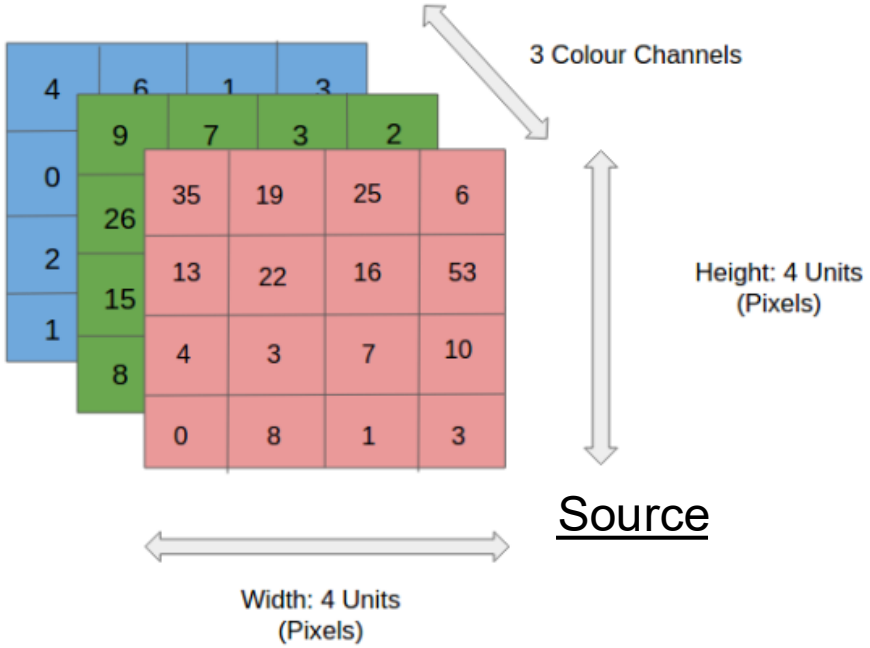


Parity plot comparing model predictions to ground truth bandgaps

- Open the Jupyter notebook again: **https://colab.research.google.com/github/mannodiarun/IMRC-2025-ML-Tutorial/blob/main/ml-tutorial.ipynb**
- Shortened url: **https://tinyurl.com/3nn8fruc**
- This time we will run the neural network cells.

Convolutional neural networks

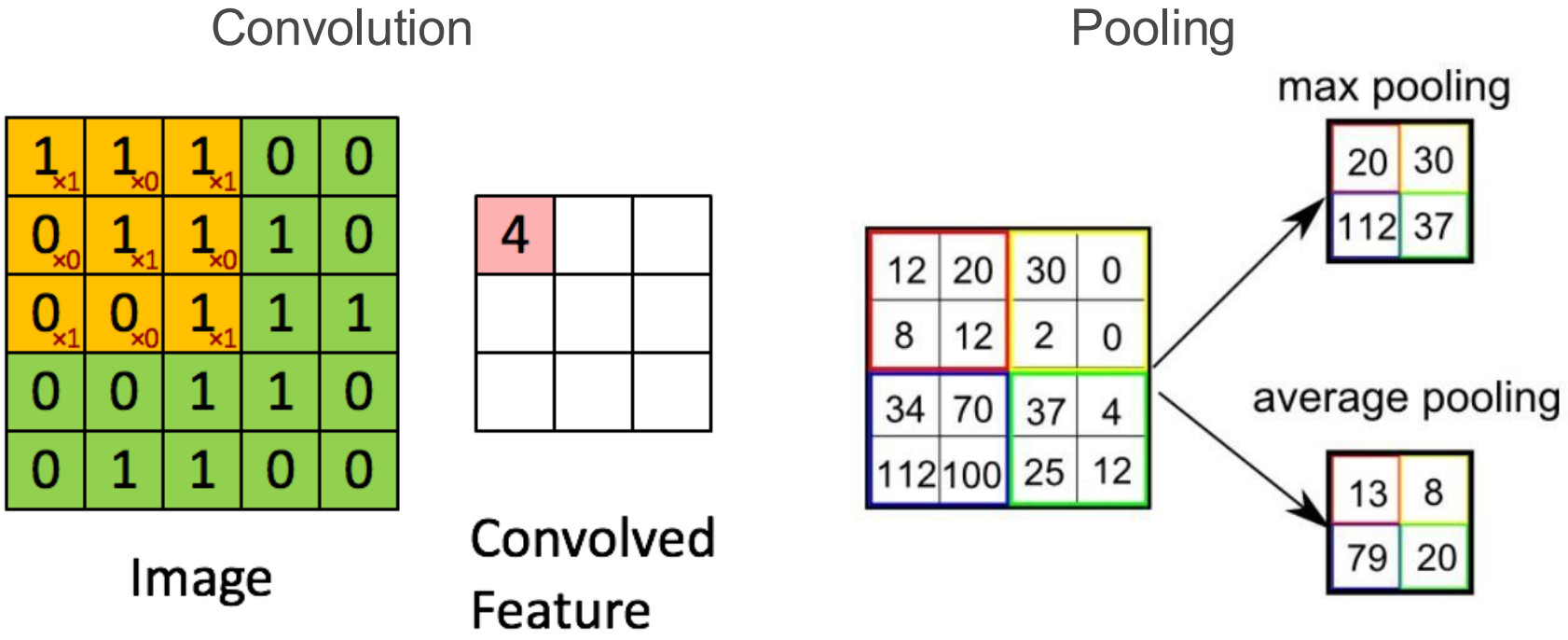




- Image based learning uses convolutional neural networks (CNNs)
- Image = (height, width, 3 channels)
- Operations in a CNN: Convolution, Pooling

Credit: Dr. Saaketh Desai, Sandia National Lab
https://github.com/mannodiarun/mrs_spring_tutorial/tree/nn_models

Convolutional neural networks



- Filter applied on each layer to get next layer
- Filter weights are learned
- Pooling combines features across multiple pixels

Source

Objective function

$$Objective = \frac{1}{N_{samples}} \sum_{i=1}^{N_{samples}} (\hat{y}_i - y_i)^2$$

ground truth

$$Objective = \frac{1}{N_{samples}} \sum_{i=1}^{N_{samples}} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

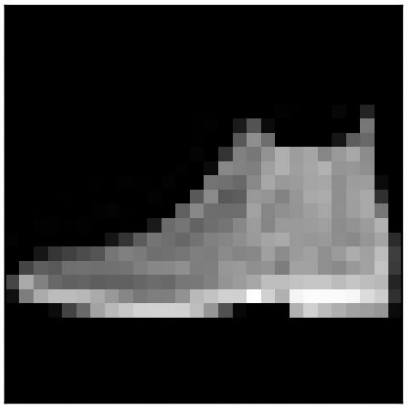
ground truth

Categorical cross entropy loss function

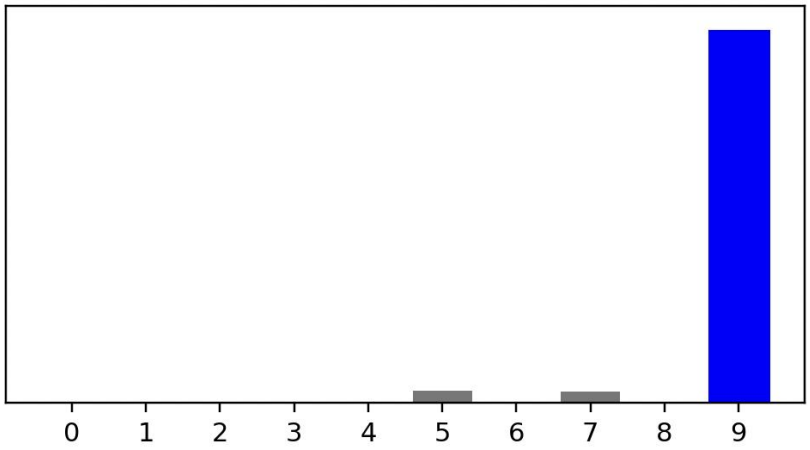
$$w_1 = w_1 - \alpha \frac{\partial(Objective)}{\partial w_1}$$

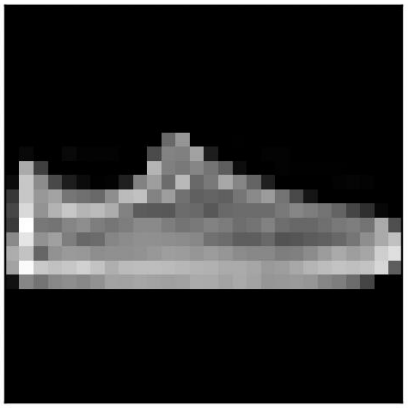
Backpropagation

Visualizing the learning

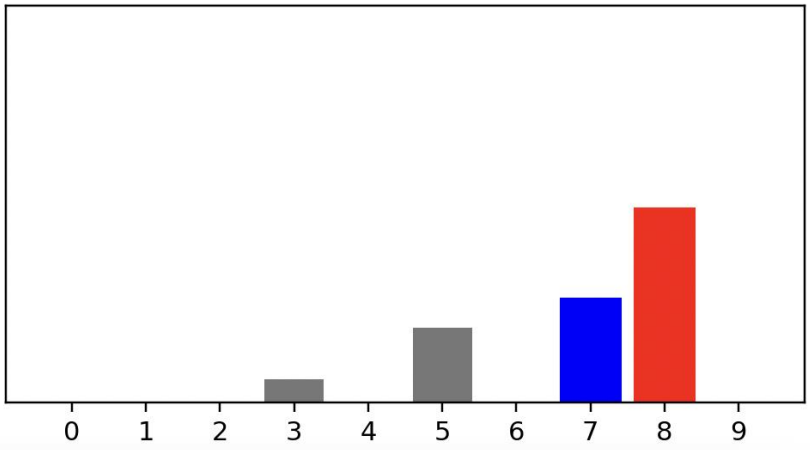


Ankle boot 94% (Ankle boot)





Bag 49% (Sneaker)



- Time permitting, we will run the CNN notebook (or please run on your own time): **https://colab.research.google.com/github/mannodiarun/mrs_spring_tutorial/blob/nn_models/Neural%20Network%20Classification.ipynb**

- Shortened url: **https://tinyurl.com/4f6mwz58**