Lecture 1:

Introduction to Machine Learning and Optimization

Week of January 17, 2023

University California, Berkeley

Machine Learning Algorithms

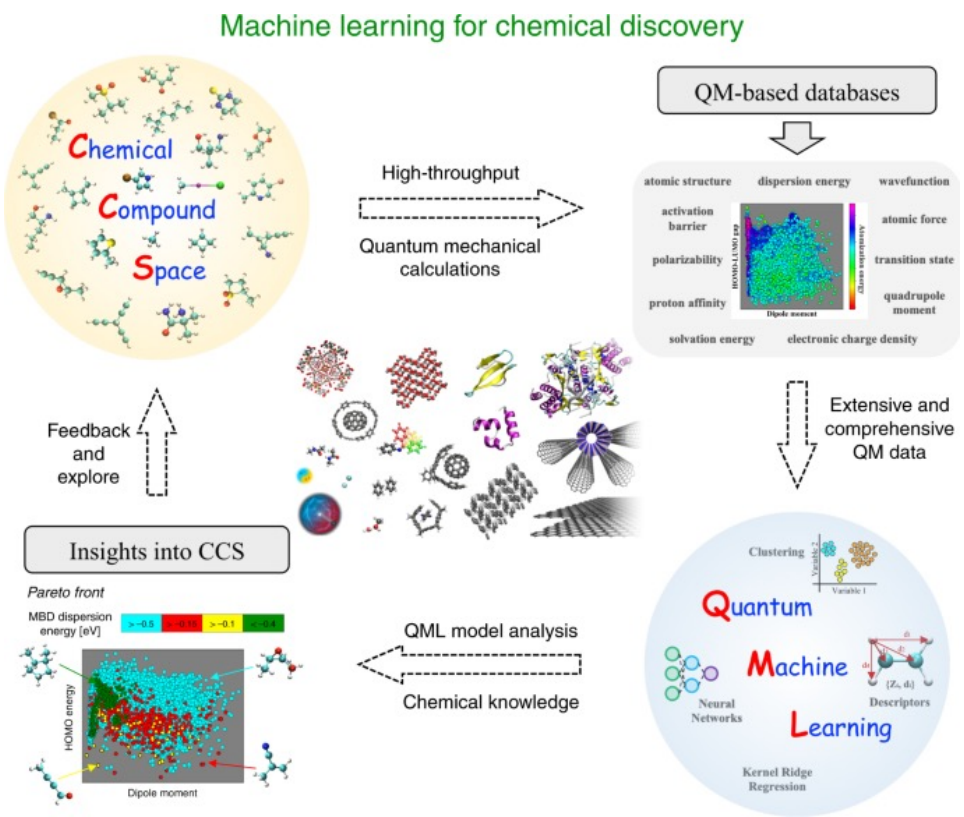
MSSE 277B, 3 Units

Spring 2023

Prof. Teresa Head-Gordon

Departments of Chemistry, Bioengineering, Chemical and Biomolecular Engineering

Machine Learning and Optimization for Molecular Problems



Course Description: An introduction to mathematical optimization and statistics and "non-algorithmic" computation using machine learning. Machine learning prerequisites are introduced including local and global optimization, various statistical and clustering models, and early meta-heuristic methods such as genetic algorithms and artificial neural networks.

Building on this foundation, current machine learning techniques are covered including Deep Learning networks, Convolutional neural networks, Recurrent and long short term memory (LSTM) networks, Generative models and Graph Neural Networks.

Various case studies in applying machine learning methods as classification and regression tasks in different areas of chemistry are covered. A finals project defined by research interest will be offered.

Course Objectives and Desired Outcomes: : (1) To introduce the basics of optimization and statistical modeling techniques relevant to MSSE students; (2) To build on optimization, statistical modeling, and data science to the field of machine learning techniques; (3) to utilize these concepts on problems relevant to the molecular sciences.Students will be able to understand the landscape and connections between numerical optimization, stand-alone statistical models, and machine learning techniques, and its relevance for chemical problems.

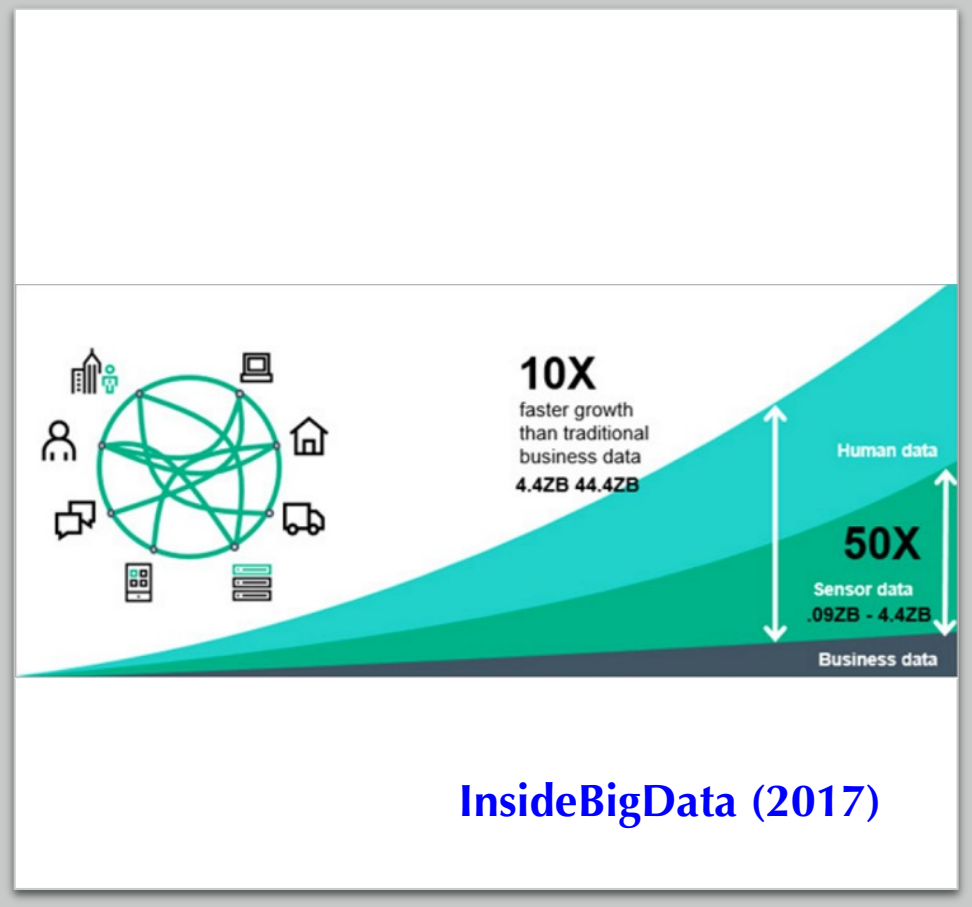
The Golden Age of Data

In 2016, 90% of the world's data has been created in the preceding two years. It has been true for the last 30 years that every two years, we produce 10 times as much data. (Hillstrom, 2005)

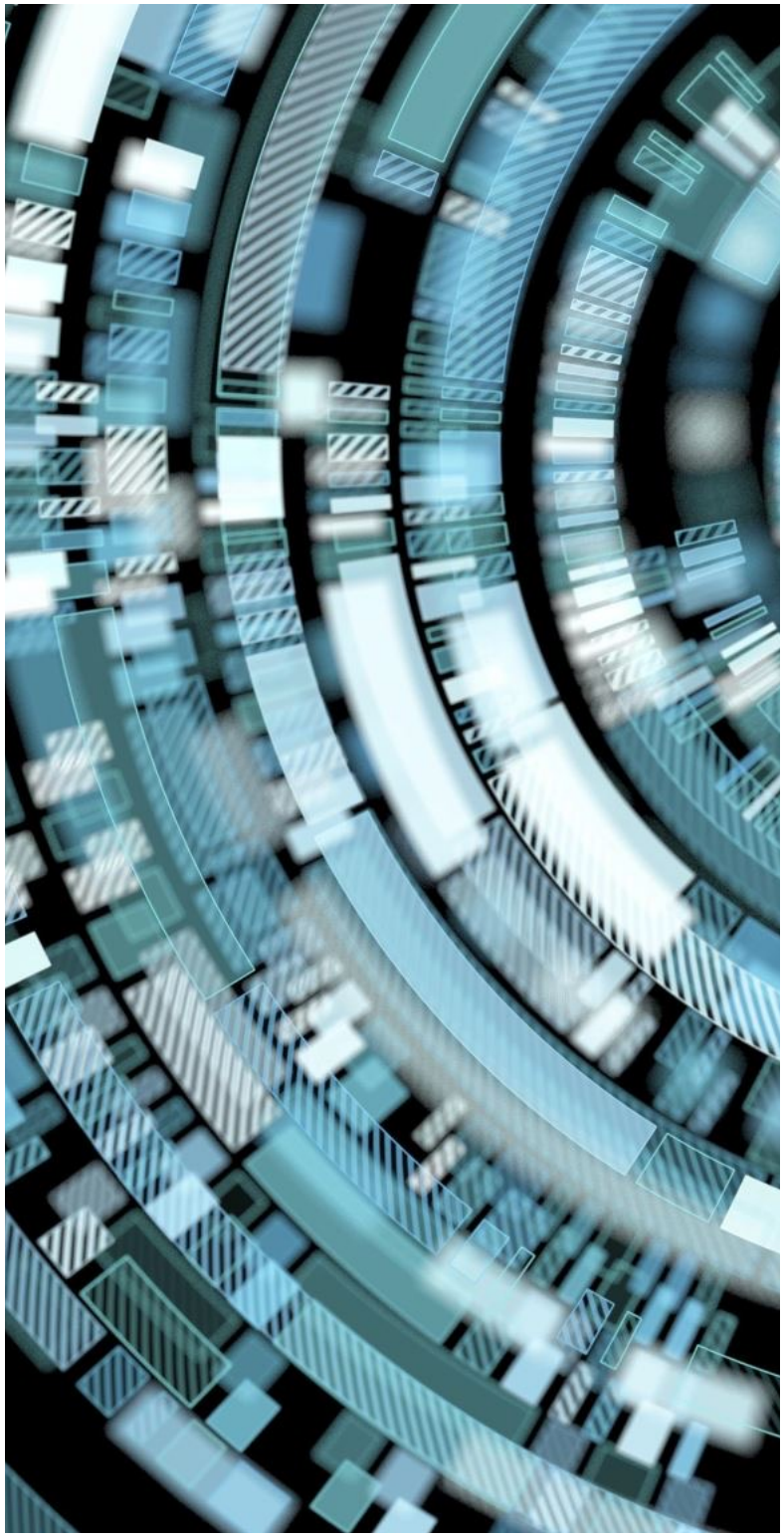
digital data storage requirements hit the "Zetta" prefix, with only one prefix, the "Yotta," left available

Rizatti, EETimes, 2016

Multiplying Factor	SI Prefix	Scientific Notation	Name
1 000 000 000 000 000 000 000 000	Yotta (Y)	10^{24}	1 septillion
1 000 000 000 000 000 000 000	Zetta (Z)	10^{21}	1 sextillion
1 000 000 000 000 000 000	Exa (E)	10^{18}	1 quintillion
1 000 000 000 000 000	Peta (P)	10^{15}	1 quadrillion
1 000 000 000 000	Tera (T)	10^{12}	1 trillion
1 000 000 000	Giga (G)	10^{9}	1 billion
1 000 000	Mega (M)	10^{6}	1 million
1 000	kilo (k)	10^{3}	1 thousand
0.001	milli (m)	10^{-3}	1 thousandth
0.000 001	micro (u)	10^{-6}	1 millionth
0.000 000 001	nano (n)	10^{-9}	1 billionth
0.000 000 000 001	pico (p)	10^{-12}	1 trillionth
0.000 000 000 000 001	femto (f)	10^{-15}	1 quadrillionth
0.000 000 000 000 000 001	atto (a)	10^{-18}	1 quintillionth
0.000 000 000 000 000 000 001	zepto (z)	10^{-21}	1 sextillionth
0.000 000 000 000 000 000 000 001	yocto (y)	10^{-24}	1 septillionth



InsideBigData (2017)



Mathematical Frameworks for Computational Chemistry and Data

Most machine learning (ML) problems reduce to optimization problems. Consider the machine learning analyst in action solving a problem for some set of data.

- The modeler formulates the problem by selecting an appropriate model and massages the data into an amenable format
- The model is typically trained by solving a core optimization problem that optimizes the variables or parameters of the model with respect to the selected loss function and possibly some regularization function.
- In the process of model selection and validation, the core optimization problem may be solved many times.

Operations Research and Machine Learning

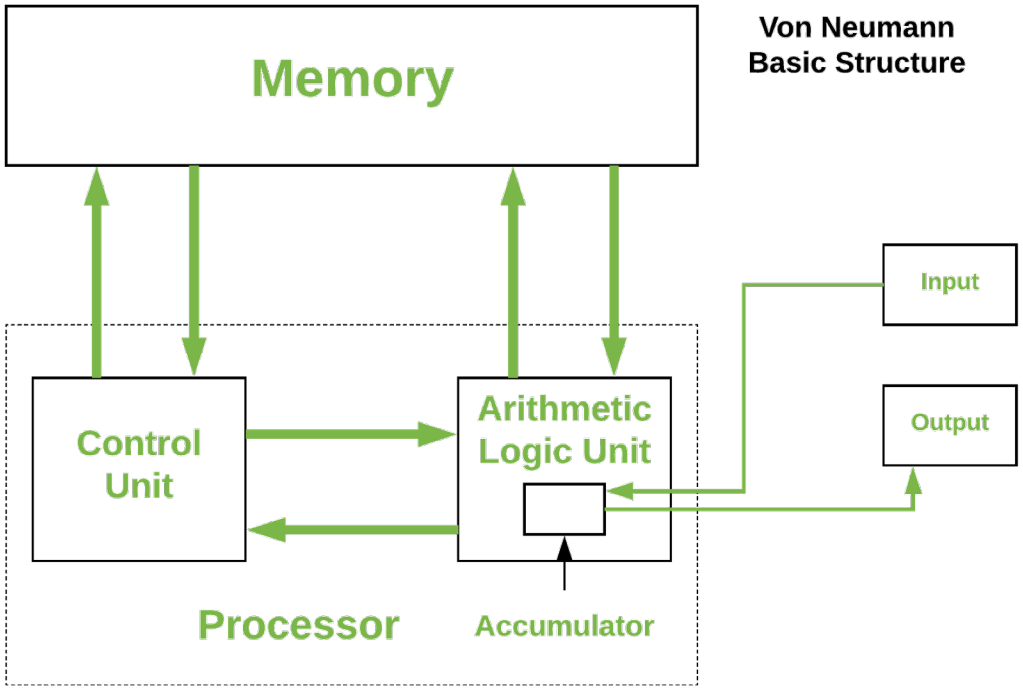
- Operations research (OR) is concerned with modeling a system. Mathematical programming is concerned with analyzing and solving the model. Both OR and ML analysts address real world problems by formulating a model, deriving the core optimization problem, and using mathematical programming to solve it.

- An OR analyst must trade off tractability – "the degree to which the model admits convenient analysis" and validity – "the degree to which inferences drawn from the model hold for real systems".

- So at a high level the OR and ML analysts face the same validity and tractability dilemmas and it is not surprising that both can exploit the same optimization toolbox.

Operations Research and Machine Learning

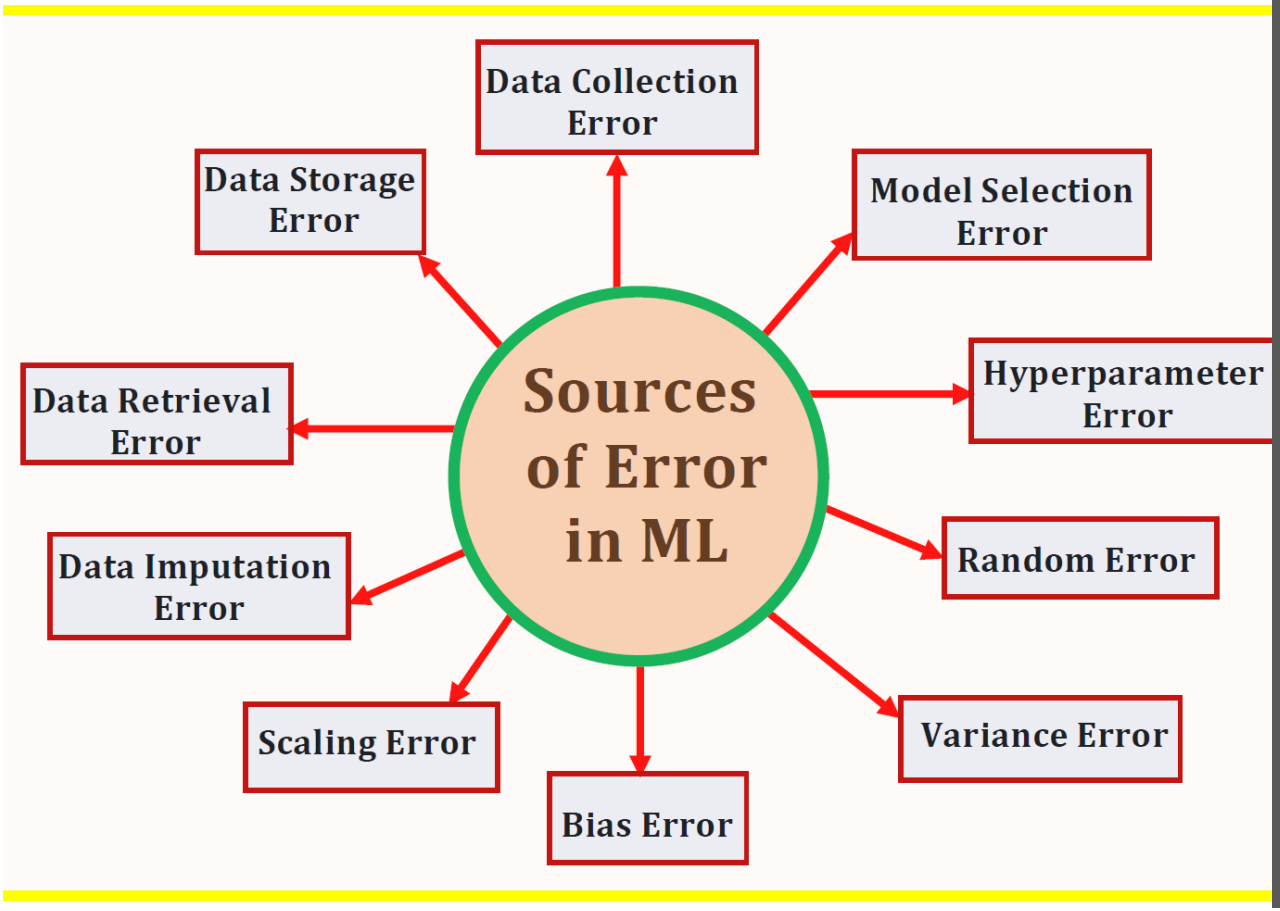
For OR, the problem is well-formulated in terms of a "von Neumann" machine, has well-defined metrics for success include solving a problem exactly, or using well-defined approximation, have better speed, better scalability to larger problems, and measures of violation of constraints.



In ML, the only quality solution is one that has minimized generalization errors and possibly computation times. For a practical ML problem, the ML analyst might pick one or more families of learning models and an appropriate training loss/regularization function, and then search for an appropriate model that performs well according to some estimate of the generalization error from the given training data.

- This search typically involves some combination of data preprocessing, mathematical optimization, and heuristics.

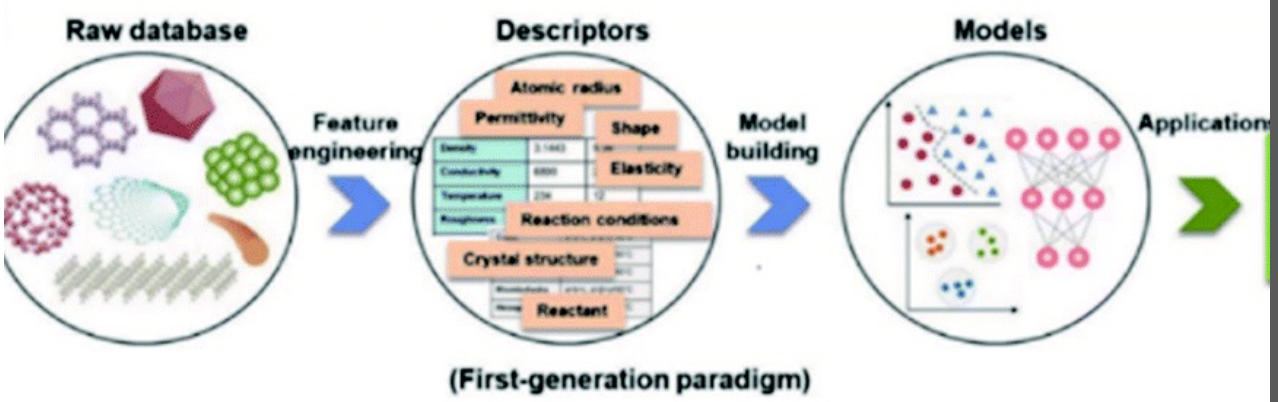
Introduction to Machine Learning

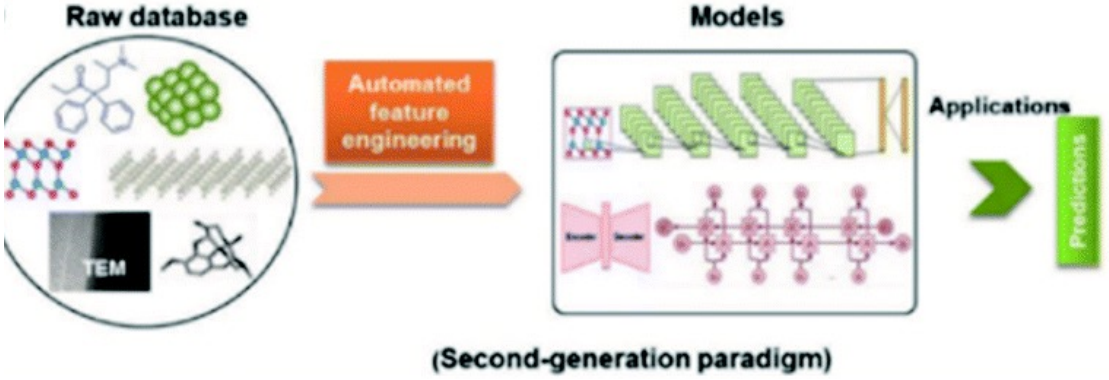


The quality of the resulting ML inductive functions involve three general sources of errors.

- The underlying true function and error distribution are unknown, thus any choice of data representation, model family and loss functions may not be suitable for the problem.
- Only a finite amount of (possibly noisy) data is available or is corrupted
- The third source of error stems from the difficulty of the search problem for parameters. Reducing the problem to a convex optimization by appropriate choices of loss and constraints or relaxations can greatly help the search problem.

Introduction to Machine Learning

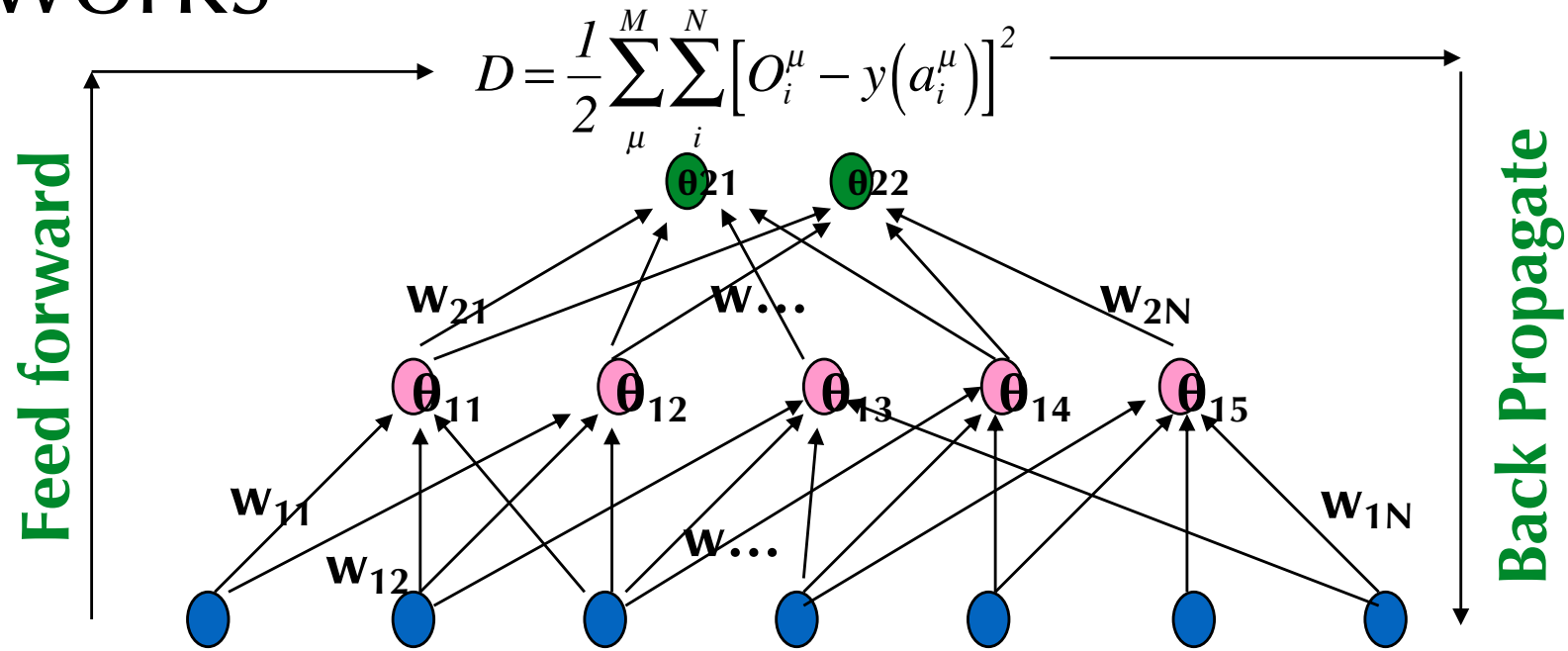




Desirable properties of an optimization algorithm from the ML perspective are

- good generalization,
- scalability to large problems,
- good performance in practice in terms of execution times and memory requirements,
- simple and easy implementation of algorithm,
- exploitation of problem structure
- fast convergence to an *approximate* solution of model
- robustness and numerical stability for class of machine learning models attempted
- theoretically known convergence and complexity

Supervised Learning with Neural Networks

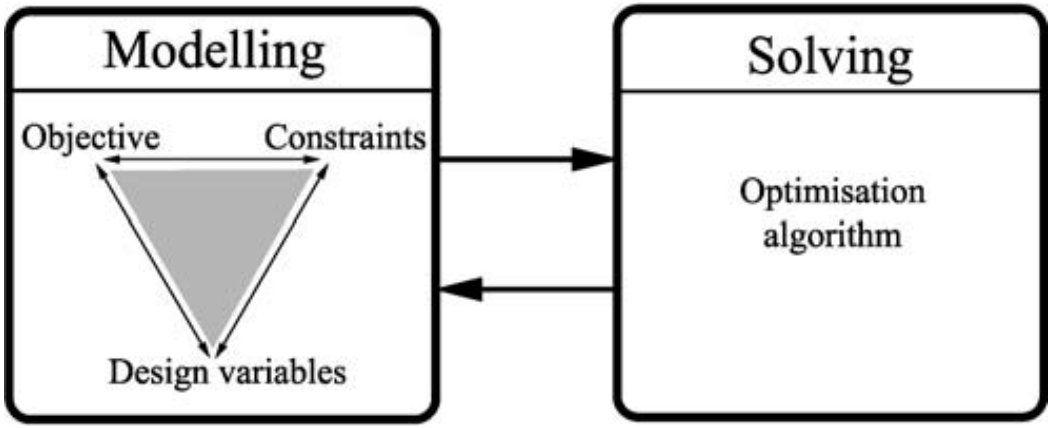


Local optimization methods are used in supervised learning to train network variables (weights, biases) to best reproduce the training data and to ultimately generalize to test or new data to make new predictions.

Adjusting weights to optimize learning via mathematical programming

$$\delta w_{ij} = -\varepsilon\frac{\partial D}{\partial w_{ij}} = \varepsilon\sum_{\mu}\left[O_i^{\mu} - y\left(a_i^{\mu}\right)\right]\frac{dy}{da_i^{\mu}}\frac{\partial a_i^{\mu}}{\partial w_{ij}}$$

Mathematical Optimization



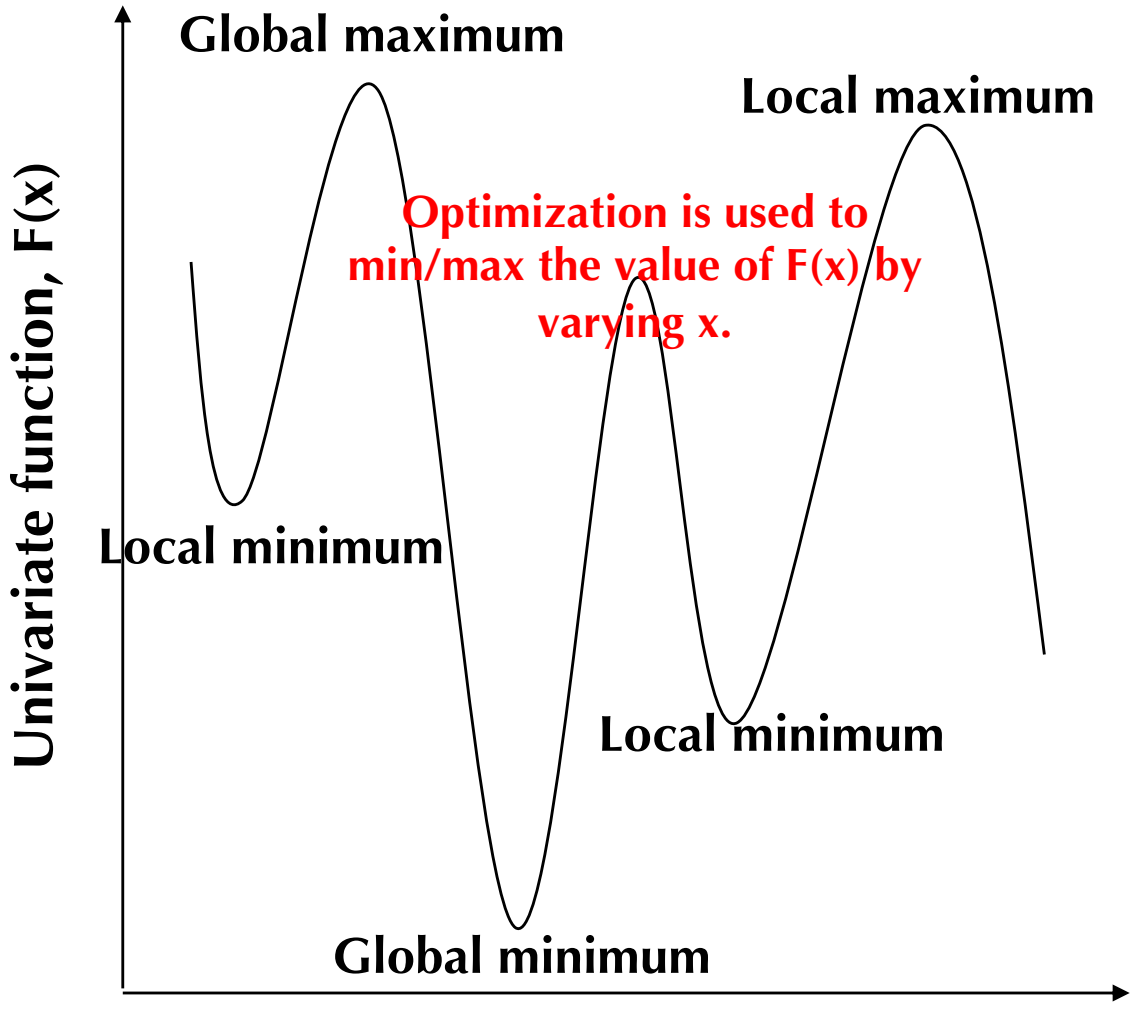
There are three basic elements of any optimization problem -

- Variables: These are the free parameters which the algorithm can tune

- Constraints: These are the boundaries within which the parameters (or some combination thereof) must fall

- Objective function: This is the set of goals towards which the algorithm drives the solution. For machine learning, often this amount to minimizing some error measure or maximizing some utility function.

Local Optimization: Univariate Function



Stationary point: at a maximum or a minimum

$$\frac{df(x)}{dx} = 0$$

At a minimum, curvature is positive

$$\frac{d^2 f(x)}{dx^2} > 0$$

At a maximum, curvature is negative

$$\frac{d^2 f(x)}{dx^2} < 0$$

Multivariate Optimization

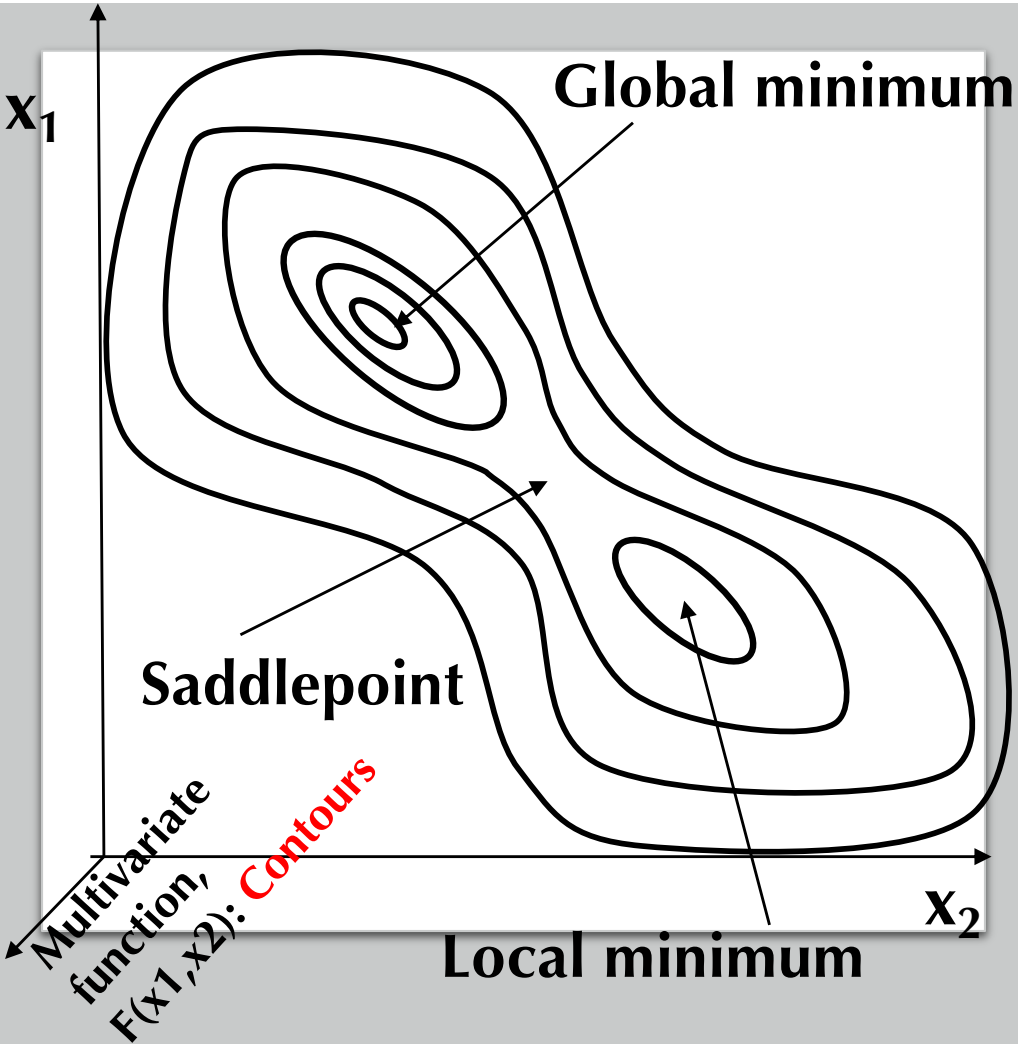
Optimization is used to min/max $F(x_n)$ by varying $\{x_1, x_2 \ldots x_n\}$.

Eigenvalues of Hessian matrix are:

all positive at a minimum

all negative at a maximum

all positive except one negative for saddle point



Hessian matrix, H

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Quadratic Approximation

Univariate

$$f(x_0+\Delta x)=f(x_0)+\left.\frac{df}{dx}\right|_{x=x_0}\Delta x+\frac{1}{2}\left.\frac{d^2f}{dx^2}\right|_{x=x_0}\Delta x^2$$

Multivariate

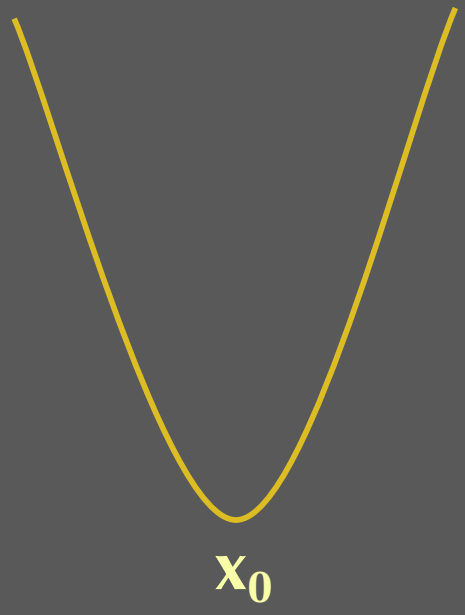
$$f(\vec{x}_0+\Delta\vec{x})=f(\vec{x}_0)+\Delta\vec{x}^T\vec{\nabla}f\Big|_{\vec{x}=\vec{x}_0}+\frac{1}{2}\Delta\vec{x}^T\,\underline{\underline{H}}\Big|_{\vec{x}=\vec{x}_0}\Delta\vec{x}$$

At a minimum, 1st derivative or ∇f is zero.

$$f(\vec{x}_0+\Delta\vec{x})=f(\vec{x}_0)+\Delta\vec{x}^T\vec{\nabla}f\Big|_{\vec{x}=\vec{x}_0}+\frac{1}{2}\Delta\vec{x}^T\,\underline{\underline{H}}\Big|_{\vec{x}=\vec{x}_0}\Delta\vec{x}$$

$\vec{\nabla}f\Big|_{\vec{x}=\vec{x}_0}$ → **0**

Therefore in the vicinity of a minimum for any function (quadratic or non-quadratic), the local surface appears quadratic.



Determining a local minimum with a method that uses Hessian will converge in a single step for a quadratic function

0th order method: uses function (energy) evaluations only

$$f(\vec{x}_0) \Rightarrow f(\vec{x})$$

Golden Section

1st order method: uses function and derivative evaluations

$$f(\vec{x}_0) \Rightarrow f(\vec{x}), \vec{\nabla} f(\vec{x})$$

Steepest Descents, Momentum SD (often used in ML), conjugate gradients (sometimes!)

2nd order method: uses function, derivative, Hessian evaluations

$$f(\vec{x}_0) \Rightarrow f(\vec{x}), \vec{\nabla} f(\vec{x}), \underline{\underline{H}}(\vec{x})$$

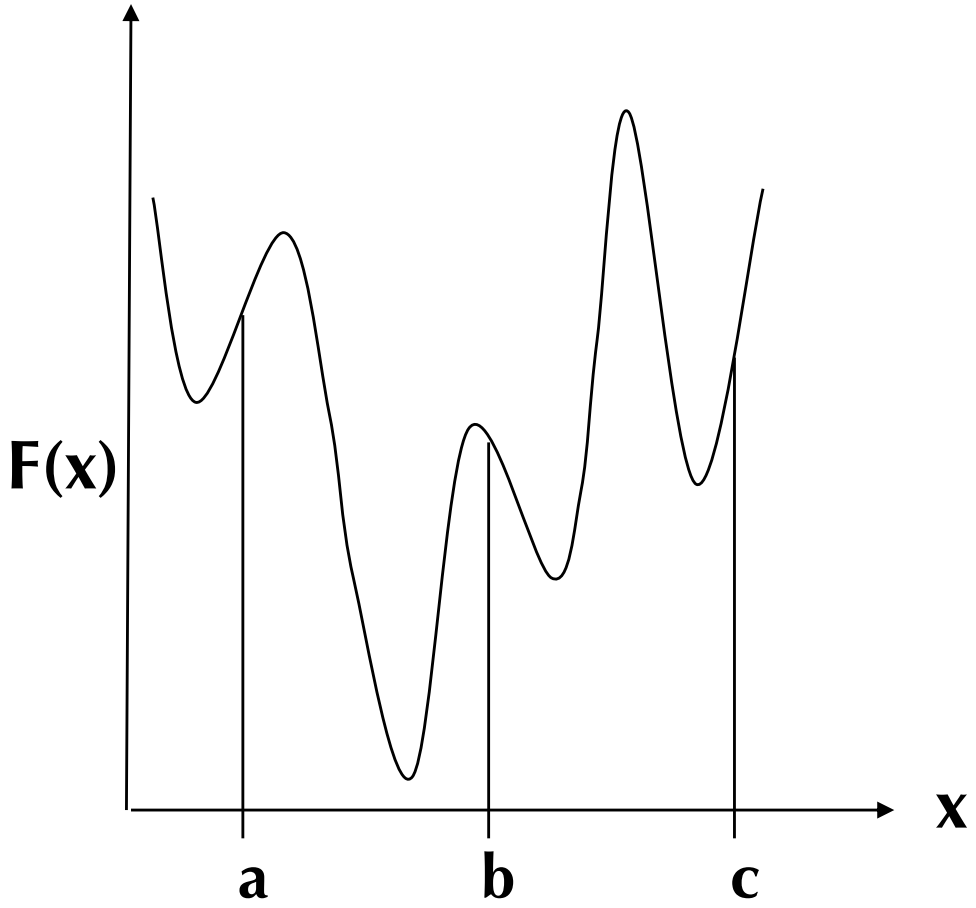
Newton Method (rarely!)

Quasi-2nd order method: uses function, derivative, approximate Hessian

$$f(\vec{x}_0) \Rightarrow f(\vec{x}), \vec{\nabla} f(\vec{x}), \underline{\underline{\approx H}}(\vec{x})$$

Davis, Fletcher, Powell (DFP)
Broyden, Fletcher Goldstein, Shanno (BFGS – sometimes!)

Classification Scheme: Local Optimization



0th Order Method: Bracketing a Minimum

Most algorithms for minimizing functions of n variables work by doing a large sequence of 1D optimizations. Hence bracketing a minimum is often advisable to better solve a 1D optimization

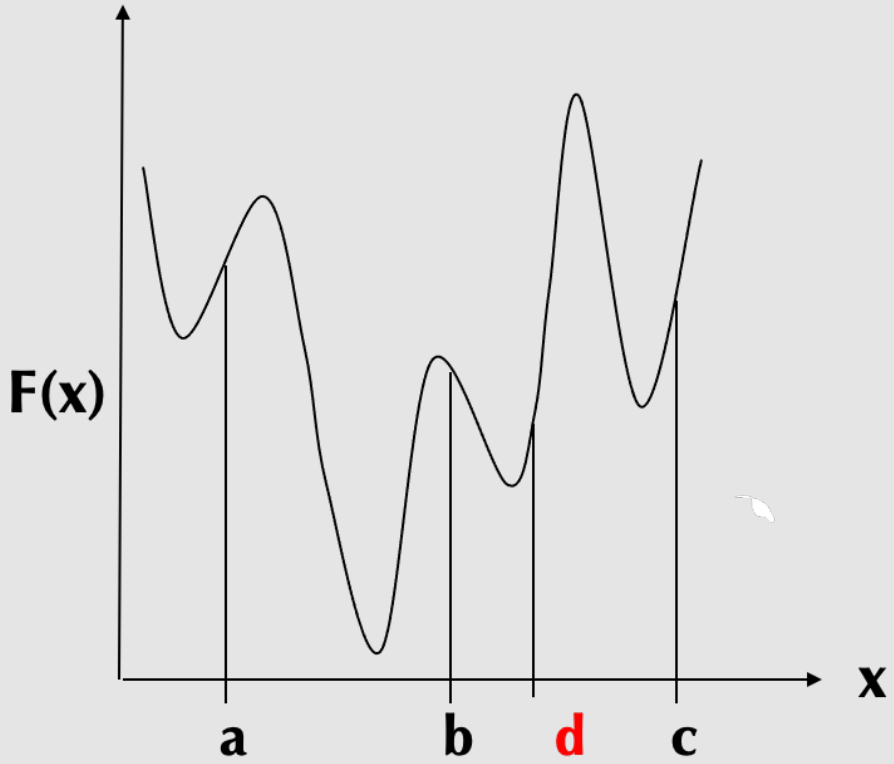
Imagine an expensive function whose minima you would like to know, but can only afford some function calls. For x= a < b < c, if

$$f(b) < f(a)$$

$$f(b) < f(c)$$

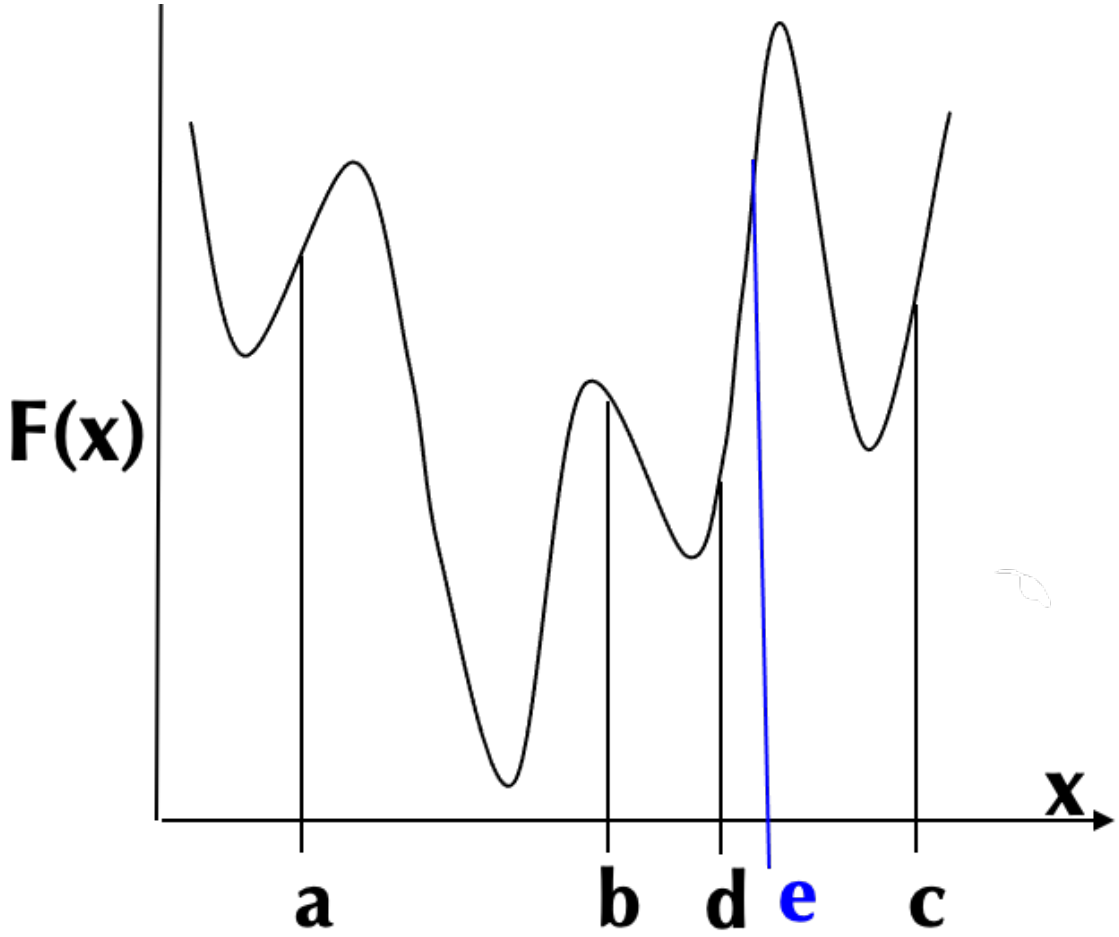
Then there is at least one minimum in the range defined by the triplet [a,b,c]

0th Order Method: Bracketing a Minimum



What we would like to do in bracketing is to reduced the search space relative to original interval [a,c] as we accumulate new information based on new input variable x, f(x) in this region. Let's illustrate!

- Evaluate new point d such that [a,b,d,c]. The new point d substitutes one of the original points in bracketing triplet [a,b,c]
- If f(d)<f(b) then minimum occurs in new triplet [b,d,c]
- If f(d)>f(b) then minimum occurs in new triplet [a,b,d]



0th Order Method: Bracketing a Minimum

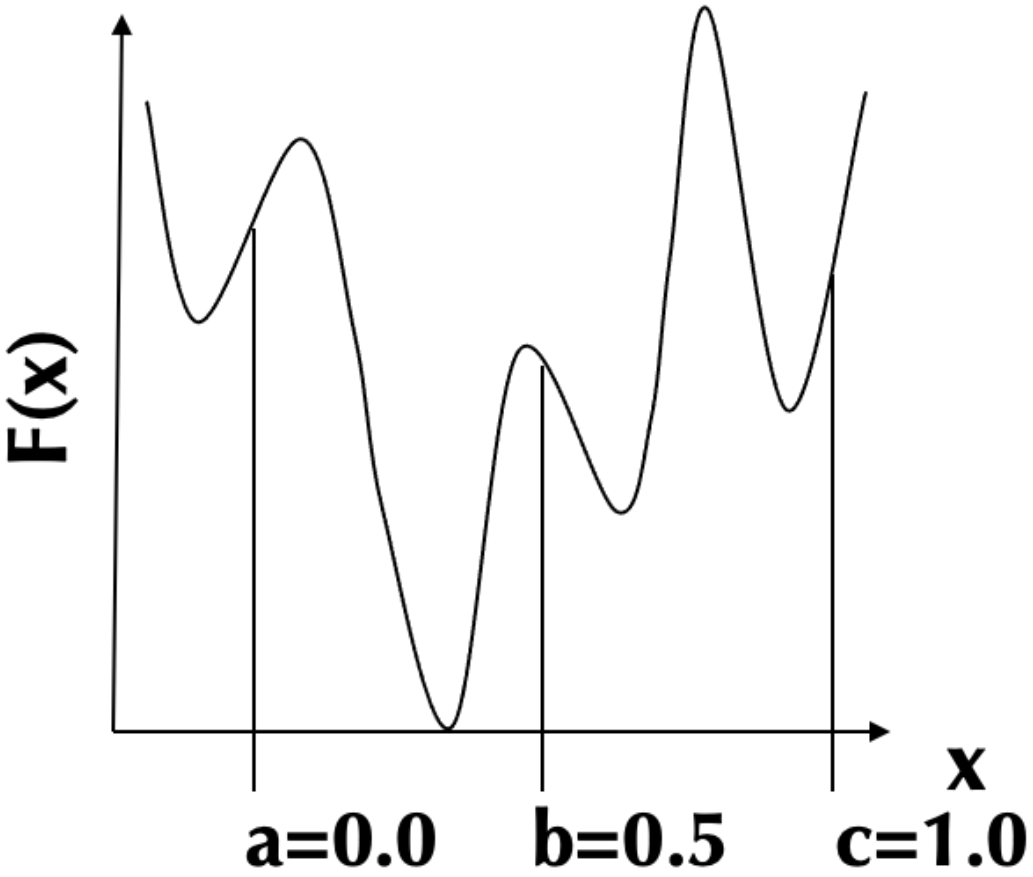
That 1st step has narrowed our search to smaller interval [b,c].

- 2nd step is to evaluate new point e in this reduced interval such that [b,d,e,c]. The new point e substitutes one of the original points in bracketing triplet

- If f(e)<f(d) then minimum occurs in new triplet [d,e,c]

- If f(e)>f(d) then minimum occurs in new triplet [b,d,e]

Continue searching in each reduced interval through n steps until converged to local minimum based on tolerance $|f(x_n)-f(x_{n-1})|<tol$



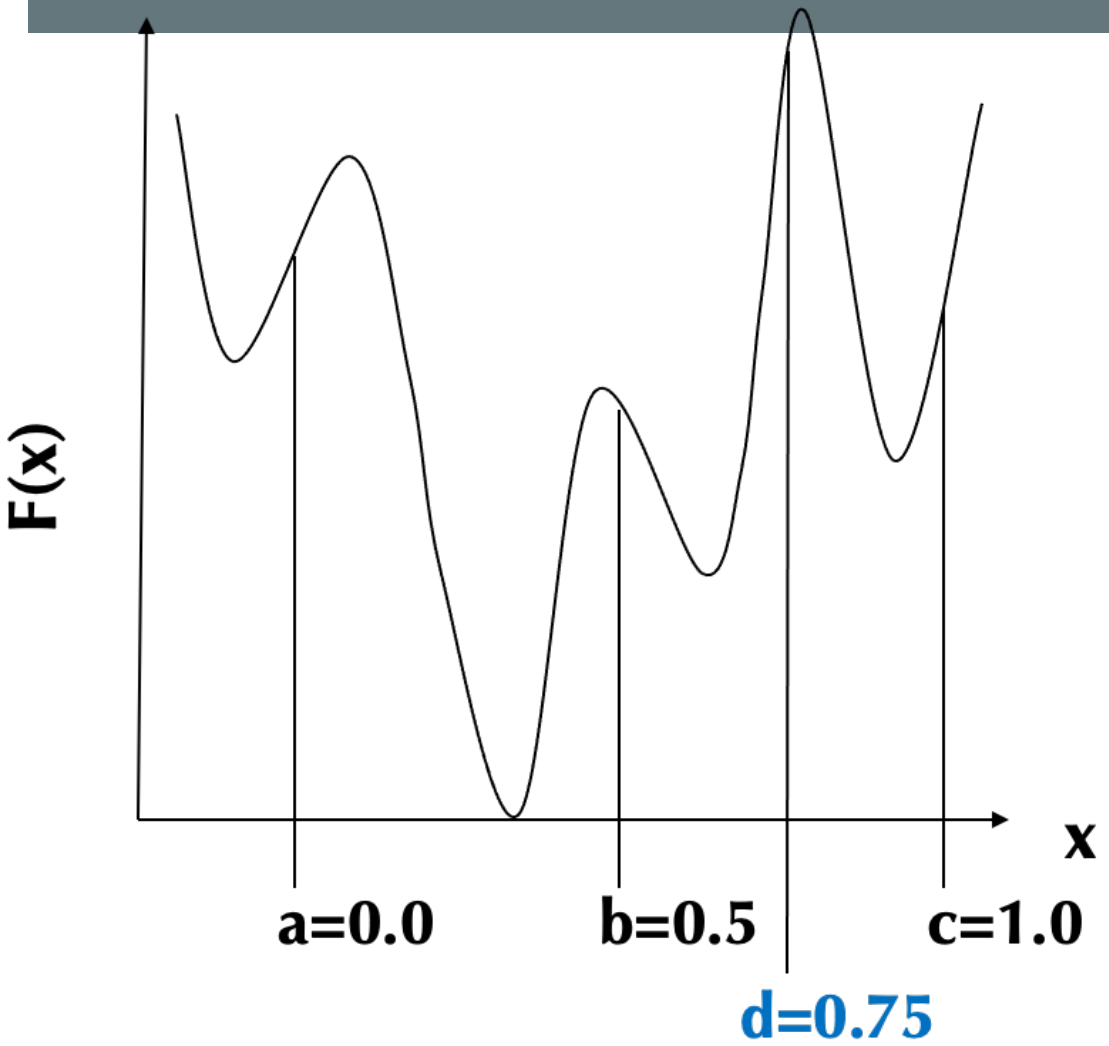
Optimizing Bracketing

How can we more systematically and optimally pick the next position for a function evaluation in the current interval?

Consider bisection (and our first HW assignment to familiarize ourselves with SciPy): divide search interval by equal sectioning and place new point in one of the bisected intervals [a,b] or [b,c]

We don't know of course which is the better interval – so bisect one of them!

BiSection Method: 0th Order Method



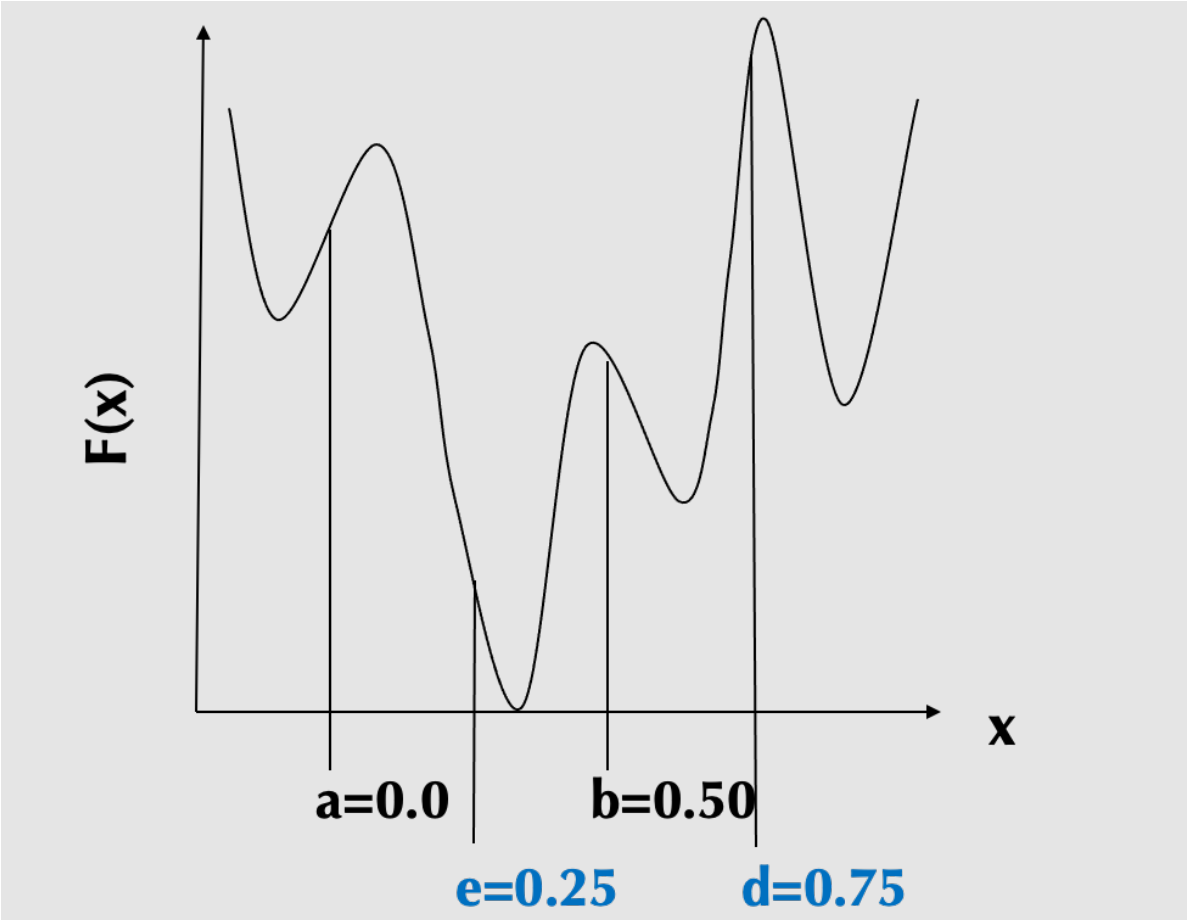
Step 1: we place point d in the right bisector of [b,c]

The new point d substitutes one of the original points in bracketing triplet [a,b,d,c]

If f(d)<f(b) then minimum occurs in new triplet [b,d,c] (reduce search by 1/2)

If f(d)>f(b) then minimum occurs in new triplet [a,b,d] (reduce search by 1/4)

We have bracketed the minimum in [a,b,d], but reducing search space by only ¼. Our search space is 0.75



BiSection Method: 0th Order Method

Step 2: reduce the size of the [a,d] interval;

Homework 1:

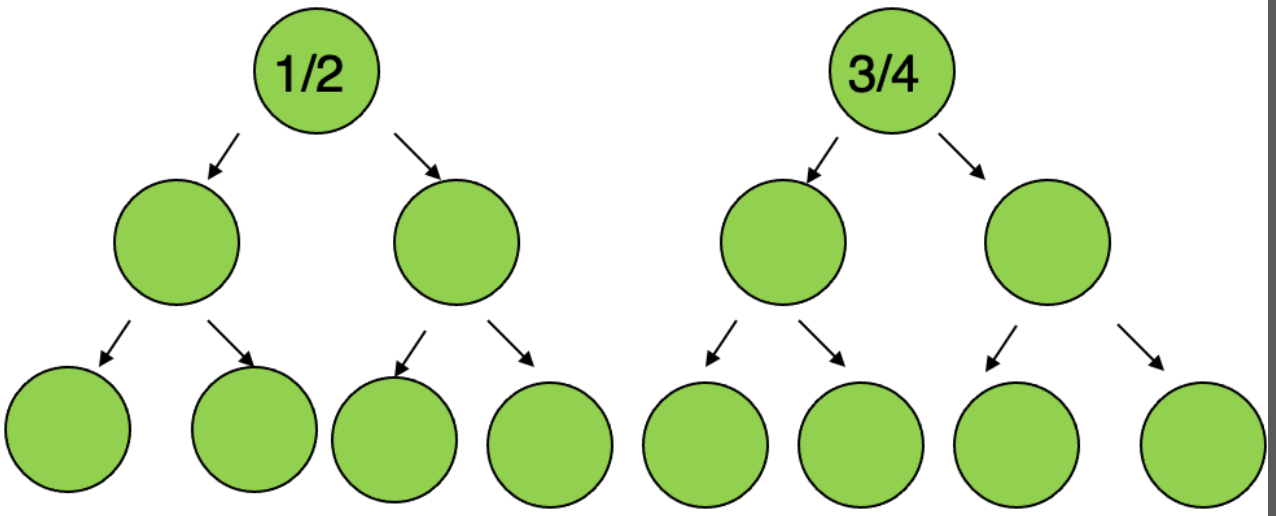
(a) place point e at the bisector of larger interval [a,b]. Why is this better than [b,d]?

(b) What is the new interval and how much search space is reduced?

BiSection Method: 0th Order Method

Step1: New size interval

<0.625> = P(0.5) x 1/2 + P(0.5) x 3/4



Homework 1:

fill in tree for size intervals for steps 2 and 3

What is average size of interval at steps 2 and 3?