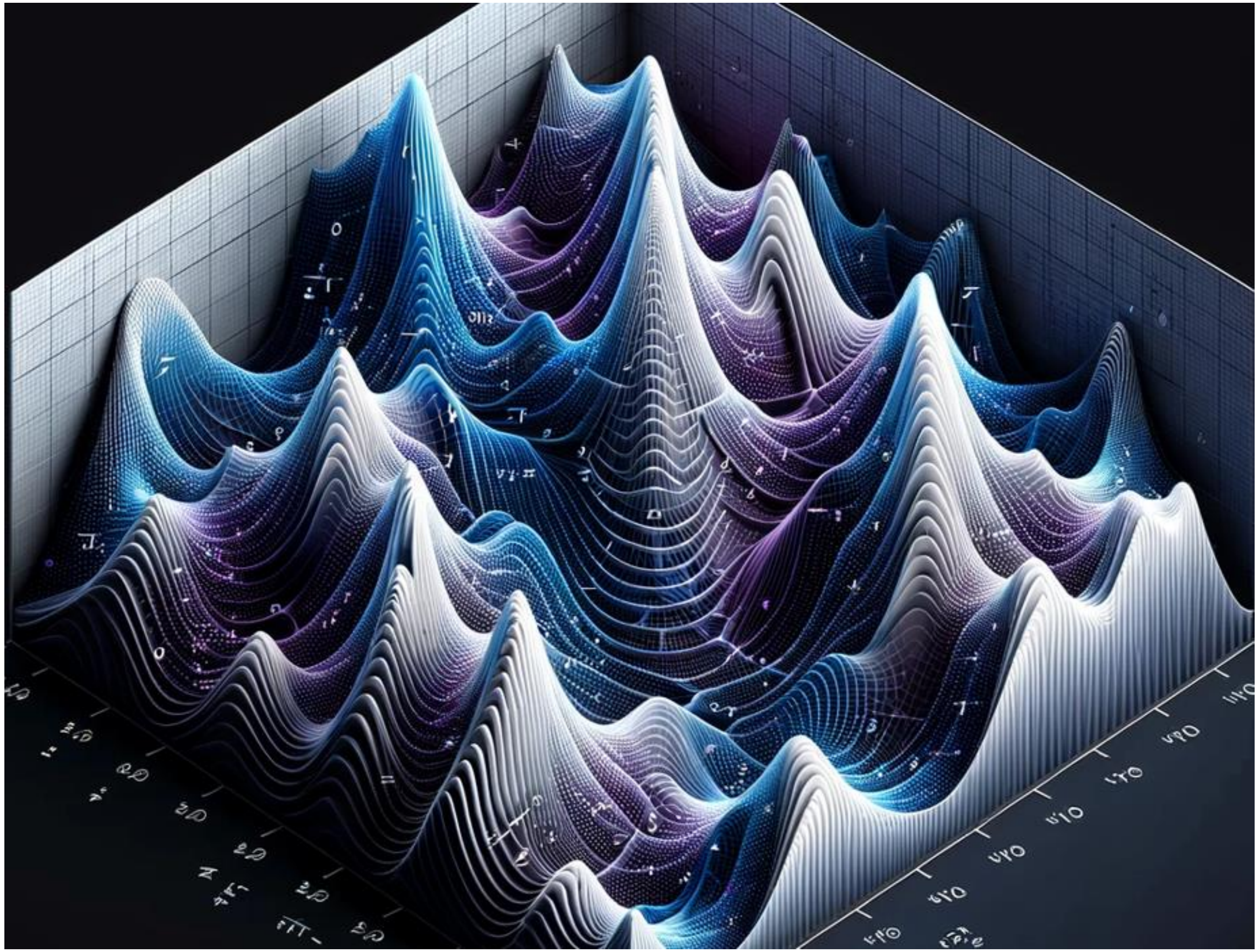


LEHIGH UNIVERSITY

NUMERICAL OPTIMIZATION

ISE – 364 / 464

DEPT. OF INDUSTRIAL & SYSTEMS ENGINEERING

GRIFFIN DEAN KENT

WHAT IS OPTIMIZATION?

What is Mathematical Optimization?

Mathematical optimization (or mathematical programming) is the selection of a best element, with regard to some criteria, from a set of available alternatives.

- This is typically done by either maximizing or minimizing a given function under some set of constraints.

Why the Need?

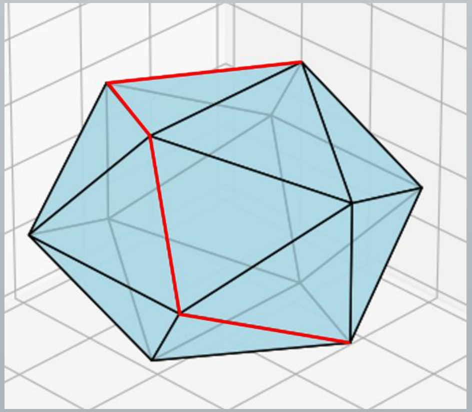
Optimization algorithms (a series rules for iteratively improving a decision or choice) are indispensable when dealing with problems that are too complex to solve analytically or do not have closed-form solutions. Some reasons why specialized optimization algorithms are needed can include: complex and highly nonlinear relationships that cannot be described by simple closed-form equations, high-dimensional spaces that are impossible to visualize and solve with standard methods, constraints that define if a solution is feasible or not, and non-trivial objective functions that could have many optima.

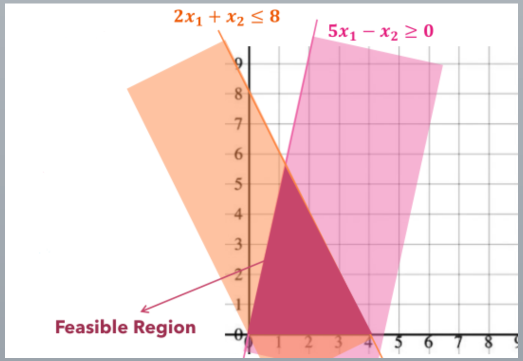
Areas of Application

Essentially, **any field** or industry that desires or requires making "optimal" decisions implement some form of mathematical optimization: **Machine Learning**, supply Chain Management, Engineering Design, Energy Management, Healthcare, Finance, Physics,

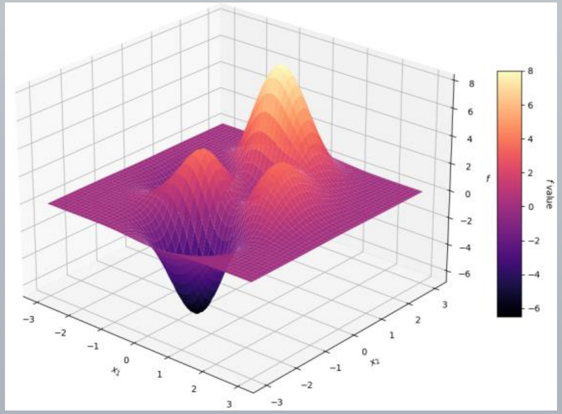
SOME OF THE TYPES OF NUMERICAL OPTIMIZATION

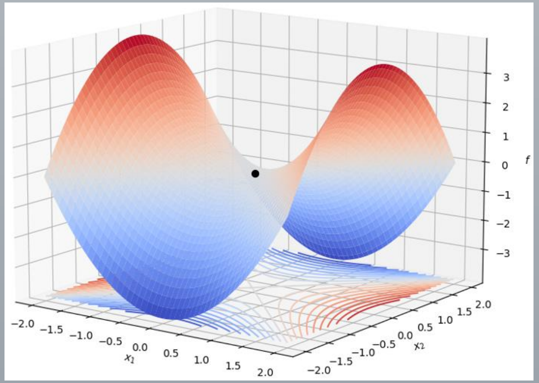
Linear Optimization



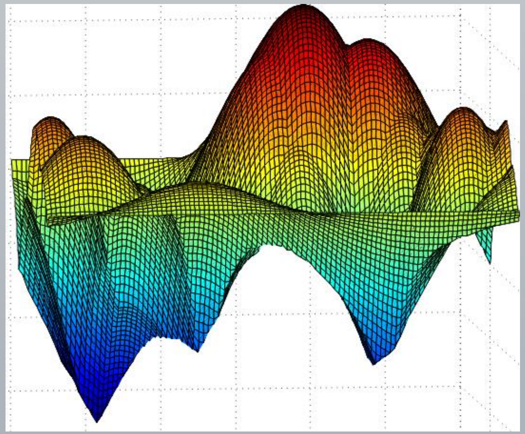


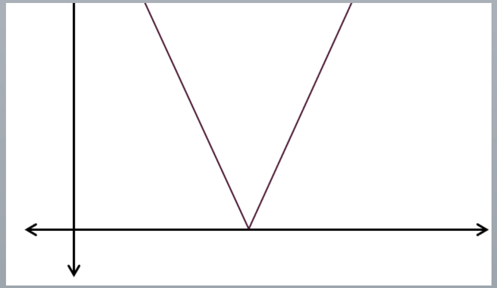
Nonlinear Optimization





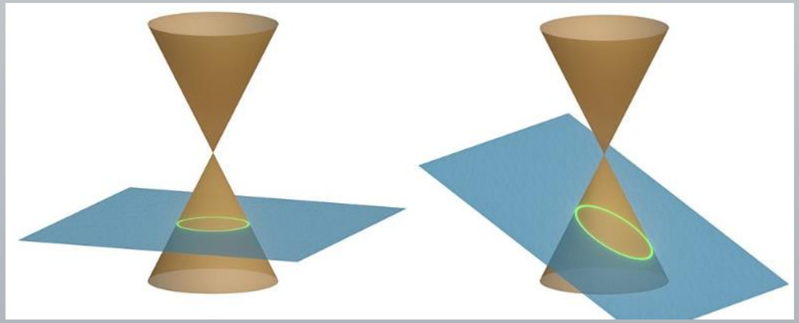
Non-smooth Optimization

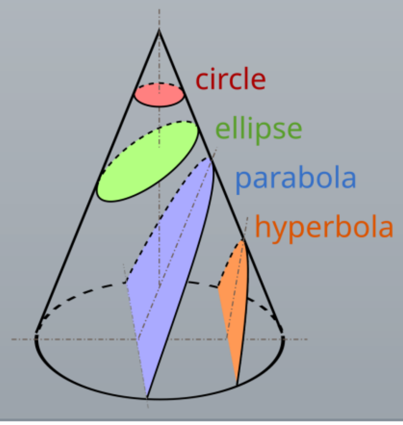




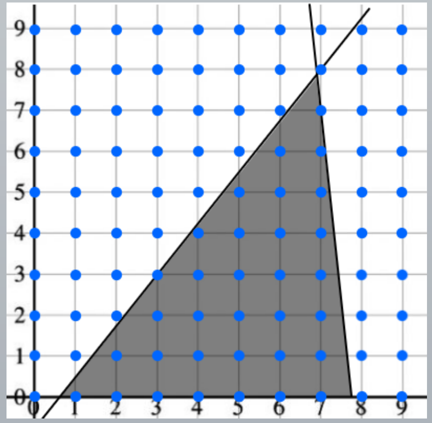
SOME OF THE TYPES OF NUMERICAL OPTIMIZATION

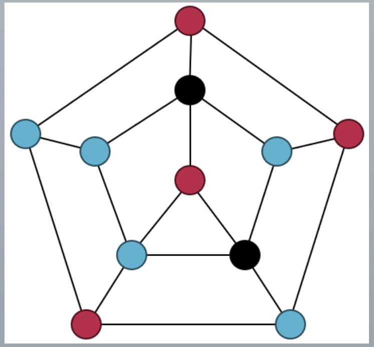
Conic Programming





Integer Programming





GENERAL NOTATION

In **mathematical optimization**, we are typically interested in minimizing an **objective function** $f : \mathcal{X} \to \mathbb{R}$ over the **decision variables** $x \in \mathcal{X} \subseteq \mathbb{R}^n$, where $\mathcal{X}$ is called the "**feasible region**". We write this as the following general optimization problem:

$$\min_{x \in \mathbb{R}^n} f(x) \qquad s.t. \qquad x \in \mathcal{X}.$$

Here, the "s.t." stands for "such that" and precedes the constraints that define the feasible region of values that the decision variables can take.

General Constrained Optimization

If the feasible region can be described by an explicit set of equations ("**equality constraints**" which we denote by the vector-valued function $E : \mathcal{X} \to \mathbb{R}^{|E|}$, where $|E|$ denotes the number of equality constraints) and inequalities ("**inequality constraints**" which we denote by the vector-valued function $I : \mathcal{X} \to \mathbb{R}^{|I|}$, where $|I|$ denotes the number of inequality constraints), then we can write the general **constrained** optimization problem:

$$\min_{x \in \mathbb{R}^n} f(x) \qquad s.t. \qquad E(x) = 0, \qquad I(x) \leq 0.$$

General Unconstrained optimization

If the feasible region is defined as $\mathcal{X} = \mathbb{R}^n$, then we can simply write the general **unconstrained** optimization problem:

$$\min_{x \in \mathbb{R}^n} f(x).$$

FUNDAMENTALS OF UNCONSTRAINED OPTIMIZATION

Global Minimizer

The vector $x^* \in \mathbb{R}^n$ is called a **global minimizer** if $f(x^*) \leq f(x), \forall x \in \mathbb{R}^n$.

Local Minimizer

The vector $\bar{x} \in \mathbb{R}^n$ is called a **local minimizer** if there exists a scalar $\varepsilon > 0$ such that $f(\bar{x}) \leq f(x)$, $\forall x \in \mathcal{N}(\bar{x}, \varepsilon)$. Here, $\mathcal{N}$ is the "ε-neighborhood" function defined by

$$\mathcal{N}(\bar{x}, \varepsilon) := \{x \in \mathbb{R}^n : \ \|\bar{x} - x\|_2 \leq \varepsilon\}.$$

Strict Local Minimizer

The vector $\bar{x} \in \mathbb{R}^n$ is called a **strict local minimizer** if there exists a scalar $\varepsilon > 0$ such that $f(\bar{x}) < f(x), \forall x \neq x$ such that $x \in \mathcal{N}(\bar{x}, \varepsilon)$.

Isolated local Minimizer

The vector $\bar{x} \in \mathbb{R}^n$ is called an **isolated local minimizer** if there exists a scalar $\varepsilon > 0$ such that $\bar{x}$ is the only local minimizer in $\mathcal{N}(\bar{x}, \varepsilon)$.

EXAMPLES OF MINIMIZERS

Local, Strict Isolated Local, and Global Minimizers

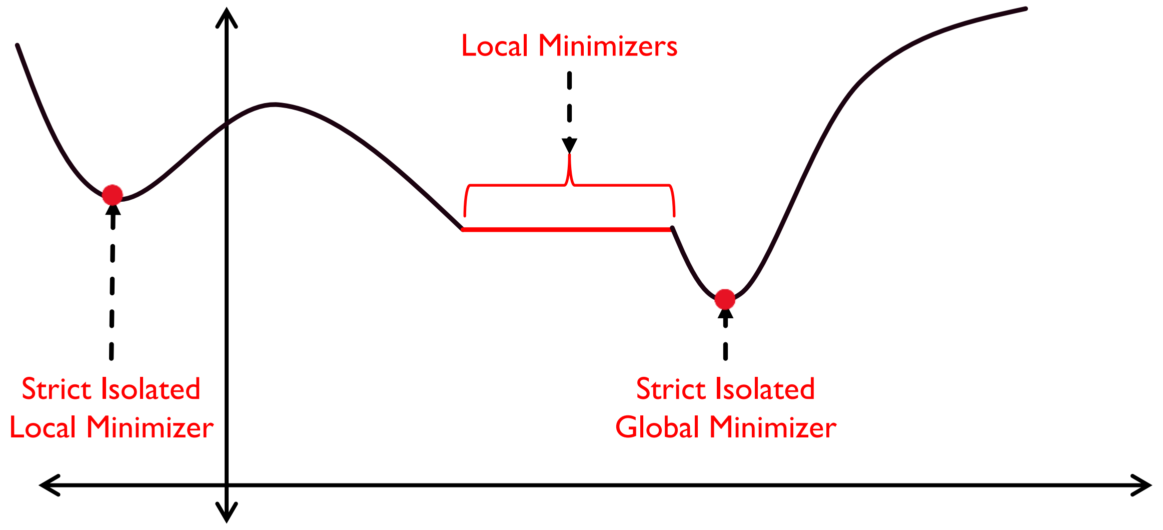
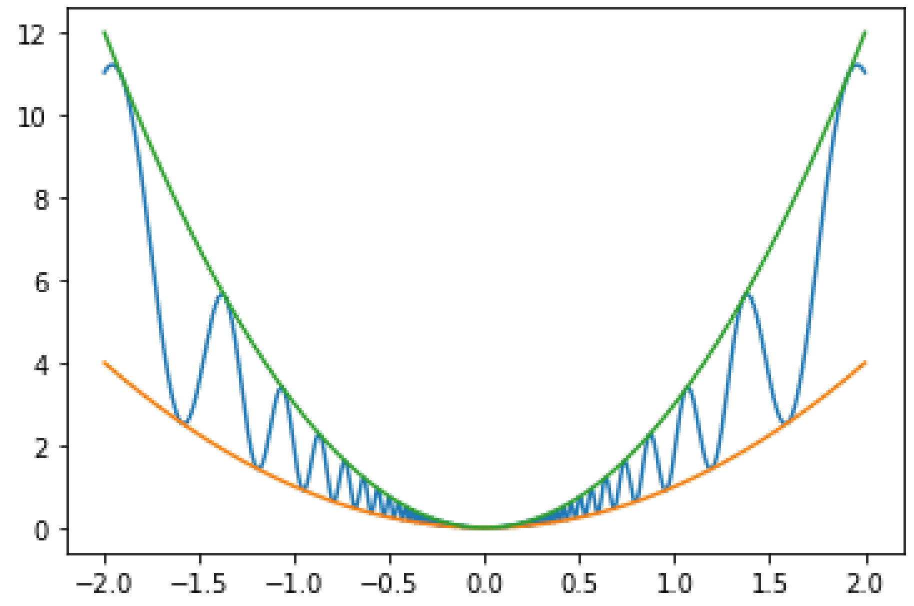

Local Minimizers
Strict Isolated Local Minimizer
Strict Isolated Global Minimizer

Illustration of a strict minimizer That is not isolated.



OPTIMALITY CONDITIONS

- Suppose that $f\colon \mathbb{R}^n \to \mathbb{R}$ is once continuously differentiable.

First-Order Necessary Optimality Condition
If the vector $x^* \in \mathbb{R}^n$ is a local minimizer, then $\nabla f(x^*) = 0$.

- Suppose that $f\colon \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable.

Second-Order Necessary Optimality Conditions
If the vector $x^* \in \mathbb{R}^n$ is a local minimizer, then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*) \succcurlyeq 0$ (positive semi-definite).

Second-Order Sufficient Optimality Conditions
If the vector $x^* \in \mathbb{R}^n$ is a local minimizer, then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*) \succ 0$ (positive definite).

WHAT IS A DIRECTION OF DESCENT?

Descent Direction

A vector $p \in \mathbb{R}^n$ is a descent direction for the continuously differentiable function $f: \mathbb{R}^n \to \mathbb{R}$ when evaluated at the point $x \in \mathbb{R}^n$ if

$$\nabla f(x)^T p < 0.$$

Proof

Recall that the dot product of two vectors p and $\nabla f(x)$ is defined geometrically as

$$\nabla f(x)^T p = \|\nabla f(x)\| \cdot \|p\| \cos\theta \, .$$

Here, θ is the angle between the two vectors. Since $\|\nabla f(x)\| \cdot \|p\|$ is always positive, we look at the value of $\cos\theta$. Since we know that $\cos\theta \geq 0$ when $\theta \in [0^{\circ}, 90^{\circ}]$ and $\cos\theta \leq 0$ when $\theta \in [90^{\circ}, 270^{\circ}]$, we can clearly see that if the direction p makes an obtuse angle with the gradient $\nabla f(x)$ (i.e., that p is a descent direction for f at the point x), then the dot product of the two vectors will be strictly less than 0, completing the proof.

THE SIMPLEST OPTIMIZATION ALGORITHM

- Assume that we are trying to find the minimum value of some function f, and we are currently at a point x that does not yield that optimal value.
- What is the most intuitive first-choice direction we can use that we **know** will give us descent at the point x?

Gradient Descent

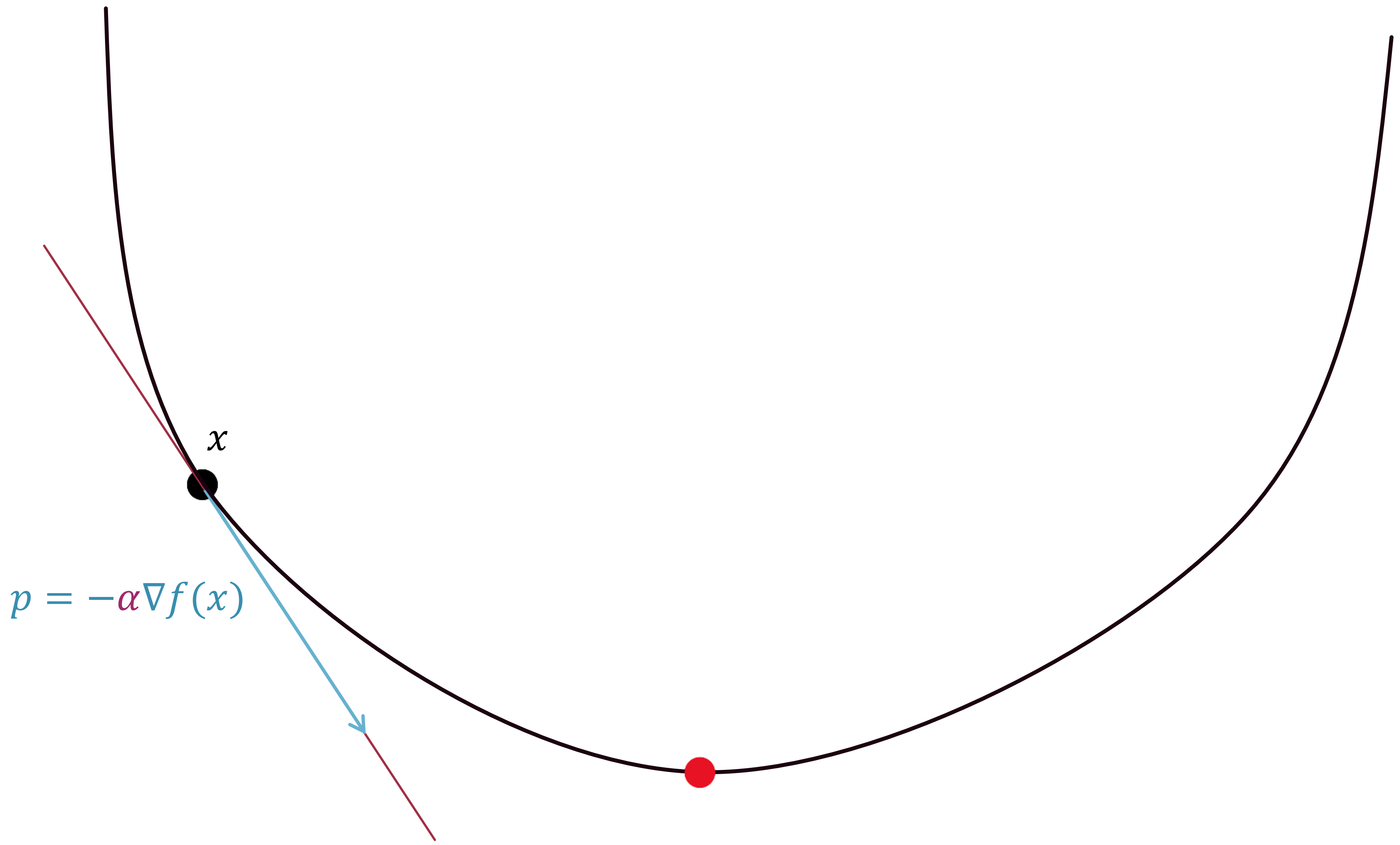
- Assuming we remember anything from Calculus 3, we know that the direction of *steepest ascent* is simply in the direction of the gradient itself, i.e., $\nabla f(x)$.
- Naturally, we immediately know that the *negative gradient* must be a direction that decreases the function (specifically, this is called the **direction of steepest descent**), simply written as $-\nabla f(x)$.
- As long as we take a "step" in the direction of the negative gradient that is small enough (remember, we're only guaranteed that this direction will be descent within a small neighborhood of the point x), then we know that the function f will decrease in value.
- Thus, if we continue to take small steps in this way, each in the direction of the negative gradient at those evaluated points, we are guaranteed to find a (local) minimum of the function f!

TYPICAL FORM OF AN OPTIMIZATION ALGORITHM

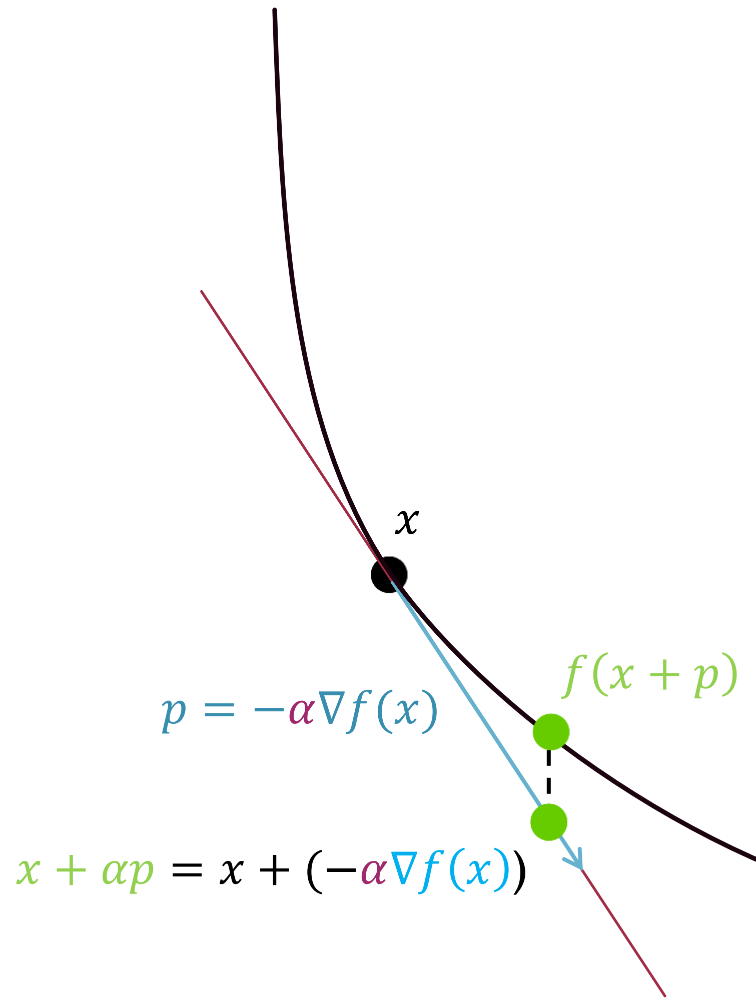
Algorithm 1 General form of an Optimization Algorithm

```
Input: $x_k \in \mathbb{R}^n$, $\{\alpha_k\}_{k\geq 0} > 0$.
For $k = 0, 1, 2, \ldots$ do
    Step 1. Choose $p_k$ to be a direction in which we think the function $f$ decreases.
    Step 2. Compute the update $x_{k+1} = x_k + \alpha_k \, p_k$.
End do
```

GRADIENT DESCENT – VISUALIZED

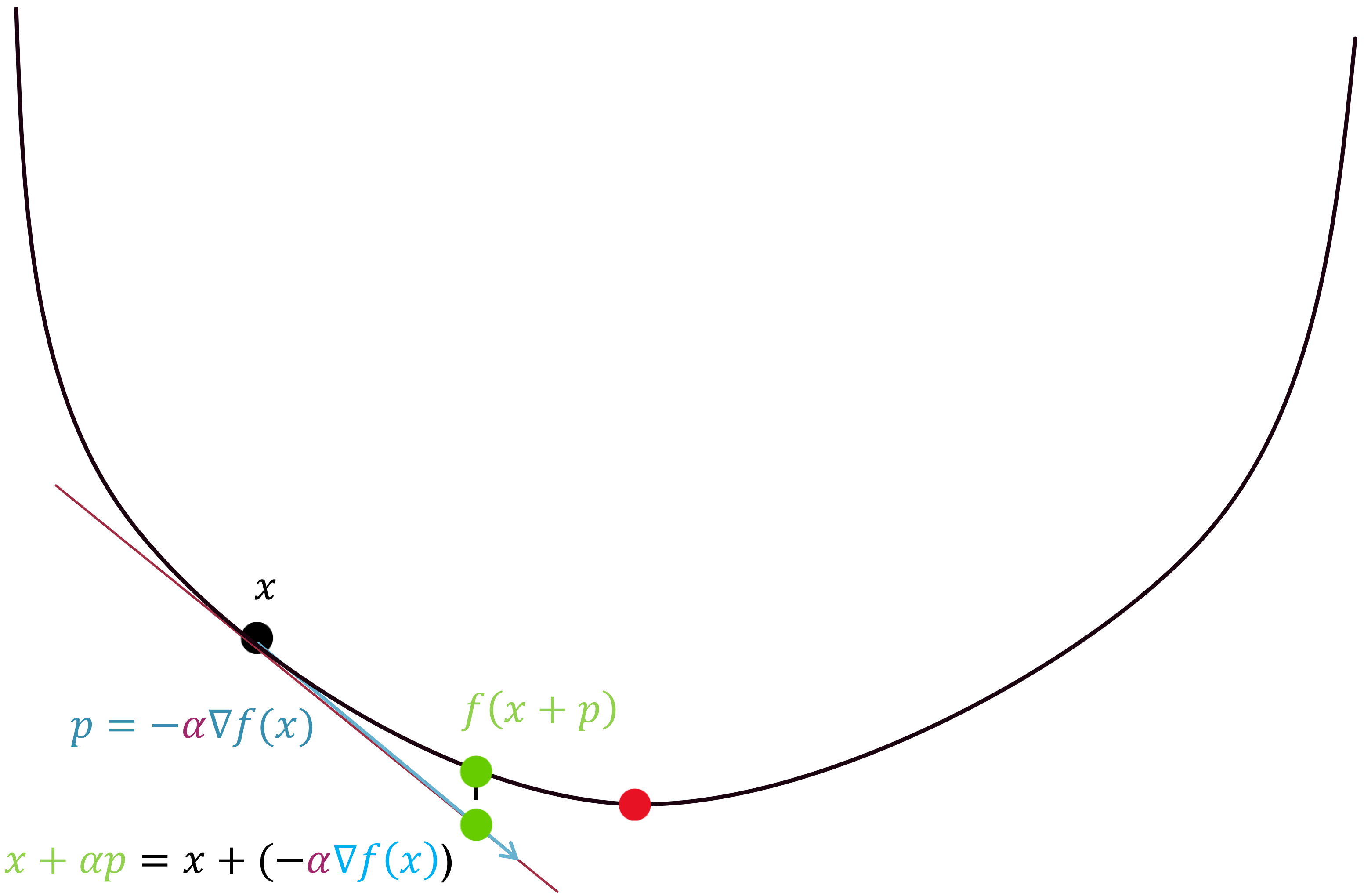


GRADIENT DESCENT – VISUALIZED



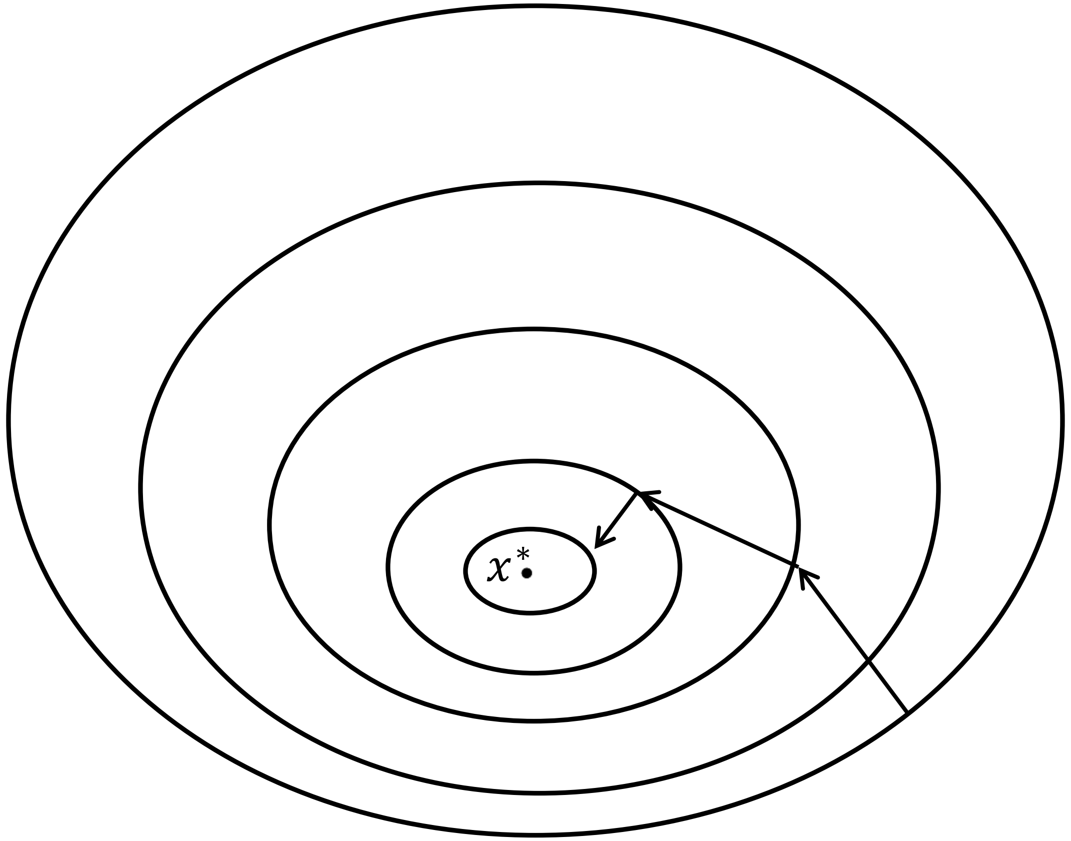
x
$f(x+p)$
$p = -\alpha \nabla f(x)$
$x + \alpha p = x + (-\alpha \nabla f(x))$

GRADIENT DESCENT – VISUALIZED


x
$p = -\alpha \nabla f(x)$
$f(x+p)$
$x + \alpha p = x + (-\alpha \nabla f(x))$

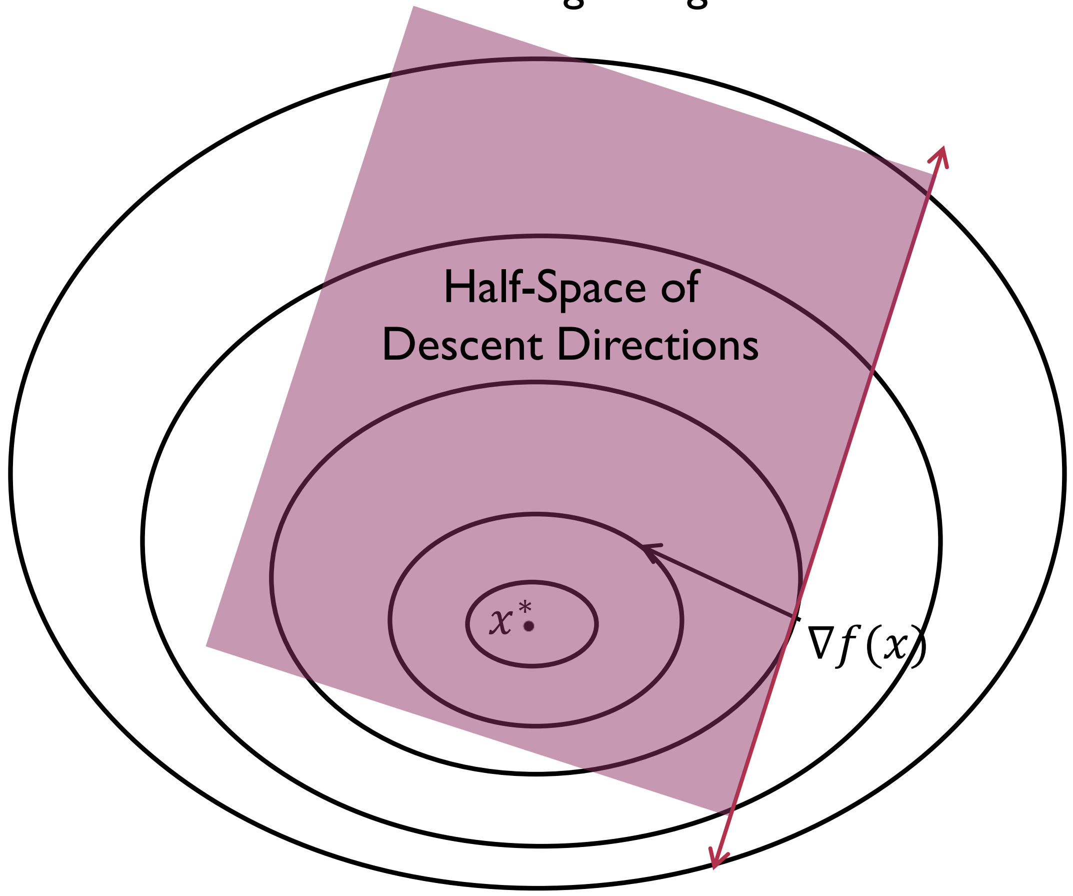
GRADIENT DESCENT – SOME GEOMETRY

The Steepest Descent Direction is perpendicular to the level curves of f.



GRADIENT DESCENT – SOME GEOMETRY

The half-space of descent directions
in relation to the negative gradient



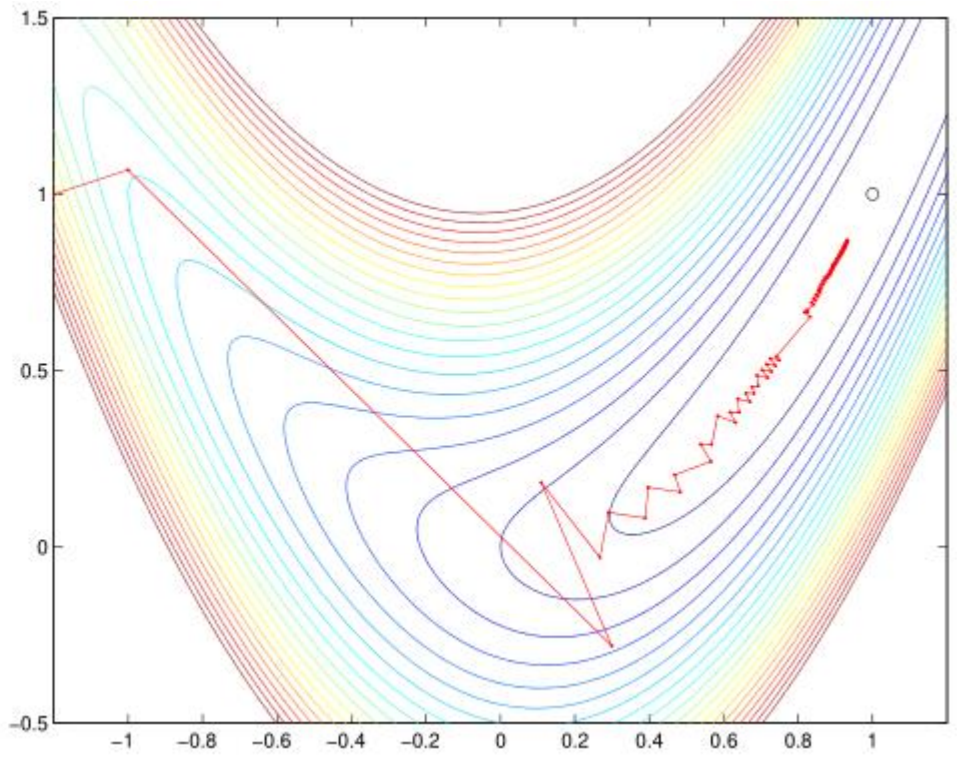
GRADIENT DESCENT – PROS & CONS

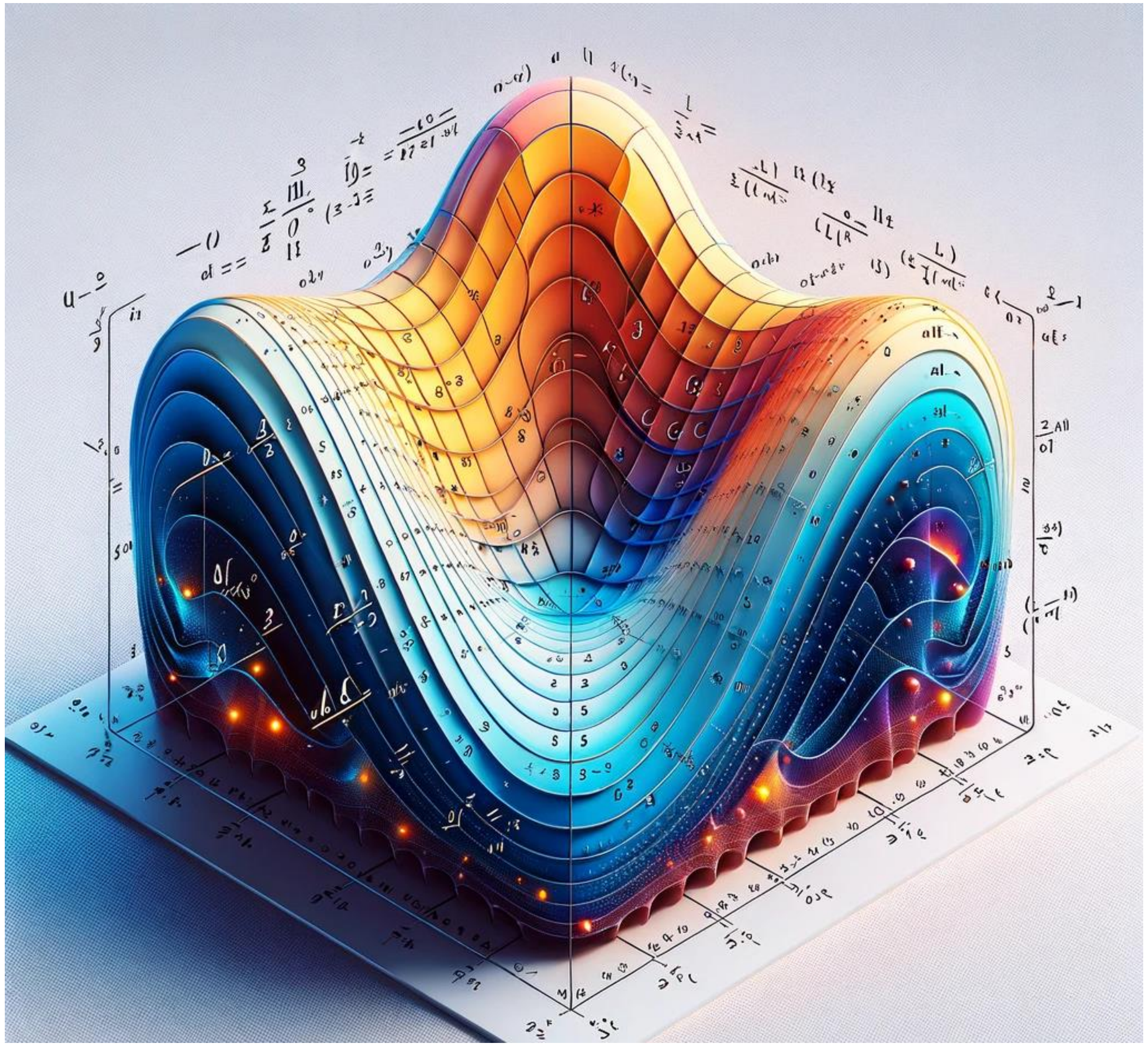
Pros

- Intuitive.
- Simple.
- Computationally cheap.
- Guarantees descent with proper step size.

Cons

- Lacks higher-order information.
- Not the best directions of descent.
- Terrible with ill-conditioned problems (more on this later).
- May take a long time to converge.





TAYLOR SERIES

-

THE HEART OF NONLINEAR OPTIMIZATION

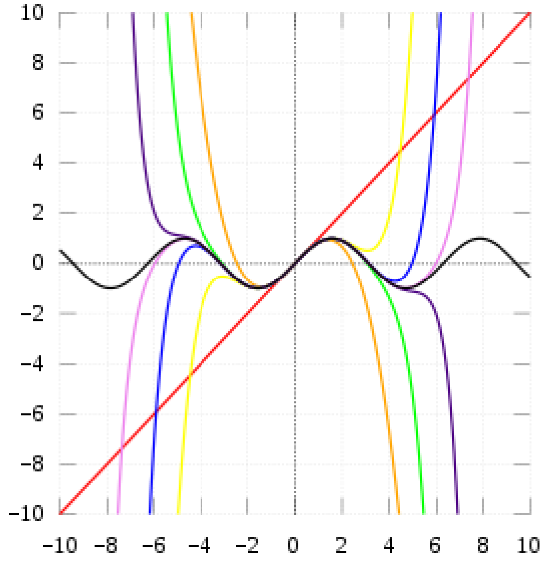
TAYLOR SERIES: THE INTUITION FOR NONLINEAR CONTINUOUS OPTIMIZATION

The Taylor Series

Let $f: \mathbb{R} \to \mathbb{R}$ such that the n-th derivative of f exists and is continuous at a point $a \in \mathbb{R}$. Then, the Taylor expansion of f around the point x is given by:

$$f(x) = f(a) + f'(a)(x-a) + \frac{1}{2!} f''(a)(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n + R_n(x).$$

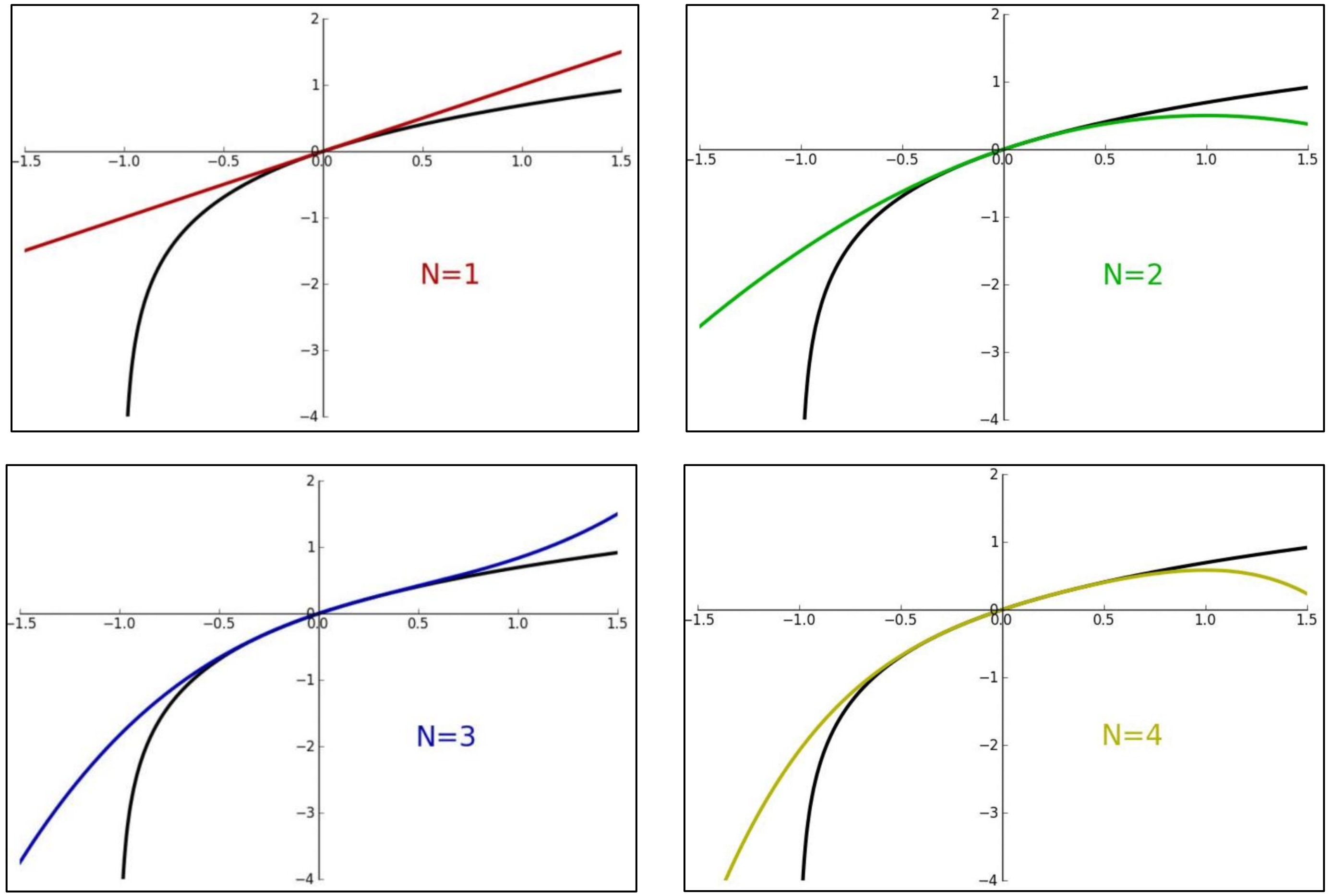
Here, $R_n(x) = \mathcal{O}\left((x-a)^{n+1}\right)$ is called the remainder.



- The Taylor Series (or Taylor Expansion) of a function is an infinite sum of terms, each defined by the function's derivatives.
- Named after Brook Taylor, who introduced it in 1715.
- TS is one of the **most powerful tools** of analysis.
- TS allows one to **approximate** smooth (differentiable) functions in a small neighborhood to **arbitrary precision** by using polynomials.

TAYLOR SERIES VISUALIZED

Taylor Series approximations to different degrees



WHY TAYLOR SERIES?

Why are Taylor Series Relevant to Optimization?

- Suppose that we are trying to optimize some function f that we can compute values of for different parameters, but we "don't know what it looks like" or how it behaves (this is the case with essentially all machine learning problems).
- However, we do know that we have this handy-dandy way of approximating any continuously differentiable function via **Taylor approximations**!
- Thus, the intuition is the following: Instead of trying to optimize the function f directly (which is difficult since we don't know its complex nature), we form an approximation of f (which is accurate in a small neighborhood around a point x) via a truncated Taylor approximation, and then proceeded to minimize that approximation.
- Of course, since we are simply minimizing a surrogate approximation of the function that we are interested in (instead of the function itself), we will need to perform several different Taylor approximations as we take small enough steps toward the optimal solutions. Thus, we are really optimizing a series of approximations, but in doing so, we are indirectly optimizing our desired function f.

COMMON TAYLOR APPROXIMATIONS

- Typically, in practice we only utilize the first-order and second-order Taylor expansions. Why?
- Remember, we are using the Taylor approximation as a function that is **very similar** to our function of interest (within a neighborhood), but that is one that we can optimize cleanly. If we start using higher-order derivatives, we will likely be trying to minimize approximation functions that now have added optima that may throw off our goal.

First-Order Taylor Expansion

$$f(x+p) = f(x) + \nabla f(x)^T p + \mathcal{O}(\|p\|^2).$$

Thus, when p is small enough (which we can control with the **learning rate** $\alpha \in \mathbb{R}$), then this function is essentially equal to the 1st-order truncated Taylor approximation:

$$f(x+p) \approx f(x) + \nabla f(x)^T p.$$

Second-Order Taylor Expansion

$$f(x+p) = f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x) p + \mathcal{O}(\|p\|^3).$$

Similarly, when p is small enough, then this function is essentially equal to the 2nd-order truncated Taylor approximation:

$$f(x+p) \approx f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x) p.$$

DERIVING STEEPEST DESCENT

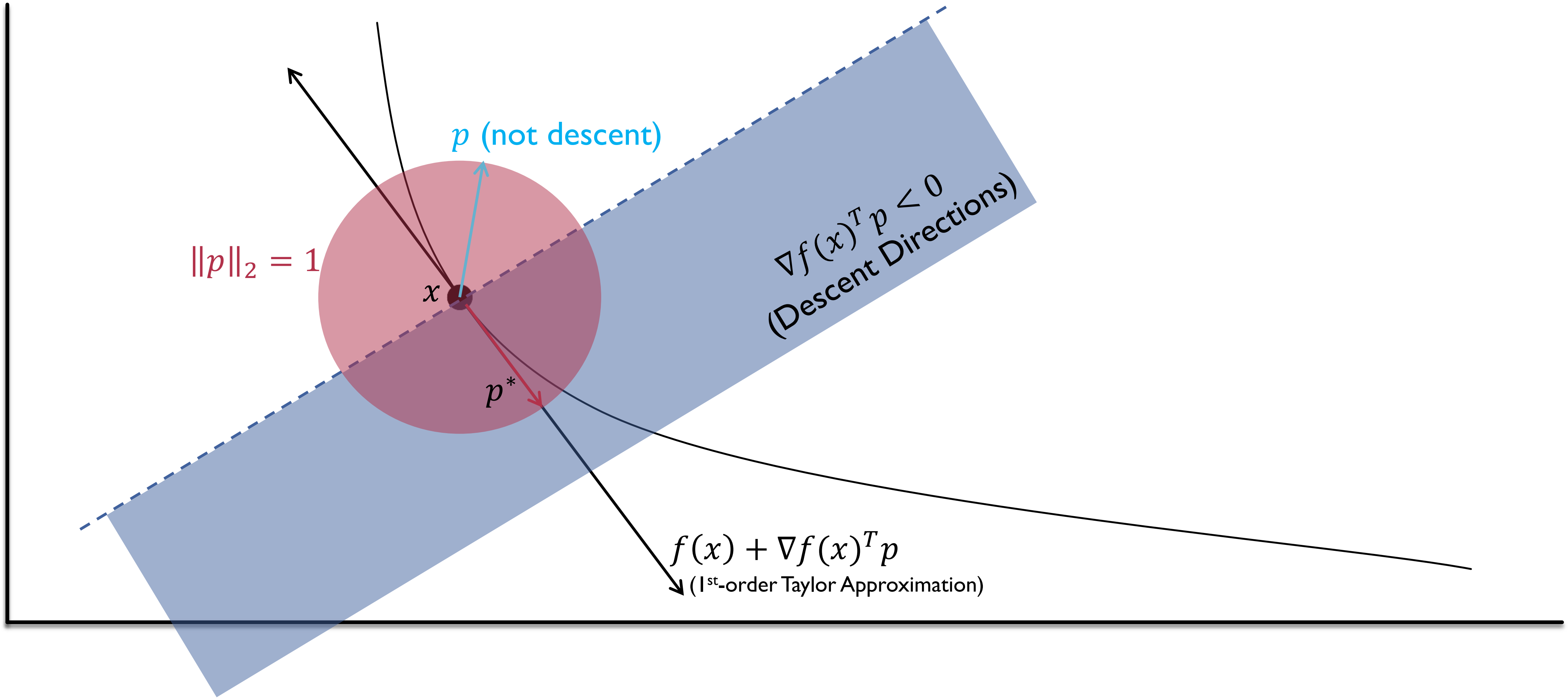
Consider the following **1st-order truncated Taylor approximation** around the point x:

$$f(x+p) \approx f(x) + \nabla f(x)^T p.$$

Notice that this is simply the **linear approximation** of f at the point x as $p \to 0$ (see the definition of the derivative).

- Assuming that $\nabla f(x) \neq 0$ (i.e., x is not at a stationary point), we wish to choose a p such that $f(x+p) \leq f(x)$.
- Thus, we can simply minimize $f(x+p)$ over p to yield the direction of greatest decrease, i.e.,
$$\min_p f(x) + \nabla f(x)^T p.$$
- Notice that since $f(x)$ is simply a constant in this equation (and adds no information when choosing p), we can simply drop it to obtain the problem
$$\min_p \nabla f(x)^T p.$$
- However, this problem, as is, has no solution (because we can minimize it infinitely). Remember, we are simply using the 1st-order Taylor Series as an approximation to our function f and trying to find a descent direction p. Thus, we can simply implement a unit-bound constraint $\|p\|_2 = 1$, yielding the well-defined problem
$$\min_{\|p\|_2=1} \nabla f(x)^T p.$$
- Recalling the definition of the dot-product, we have $\nabla f(x)^T p = \|p\| \|\nabla f(x)\| \cos\theta = \|\nabla f(x)\| \cos\theta$ which will obtain a minimum when $\cos\theta = -1$, yielding $p^* = -\frac{\nabla f(x)}{\|\nabla f(x)\|}$, i.e., the **unit steepest descent direction**.

DERIVING STEEPEST DESCENT FROM TAYLOR SERIES



DESCENT OF THE STEEPEST DESCENT DIRECTION

Descent of the Steepest Descent Direction

Let the function $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable. Given some $x \in \mathbb{R}^n$, the search direction $p = -\nabla f(x)$ is a descent direction as long as $\nabla f(x) \neq 0$.

Proof

By the definition of a descent direction, we have

$$\nabla f(x)^T p = -\nabla f(x)^T \nabla f(x) = -\|\nabla f(x)\|^2 < 0.$$

Therefore, $p = -\nabla f(x)$ is a descent direction.

DERIVING NEWTON'S METHOD

Consider the following **2nd-order truncated Taylor approximation** around the point x:

$$f(x+p) \approx f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x) p.$$

Notice that this is a **quadratic approximation** of f at the point x as $p \to 0$.

- Assuming that $\nabla f(x) \neq 0$ (i.e., x is not at a stationary point), we wish to choose a p such that $f(x+p) \leq f(x)$.
- Thus, we can simply minimize $f(x+p)$ over p to yield the direction of greatest decrease, i.e.,

$$\min_p f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x) p\,.$$

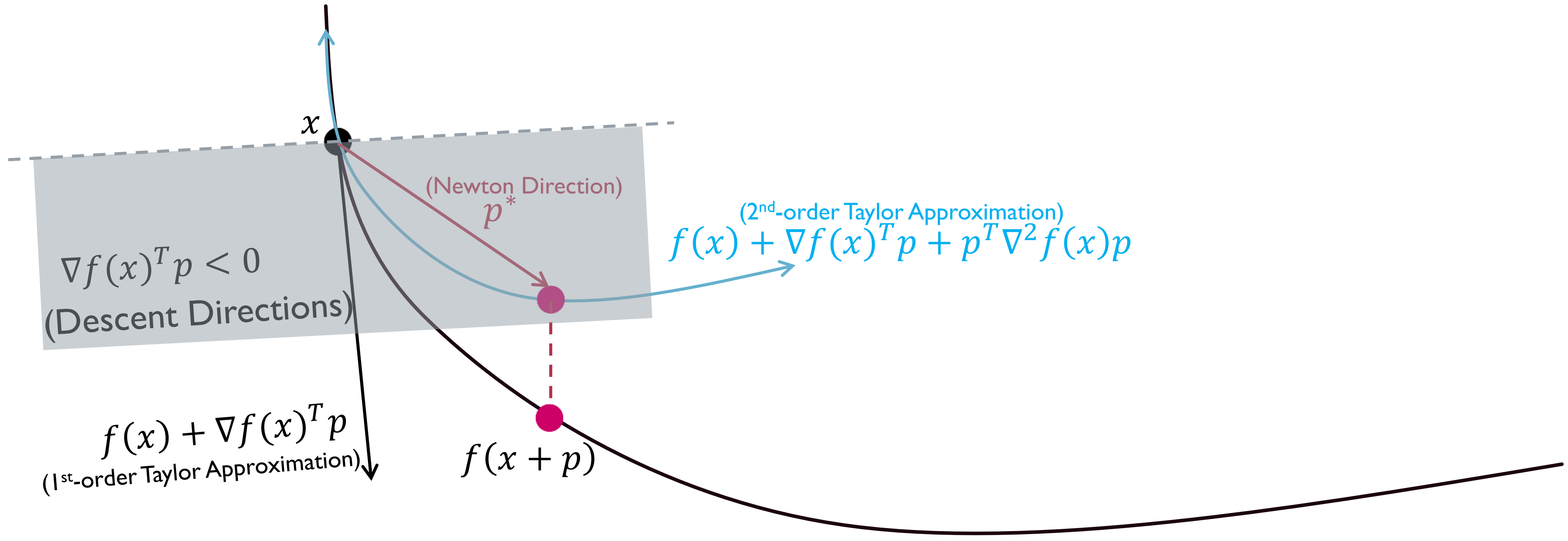
- Setting the derivative of this equation equal to 0 and solving for p, we obtain the following direction:

$$p = -\nabla^2 f(x)^{-1} \nabla f(x).$$

This is known as the **Newton Direction** (hence the name **Newton's Method**).

- What is an important caveat of this approach? Do we know that the Newton Direction will be descent?

DERIVING NEWTON'S METHOD FROM TAYLOR SERIES



x
(Newton Direction)
p^*
(2nd-order Taylor Approximation)
$f(x) + \nabla f(x)^T p + p^T \nabla^2 f(x) p$
$\nabla f(x)^T p < 0$
(Descent Directions)
$f(x) + \nabla f(x)^T p$
(1st-order Taylor Approximation)
$f(x + p)$

DESCENT OF THE NEWTON DIRECTION

Descent of the Newton Direction

Let the function $f: \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable. Given some $x \in \mathbb{R}^n$, the search direction $p = -\nabla^2 f(x)^{-1} \nabla f(x)$ is a descent direction as long as $\nabla f(x) \neq 0$ and the Hessian $\nabla^2 f(x)$ is positive definite, i.e., $s^T \nabla^2 f(x) s > 0$ for all non-zero $s \in \mathbb{R}^n$.

Proof

Recall that the inverse of a symmetric positive definite matrix is positive definite, i.e., $s^T \nabla^2 f(x)^{-1} s > 0$ for all non-zero $s \in \mathbb{R}^n$ (this can be proven). By definition, $p \in \mathbb{R}^n$ is a descent direction since we have

$$\nabla f(x)^T p = -\nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x) < 0.$$

Therefore, we know that the Newton Direction, when the Hessian is positive definite, is a descent direction.

NEWTON'S METHOD – PROS & CONS

Pros

- Very fast convergence (much faster than GD).
- Intuitive (NM is essentially a root-finding algorithm).

Cons

- Must be "sufficiently" close to the minimum already (i.e., the Hessian must be positive definite, otherwise NM may find a saddle point or even a maximum).
- If the Hessian is not positive definite, the Newton direction may not even be defined since $\nabla^2 f(x)^{-1}$ may not exist.
- Even if $\nabla^2 f(x)^{-1}$ does exist, the Newton direction may not be a descent direction.
- Not computationally feasible for large scale problems due to the use of the Hessian.

Alterations to Newton's Method

- When the Hessian $\nabla^2 f(x)$ is far away from the solution it may not be positive definite (meaning the Newton Direction may not be descent), so instead, replace the Hessian with a generic symmetric definite matrix $B \in \mathbb{R}^{n\times n}$ that is updated at every iteration.
- This is known as the **Modified Newton Method** and the corresponding Modified Newton Direction would be

$$p = -B^{-1}\nabla f(x).$$

- For large-scale problems, **Quasi-Newton Methods** can be implemented which do not require the computation of the Hessian but are still able to obtain fast convergence rates.

A NOTE ON GRADIENT DESCENT VS. NEWTON'S METHOD

The Modified Newton Direction is obtained from the following 2nd-order Taylor Series (with the Hessian replaced with a generic symmetric definite matrix $B \in \mathbb{R}^{n\times n}$):

$$f(x+p) \approx f(x) + \nabla f(x)^T p + \frac{1}{2} p^T B p.$$

Further, the Modified Newton direction is given by

$$p = -B^{-1}\nabla f(x).$$

- If one were to use the exact Hessian $B = \nabla^2 f(x)$, then the exact Newton Direction is obtained.
- However, if we instead choose B to be the simplest symmetric positive definite matrix I (the identity matrix), then we obtain the steepest descent direction!

This highlights an important difference between gradient descent and Newton's method and the reason for why Newton's method is so much more efficient (in terms of better search directions, yielding a lower number of required iterations).

- Gradient descent approximates the Hessian matrix $\nabla^2 f(x)$ with the identity matrix I.
- Further, if the exact Hessian $\nabla^2 f(x)$ is approximately equivalent to the identity matrix, then gradient descent will perform the same as Newton's method.

STOCHASTIC GRADIENT DESCENT – A BRIEF INTRODUCTION

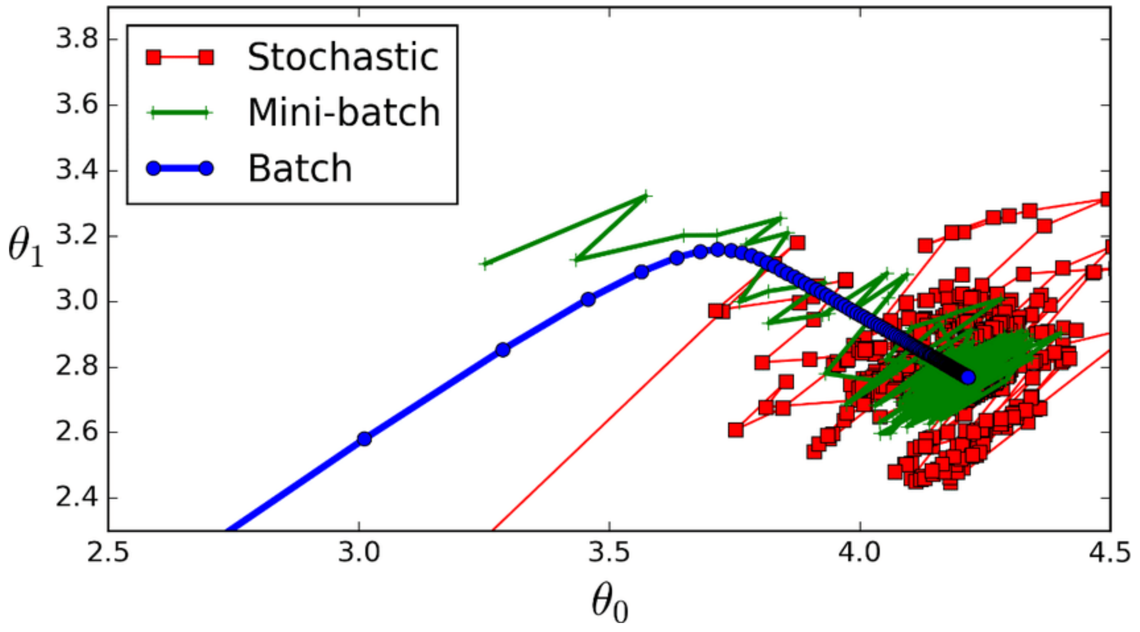
Stochastic Gradient Descent (SGD)

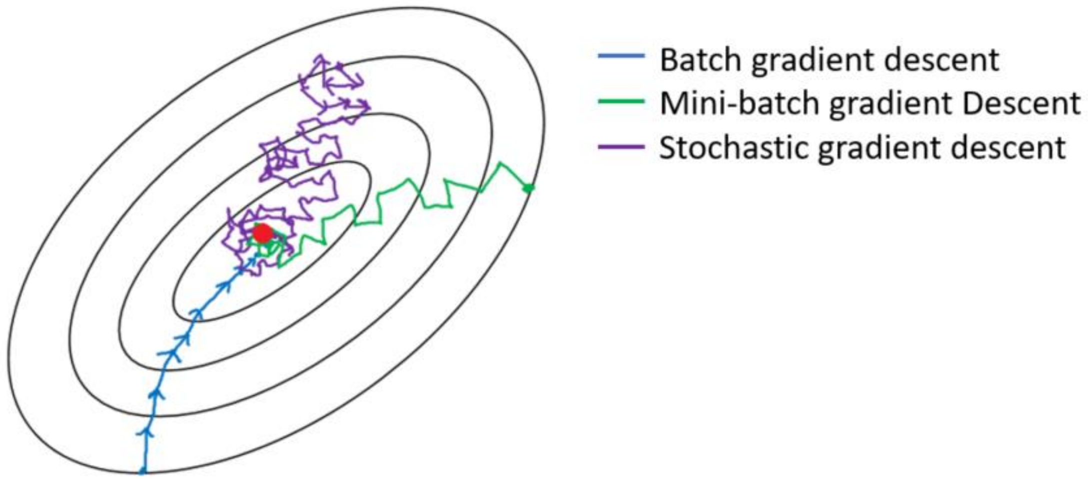
- Often, the function f that we wish to optimize has some amount of randomness (**stochasticity**) present, which we can denote with the random variable $\xi \sim \mathcal{D}$ that parameterizes f, where $\mathcal{D}$ is the underlying distribution and is typically unknown.
- In such cases, we are no longer in trying to optimize f, but we are instead trying to optimize what is referred to as the **expected risk**, defined as $\mathcal{R}(x) \coloneqq \mathbb{E}_{\xi \sim \mathcal{D}}[f(x;\xi)]$.
- As such, SGD simply applies classical gradient descent to minimize $\mathcal{R}(x)$, only now the gradient that we use as our search direction is a stochastic gradient, denoted by $\nabla f(x;\xi)$, which is typically assumed to be an unbiased estimator of $\nabla \mathcal{R}(x)$ (the true gradient of the expected risk function; which we don't typically have access to).

SGD in Practice

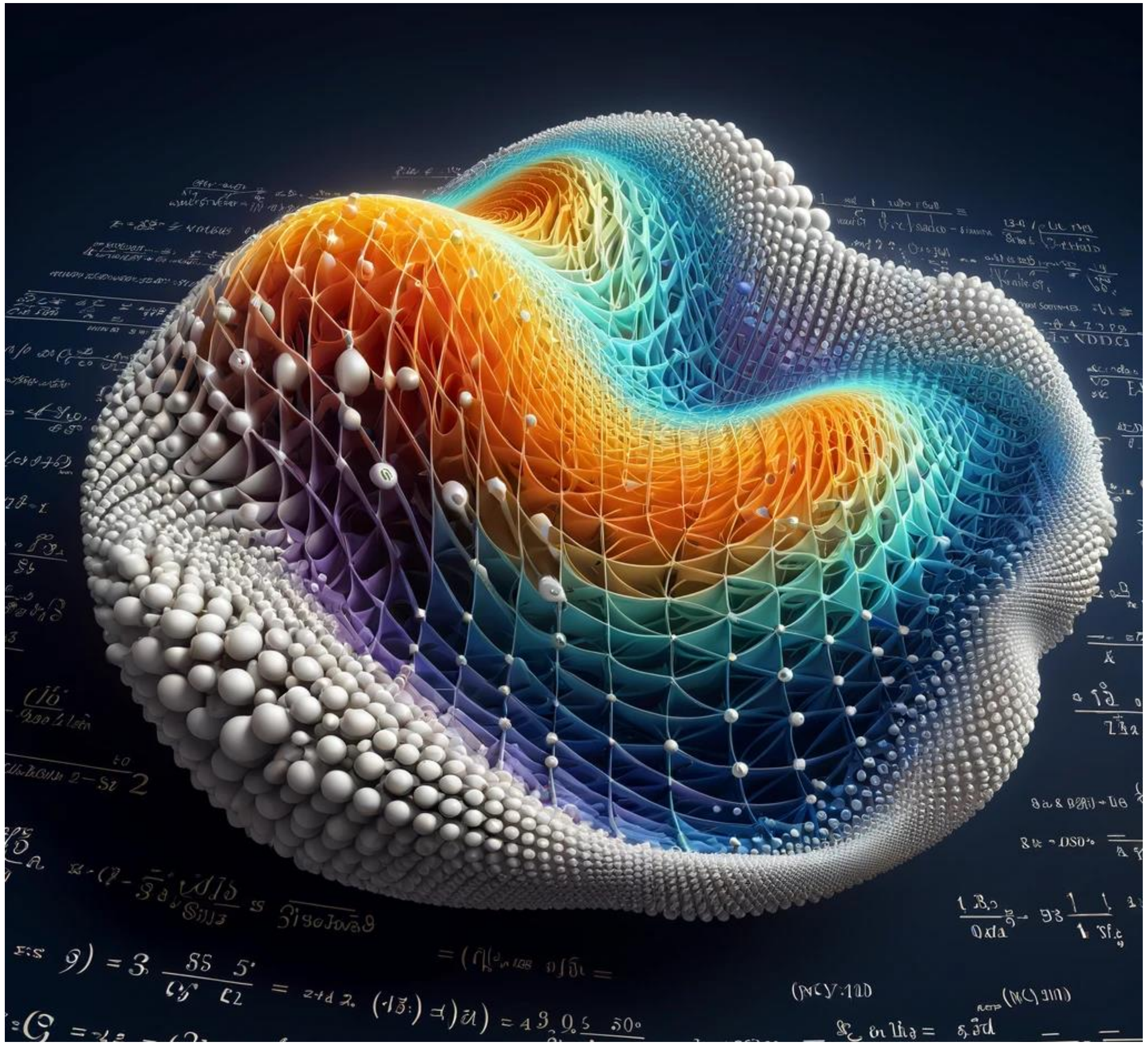
- SGD (and its variants) is the algorithm of choice when it comes to optimize machine learning problems. Why?
- Because SGD is **VERY** computationally efficient on ML problems that have massive datasets (millions+ datapoints).
- Applying classical GD to a ML problem would require you to use ALL the data to compute a gradient at a given point.
- The idea behind SGD is to instead use a **small random sample** of the datapoints to obtain an approximation of the gradient (a stochastic gradient), which will still most likely be relatively close to the real thing.
- As such, SGD will require different assumptions compared to GD to ensure convergence, but more on that later.

ILLUSTRATION OF DIFFERENT GRADIENT DESCENT SCHEMAS





***See Reference Slide**



CONVERGENCE & ANALYSIS OF ALGORITHMS

-

CONCLUDING REMARKS

CONVERGENCE ANALYSIS OF ALGORITHMS

What is Convergence Analysis?

In the research and design of a new optimization algorithm, it is typically expected that the algorithm has some sort of guarantee (in the form of a derived theorem) that it will yield a type of optimal solution to a certain type of problem. These are known as **convergence theorems** and offer different results: either rates of convergence to a solution or simply guarantees to a stationary point in the limit. Algorithms that don't have these guarantees are called Heuristics.

Some Typical Assumptions

Before any sort of analysis can begin, one is required to define the specifics of the problems that the proposed algorithm is meant to solve. This consists of a series of assumptions on the nature of the problem. Typical assumptions: **Continuity & Differentiability** (ensures derivative information exists), **Lipschitz Continuity** (ensures that the gradients and Hessians are "well-behaved"), **Convexity** or **Strong Convexity** (ensures certain properties and possibly the existence of an optimal solution), **Boundedness** (ensures the existence of a solution in convex and nonconvex problems), **Variance Bounds** and **Unbiased Expectation** (for stochastic problems), and **Step-Size Criteria**.

How to Choose the Step Sizes?

This is a task that is paramount to deriving convergence guarantees. There are a variety of different ways to go about doing this and we will only discuss a few.

CHOOSING STEP SIZES – FIXED & DECAY

Fixed Step Size

This is the simplest way to choose a step size as you essentially just choose a value and use it for every update of the variables. Although this is typically not a good strategy in practice, this is an approach that is seen everywhere in deriving convergence results for algorithms in research papers.

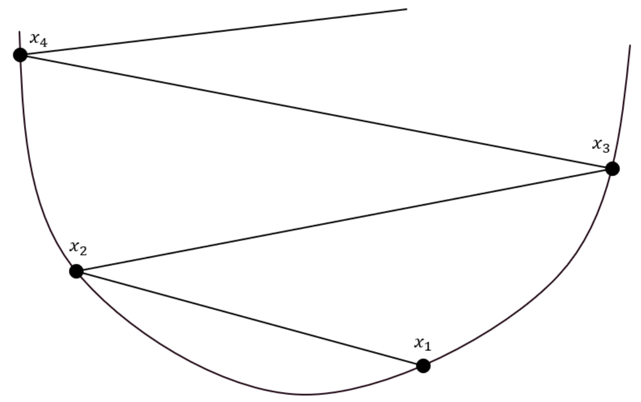
Decaying Step Size

Suppose that we take some number of iterations $k \in \{1,2,3, \dots\}$ in our algorithm. A decaying step size essentially has some form of $\alpha = \frac{1}{k}$, such that at an early number of iterations the step size is large, but "decays" to smaller and smaller values the more iterations that you take. Some form of this approach is the most common in ML problems.

Ways to Guarantee Descent?

There are indeed methods one can use to guarantee a step length that will yield descent.

- Some of these methods are Line Search methods, which essentially choose a step size that satisfies a "sufficient decrease" condition.
- However, we will not discuss these methods in this class since they are not typically used in ML algorithms.



Insufficient decrease in f.

A FINAL NOTE ON TYPES OF PROBLEMS & THEIR SOLUTIONS
(FOR CONTINUOUS NONLINEAR PROBLEMS)

Strongly Convex Problems

These problems are the "easiest" to solve in the sense that the function will have very nice properties, some of which are the guarantee of a unique optimal solution as well as Hessian-related guarantees (PD and non-singularity).

Convex Problems

A bit more difficult, but still considered "nice" in terms of properties. However, convexity alone does not guarantee that an optimal solution will exist (think of trying to minimize a line with a nonzero slope) and if one does exist, it may not be unique.

Nonconvex Problems

These are very difficult problems in terms of the types of convergence guarantees that one can guarantee. One doesn't have nice function properties as above and there can be many different isolated local optimum. Typically, the best one can do is ensure convergence to a stationary point.

Global Problems

The goal of these problems is to obtain the best possible solution for a problem. For strongly convex problems and most convex problems, this is straightforward. However, for nonconvex problems, one must be dealing with a bounded space, and even then, one may only be able to obtain global solution convergence in probability.

Stochastic Problems

These are problems that can have of the functional properties mentioned above, only with the additional need for handling noisy estimates (inexact gradients and Hessians, etc.). Convergence results are guarantees in expectation.

REFERENCES

- https://medium.com/analytics-vidhya/gradient-descent-vs-stochastic-gd-vs-mini-batch-sgd-fbd3a2cb4ba4
- https://en.wikipedia.org/wiki/Conic_section#/media/File:Conic_Sections.svg
- https://en.wikipedia.org/wiki/Taylor_series#/media/File:Logarithm_GIF.gif
- https://en.wikipedia.org/wiki/Conic_section#/media/File:TypesOfConicSections.jpg
- https://math.stackexchange.com/q/3495934
- Abramson, Kent, et al. "Penetration Depth Between Two Convex Polyhedra: An Efficient Stochastic Global Optimization Approach." 2022. doi: 10.1109/TVCG.2021.3085703
- Gomez, Lopez, et al. "Image Classification with Convolutional Neural Networks Using Gulf of Maine Humpback Whale Catalog." 2020. DOI: 10.3390/electronics905073.
- Gould, Leyffer. "An Introduction to Algorithms for Nonlinear Optimization." 2003. DOI: 10.1007/978-3-642-55692-0_4.