Curse of Dimensionality

Week3

Agenda

- Curse of dimensionality (limitation of KNN)
- Optimization
- Convexity
- Unconstrained optimization

Curse of dimensionality for k-NN

Consider:

Space: a unit cube

Points: uniformly distributed

Mini-cube: contain k=10 point

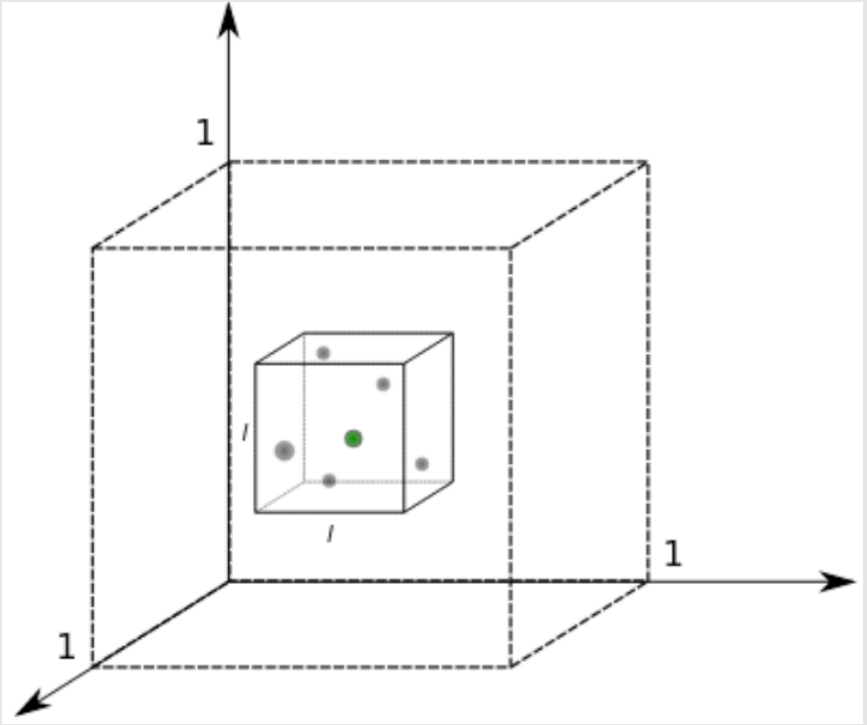
What's the volume given edge length l ?

$$l^D \approx k/N$$

Rewrite:

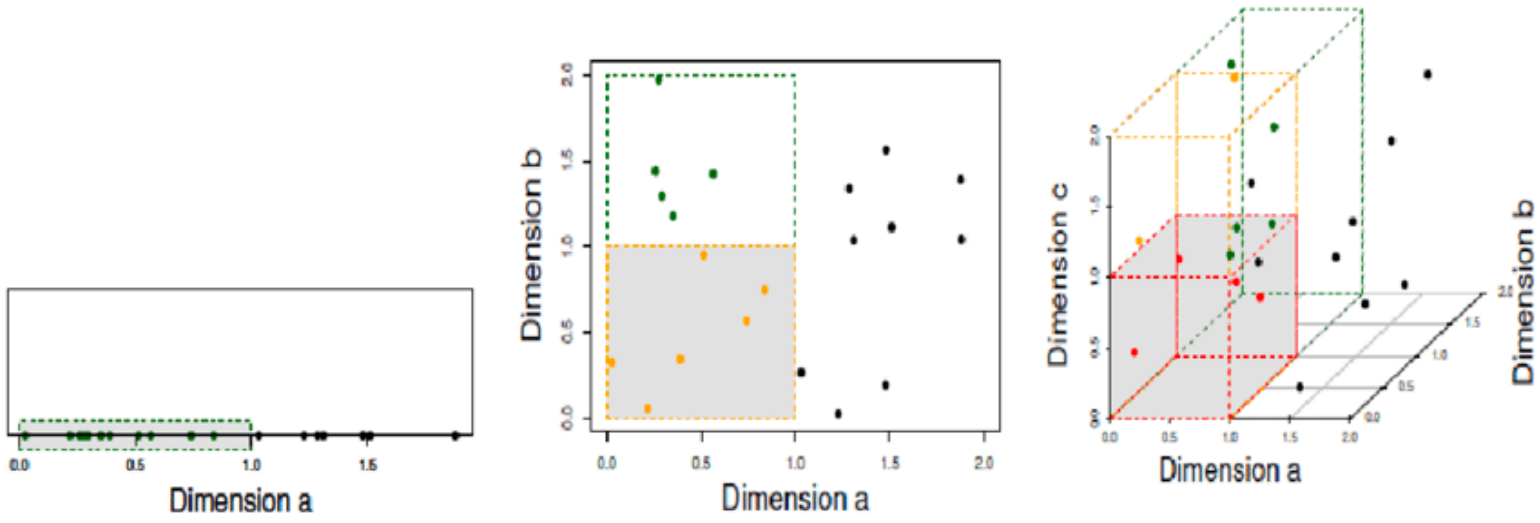
$$l = (k/N)^{1/D}$$

$$N = k/l^D$$



As we scale up N we cross more space

$$l = (k/N)^{1/D}$$

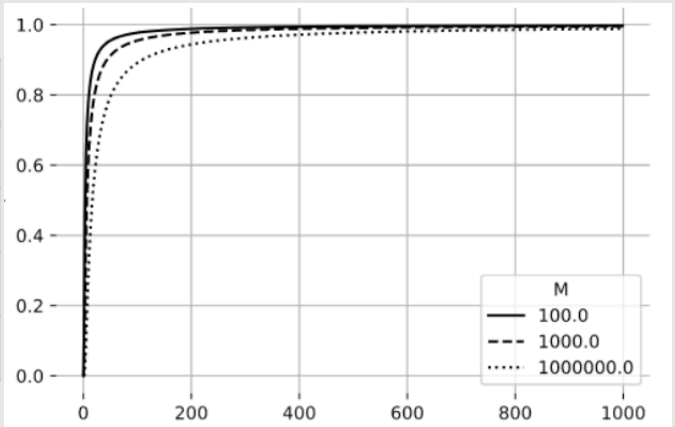


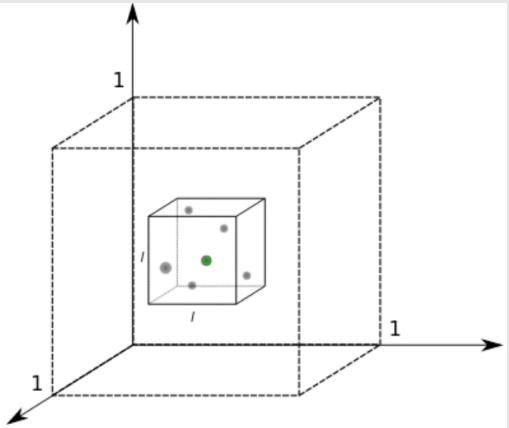
The effect is moderate in 1D-2D-3D

Beyond 3D

$$l = (k/N)^{1/D}$$

K/N	D	l
10/1000	2	0.1
10/1000	10	0.63
10/1000	100	0.955
10/1000	1000	0.9954

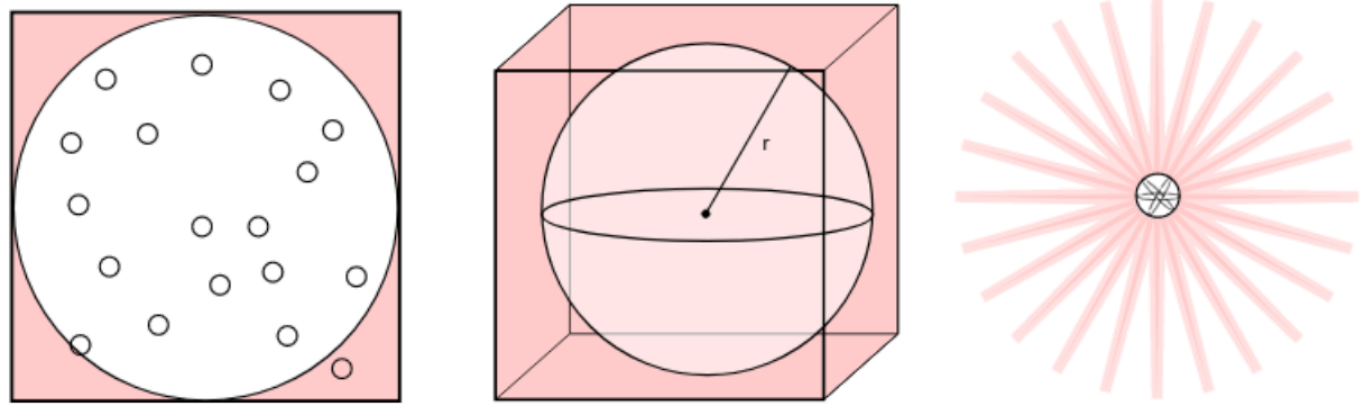




In 1000D the effect is huge, you need to cross the whole space to find 10NN

What does high dimensional space look like?

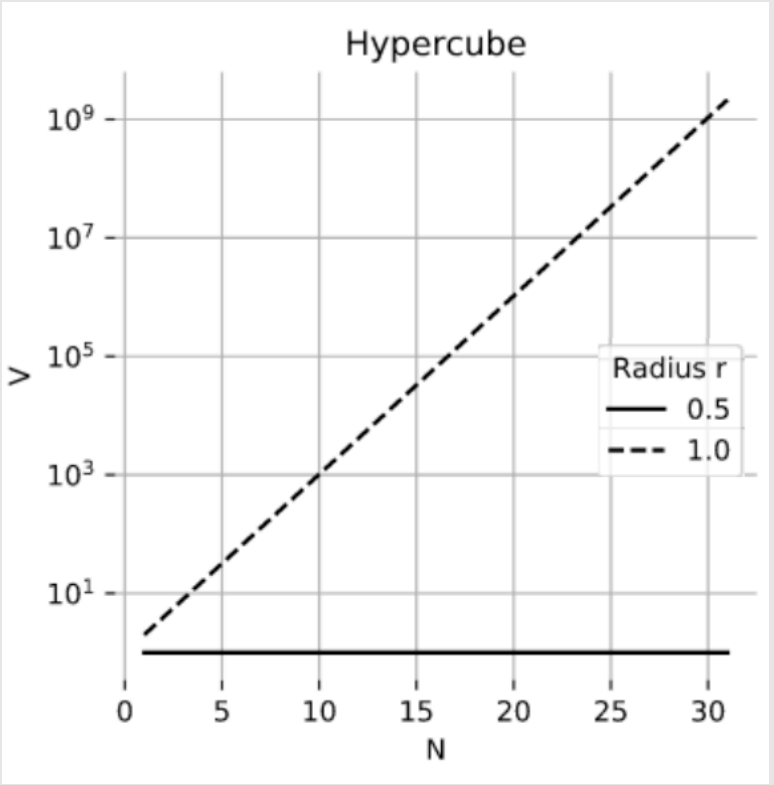
High dimensional spaces are extremely spiky
All the space is at the edges



Intuition: volume of hypercubes just grow

For a hypercube with side $l = 2r$

$$V_{cube}(N) = (2r)^N$$

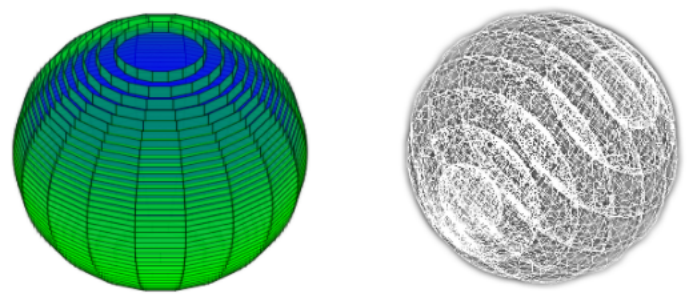


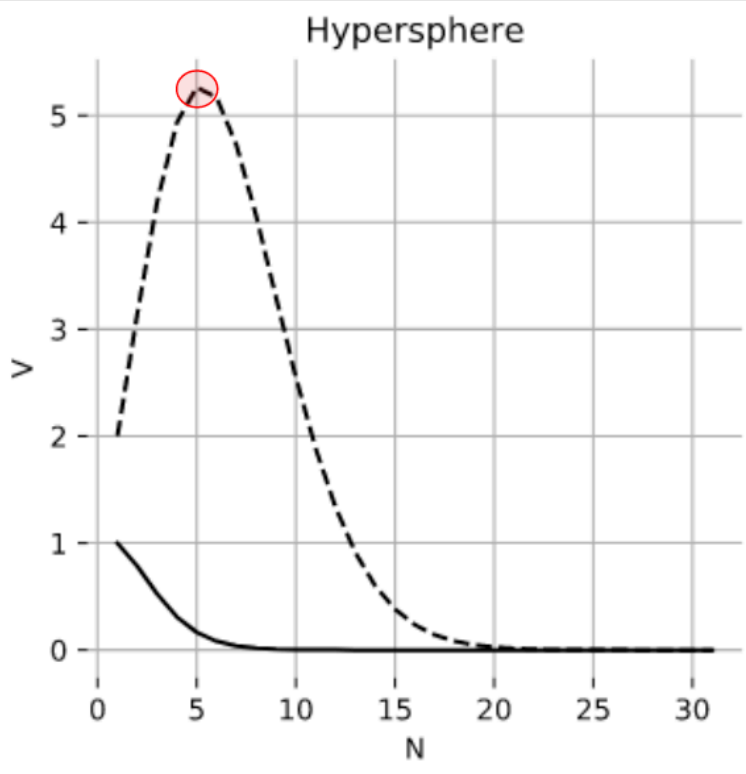
Intuition: volume of spheres drop

For a hypersphere with radius r

$$V_{cube}(N) = V_{sphere}(N-1)\int_{-1}^{1}\left(\sqrt{1-x^2}\right)^{N-1}dx$$

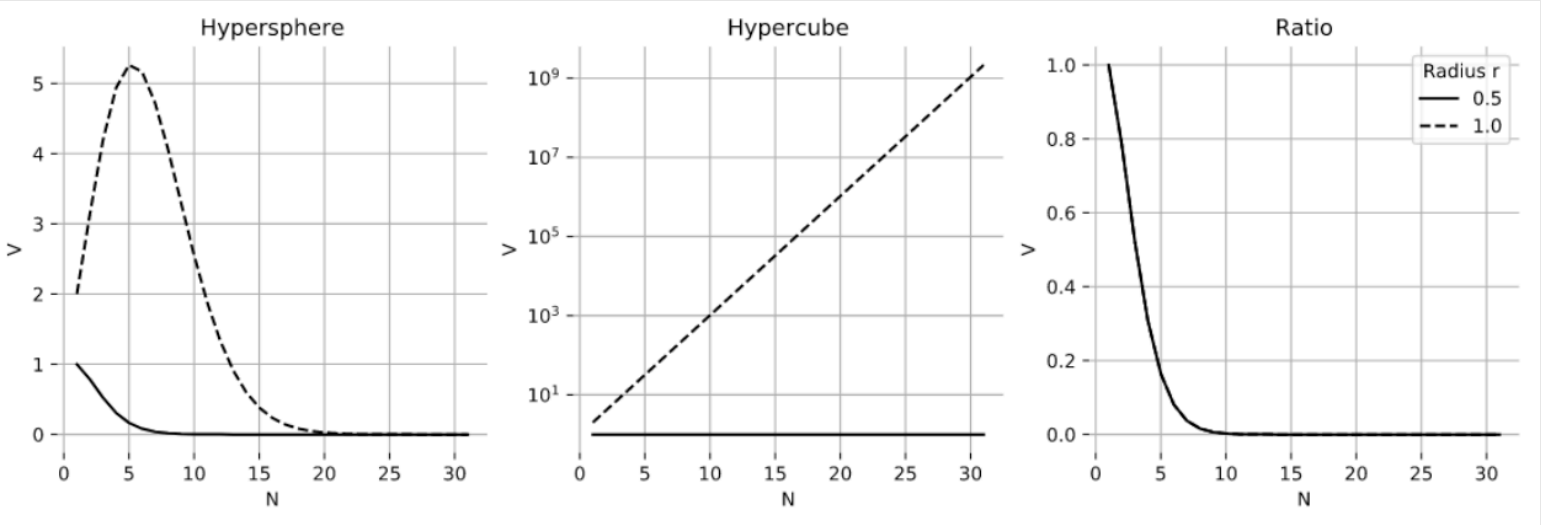
$$= \frac{r^N \pi^{N/2}}{\Gamma(\frac{N}{2}+1)}$$





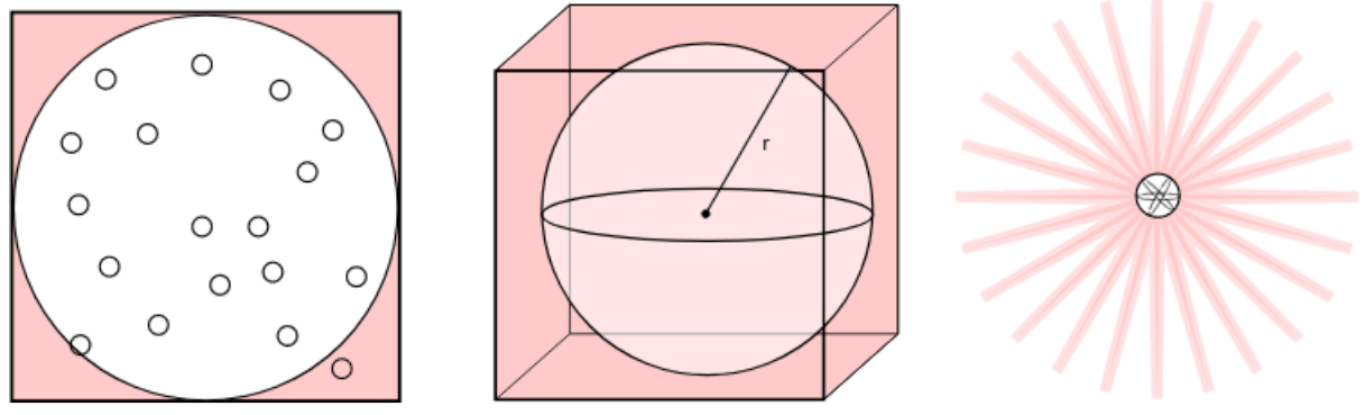
Intuition: all space is in the cube's edges

$$\lim_{N\to\infty} \frac{V_{\text{sphere}}}{V_{\text{cube}}} = \frac{\pi^{N/2}}{2^N \Gamma(N/2+1)} \to 0$$

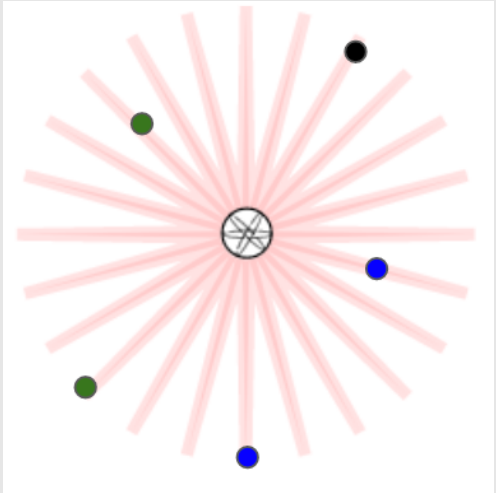


What do high dimensional space look like?

High dimensional spaces are extremely spiky
All the space is at the edges

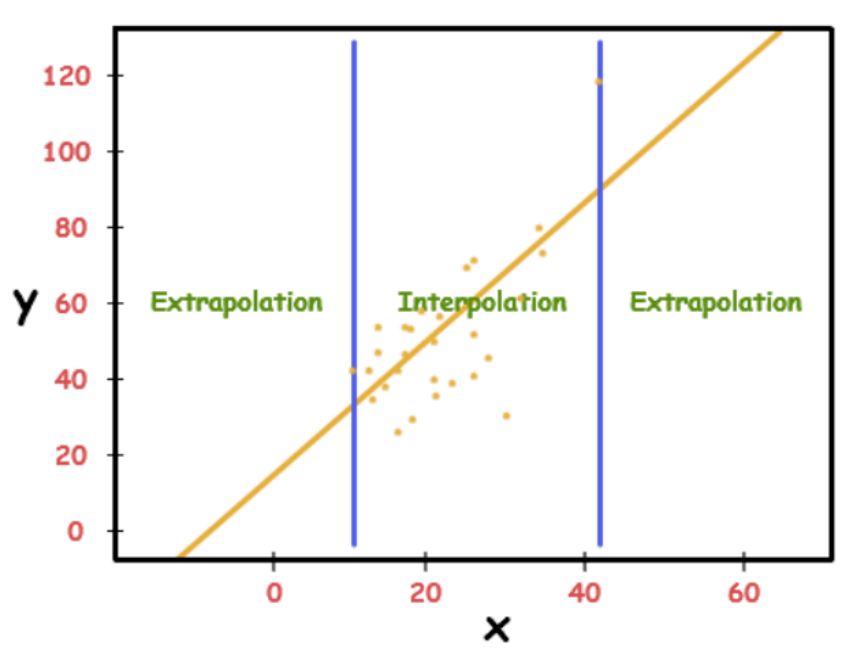


Why is this a problem



*“The reason that this presents a problem is that prediction is much more difficult near the edges of the training sample. [There ...] One must **extrapolate** from neighboring sample points rather than **interpolate** between them.” - Trevor Hastie*

Extrapolation and Interpolation



A couple of remedies

- Other algorithms suffers less from the curse of dimensionality
- ~~Adding data does not help (need to much)~~
- Reducing the dimensionality of the data does help

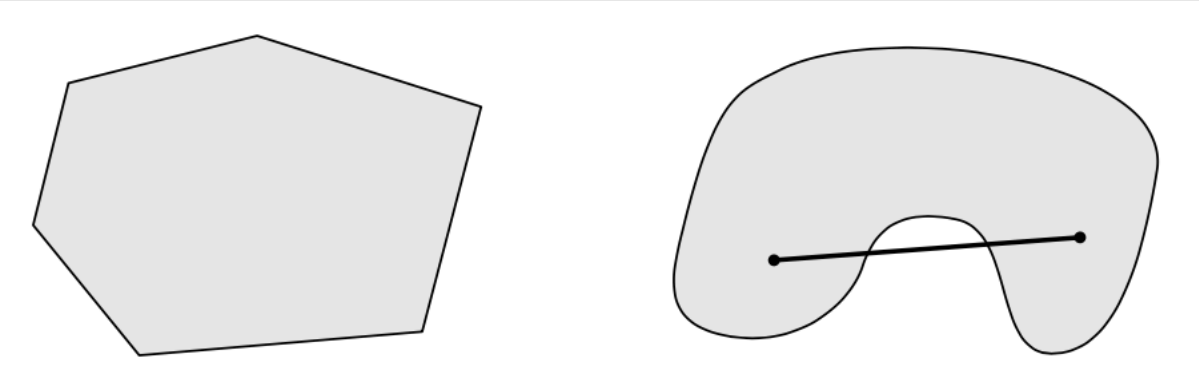
Convex Optimization

Convex sets

A set $\mathcal{C}$ is convex if, for $\forall x, y \in \mathcal{C}$ and $\theta \in R$ with $0 \leq \theta \leq 1$, we have

$$\theta x + (1 - \theta) y \in \mathcal{C}$$

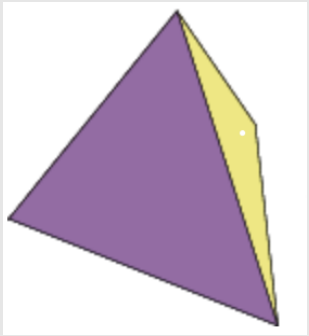
If we take any two elements in $\mathcal{C}$, draw a line between the two elements, then each point on the line also belongs to $\mathcal{C}$



Convex Set

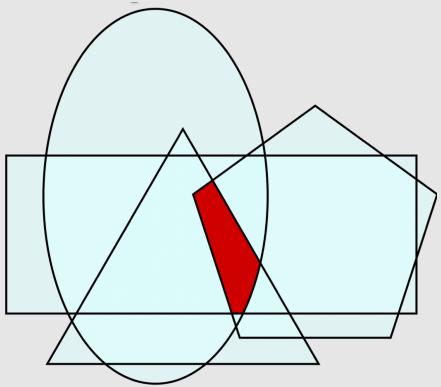
- R^n is a convex set for $\forall n$
- Affine subspaces $\{x|Ax = b\}$ where $A \in R^{m \times n}, b \in R^m$
- Polyhedron $\{x|Ax \leq b\}$

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



- Intersections of convex sets are convex.

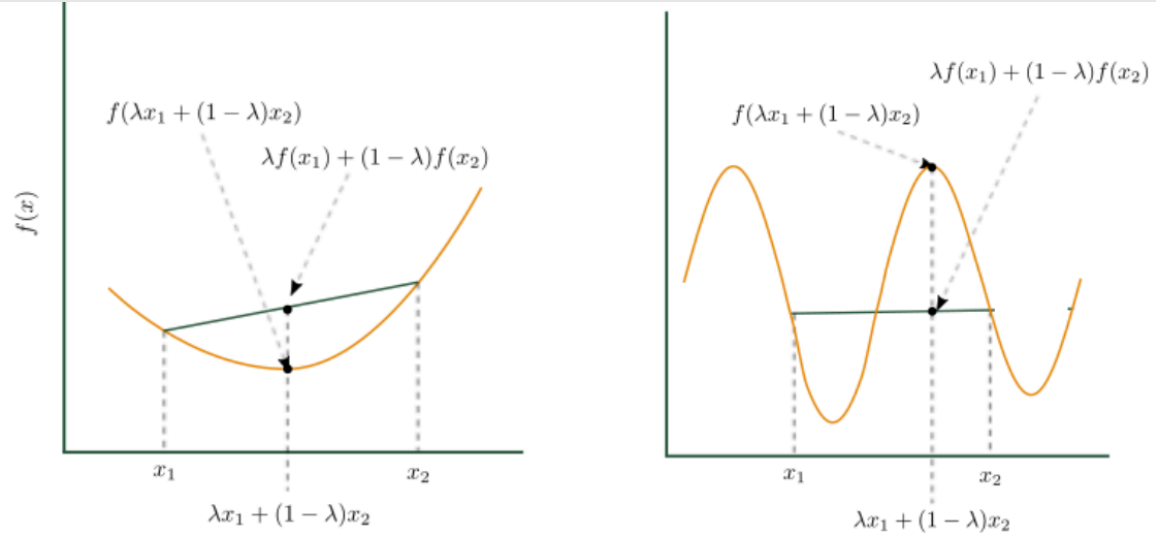
$\bigcap_i C_i$ with $C_1, C_2, \ldots$ are convex sets



Convex Functions

A function is convex if its domain $\mathcal{D}(f)$ is a convex set, and for $\forall x_1, x_2 \in \mathcal{D}(f)$, we have

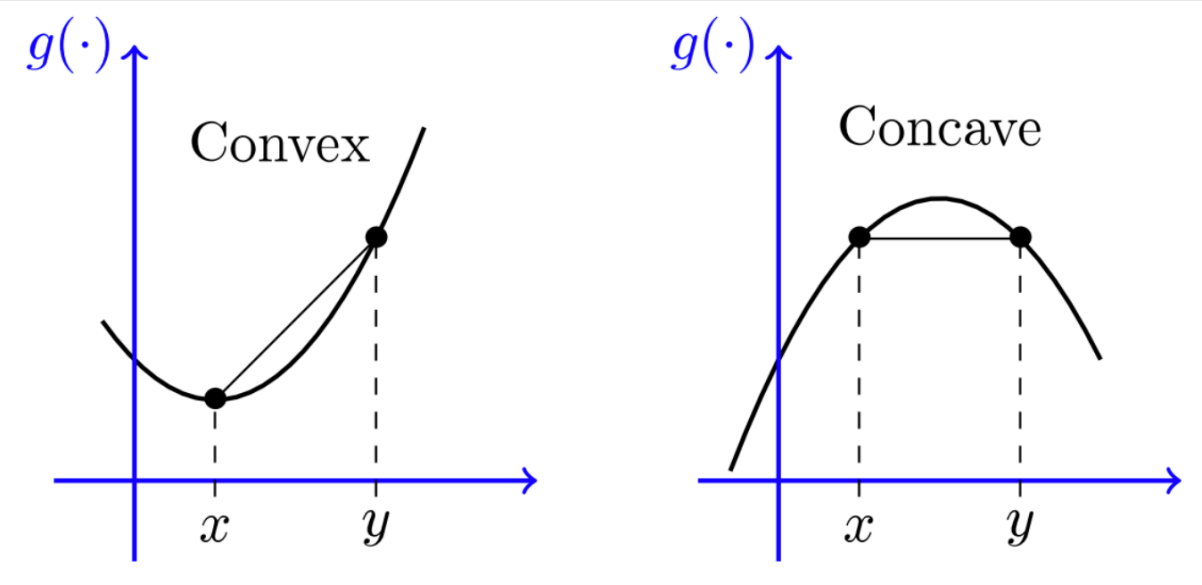
$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2) \quad \lambda \in [0,1]$$



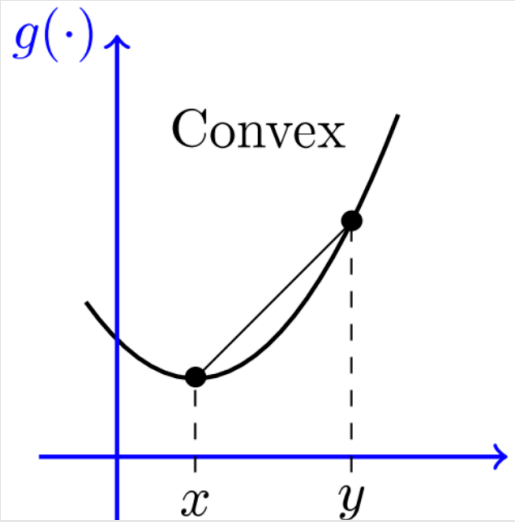
A function is strictly convex if $f(\lambda x_1 + (1-\lambda)x_2) < \lambda f(x_1) + (1-\lambda)f(x_2) \quad \lambda \in [0,1]$

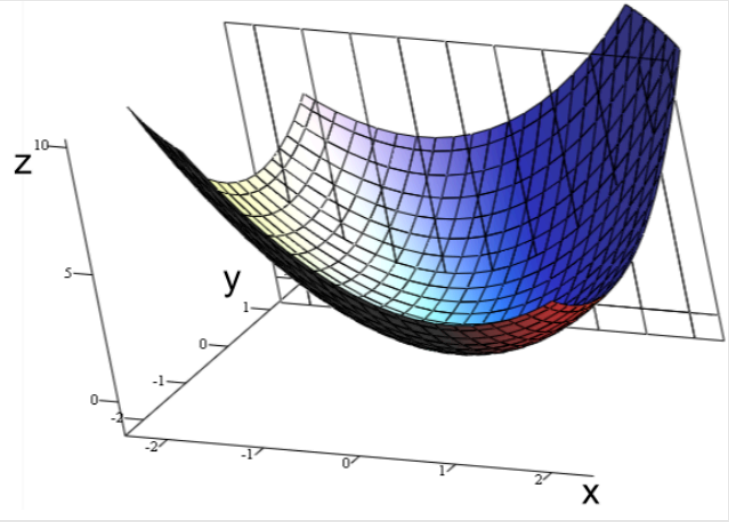
Convex function

Function f is concave if $-f$ is convex



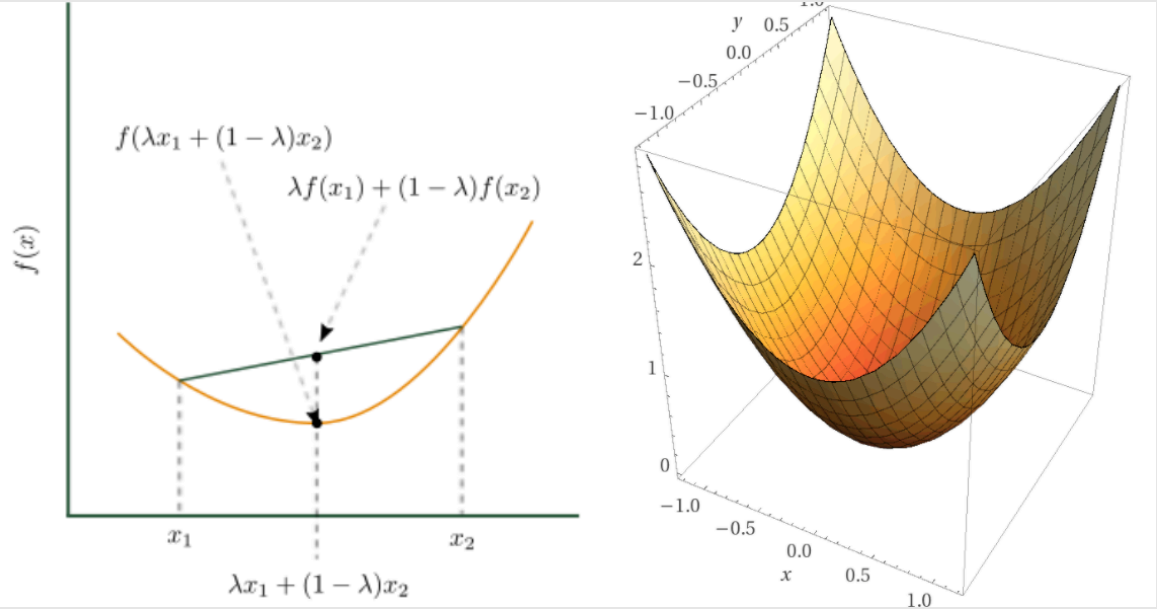
Convex functions





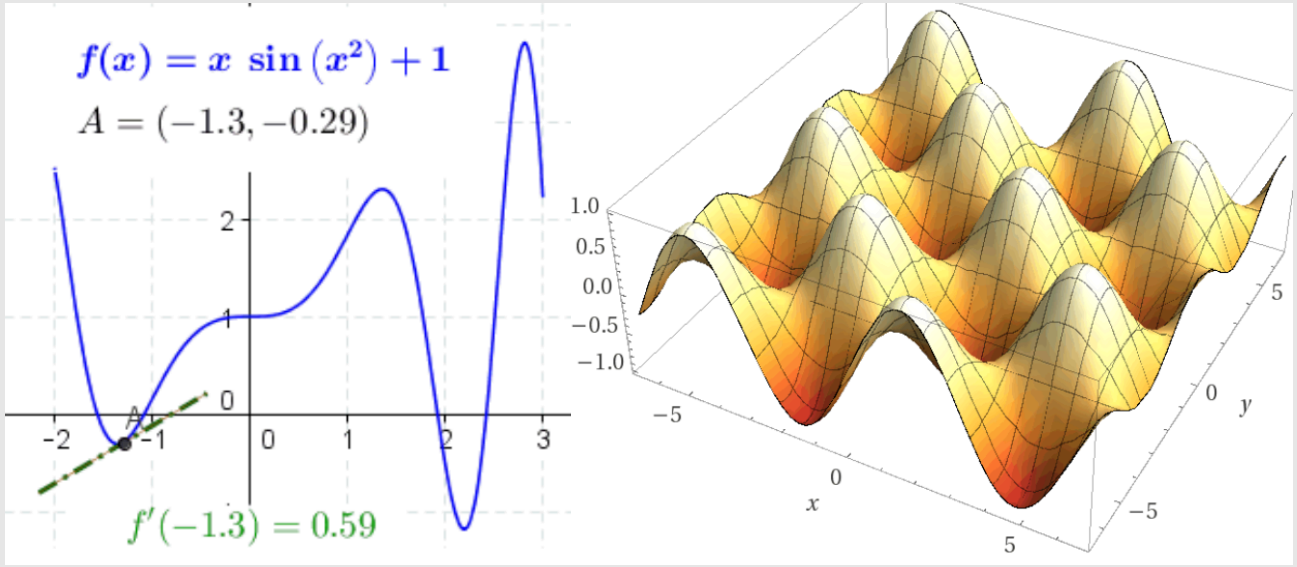
Convexity = Convexity along all lines

Why is convexity so important



If you can find a **critical point** in a convex function, it's a **global minimum**.
Critical point: $f'(x) = 0$ or $f'(x)$ does not exist.

Why is convexity so important



If you can find a **critical point** in a convex function, it's a **global minimum**.

3 ways to check convexity for 1D function

- 0^{th} order: prove the original convexity condition

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$$

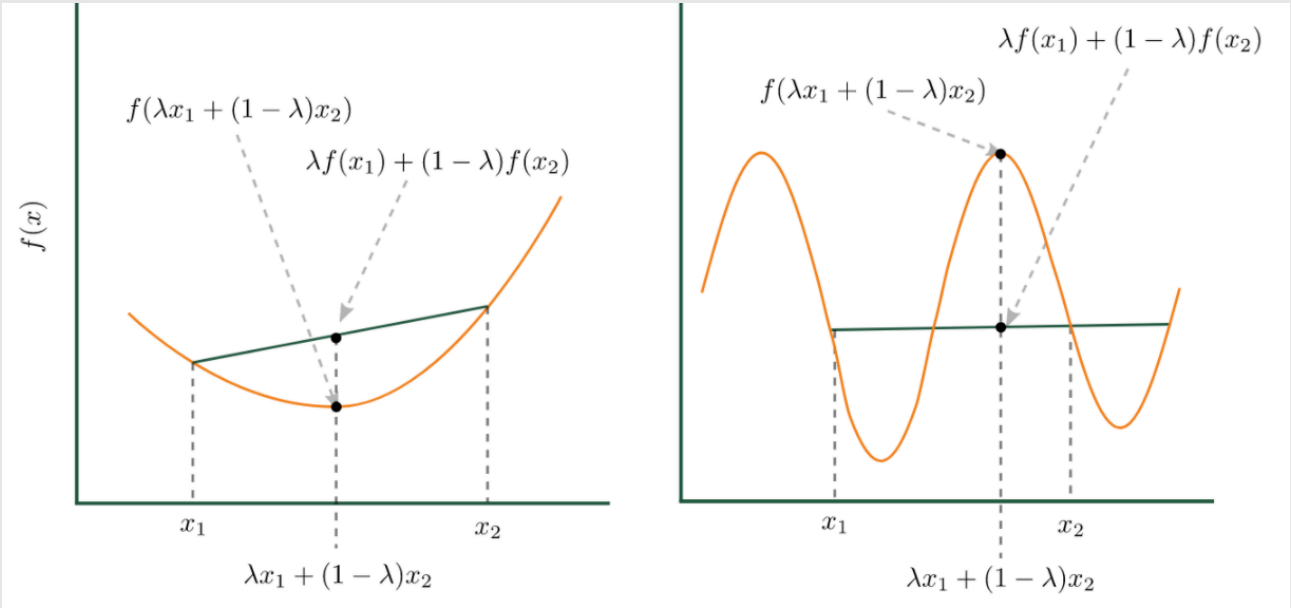
- 1^{st} order: tangent line must lie below the function

$$f(x_2) \geq f(x_1) + \frac{d}{dx} f(x_1)(x_2 - x_1)$$

- 2^{nd} order: curvature is always positive

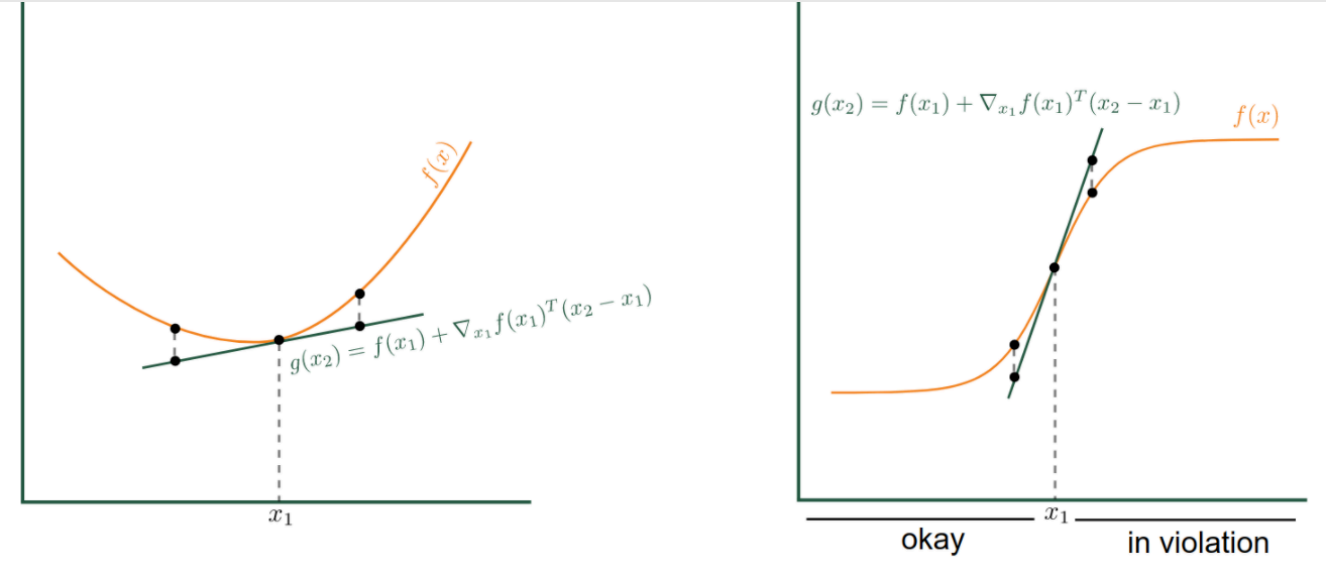
$$\frac{d^2}{dx^2} f(x) \geq 0$$

Checking convexity: 0th order



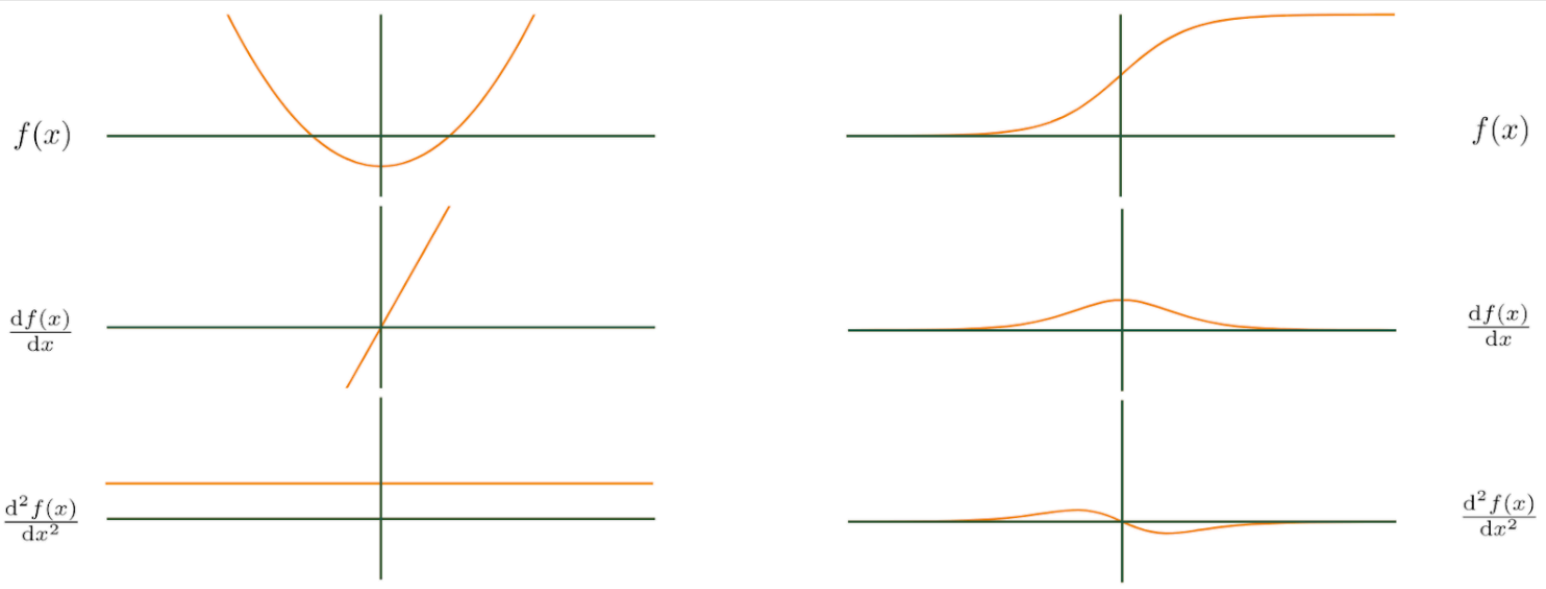
Prove $f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2)$

Checking convexity: 1st order



Confirm $f(x_2) \geq f(x_1) + \frac{d}{dx} f(x_1)(x_2 - x_1)$

Checking convexity: 2nd order



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

3 ways to check convexity for N-D function

- 0^{th} order: prove the original convexity condition

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$$

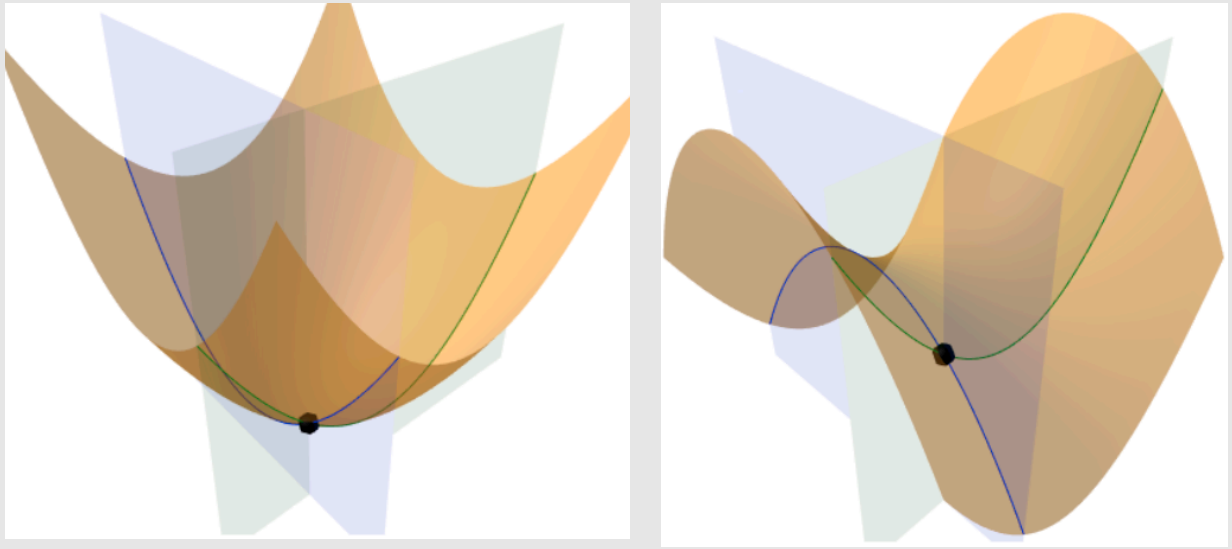
- 1^{st} order: tangent plane must lie below the function

$$f(x_2) \geq f(x_1) + \nabla_x f(x_1)^T \, (x_2 - x_1)$$

- 2^{nd} order: Hessian is positive semi-definite

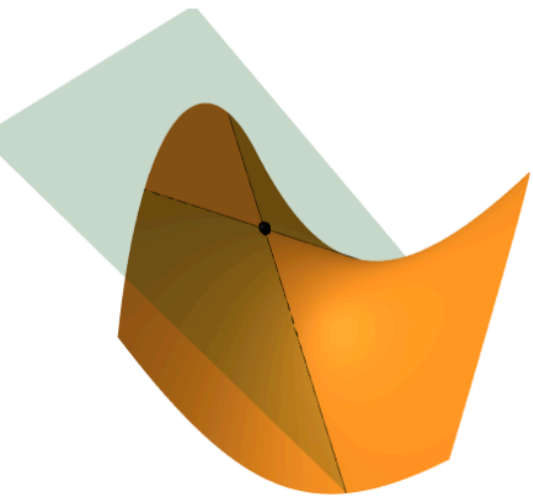
$$\nabla_x^2 f(x) \geq 0$$

Checking convexity: 0^{th} order



Convexity in each direction

Checking convexity: 1st order



Tangent plane is the span of tangents $f(x_2) \geq f(x_1) + \nabla_x f(x_1)^T (x_2 - x_1)$

Checking convexity: 2nd order

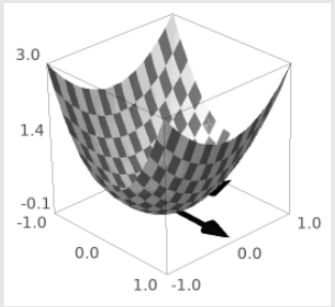
Hessian Matrix positive semi-definite: $\nabla_x^2 f(x) \geq 0$

- Matrix A is positive definite if $x^T Ax \geq 0 \; for \; \forall x \in R^n$
- A symmetric matrix is p.s.d. if and only if all eigenvalues are non-negative.

$$\nabla^2_{\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\, \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\, \partial x_n} \\ \frac{\partial^2 f}{\partial x_2\, \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\, \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n\, \partial x_1} & \frac{\partial^2 f}{\partial x_n\, \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

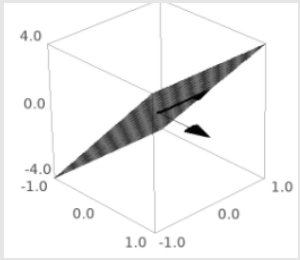
Hessian

$f(x) = 2x_1^2 + x_2^2$



x_1

$\frac{\partial}{\partial x_1} f(x)$

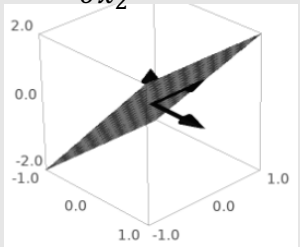


x_1 → $\frac{\partial^2}{\partial x_1^2} f(x) = 4$

x_2 → $\frac{\partial^2}{\partial x_1 x_2} f(x) = 0$

x_2

$\frac{\partial}{\partial x_2} f(x)$



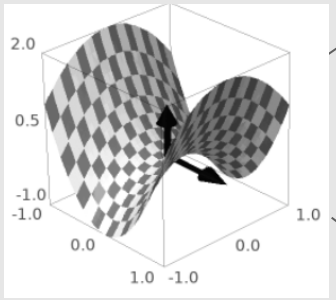
x_1 → $\frac{\partial^2}{\partial x_1 x_2} f(x) = 0$

x_2 → $\frac{\partial^2}{\partial x_2^2} f(x) = 2$

$$H = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix}$$

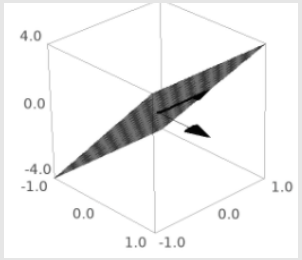
Hessian

$f(x) = 2x_1^2 - x_2^2$



x_1

$\frac{\partial}{\partial x_1} f(x)$

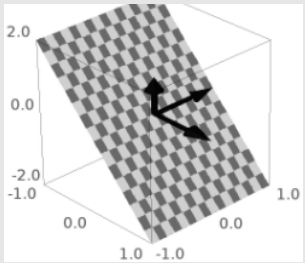


x_1 → $\frac{\partial^2}{\partial x_1^2} f(x) = 4$

x_2 → $\frac{\partial^2}{\partial x_1 x_2} f(x) = 0$

x_2

$\frac{\partial}{\partial x_2} f(x)$



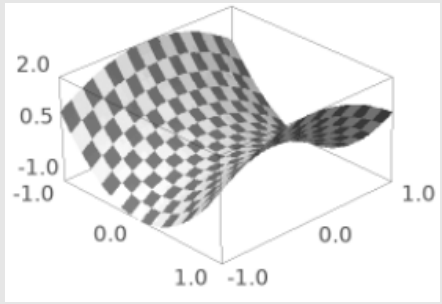
x_1 → $\frac{\partial^2}{\partial x_1 x_2} f(x) = 0$

x_2 → $\frac{\partial^2}{\partial x_2^2} f(x) = -2$

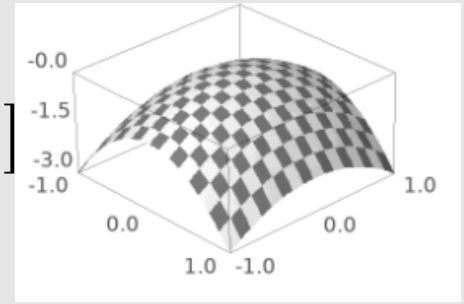
$$H = \begin{bmatrix} 4 & 0 \\ 0 & -2 \end{bmatrix}$$

Checking convexity: 2nd order

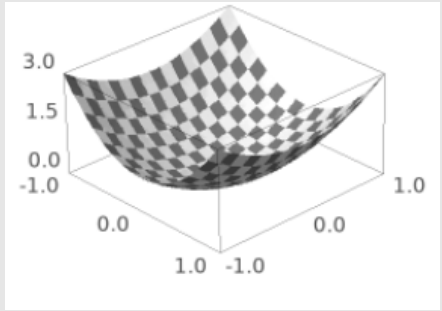
$$H = \begin{bmatrix} 4 & 0 \\ 0 & -2 \end{bmatrix}$$



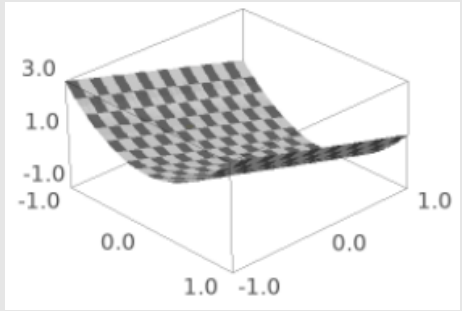
$$H = \begin{bmatrix} -4 & 0 \\ 0 & -2 \end{bmatrix}$$



$$H = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix}$$



$$H = \begin{bmatrix} 4 & 0 \\ 0 & 0 \end{bmatrix}$$



The diagonals contain all you need. What if it's not diagonal?

Checking convexity: 2nd order

$$f(x) = x_1 + 2x_1^2 + x_2^2 - x_1x_2 - 2$$

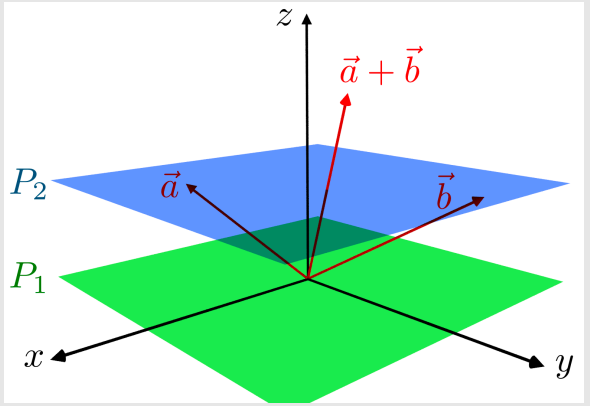
$$H = \begin{bmatrix} 4 & -1 \\ -1 & 2 \end{bmatrix}$$

Eigenvalues: $\{\lambda\} = \left[-\sqrt{2}+3, \sqrt{2}+3\right]$

Eigenvectors: $\{e\} = \left\{\begin{pmatrix} 1 \\ \sqrt{2}+1 \end{pmatrix}, \begin{pmatrix} 1 \\ -\sqrt{2}+1 \end{pmatrix}\right\}$

Example of convex functions

- Exponential: $f(x) = e^{ax}$ for $\forall a \in R$
- Affine functions: $f(x) = a^T x + b$ where $x \in R^n$



- Quadratic functions: $f(x) = \frac{1}{2} x^T A x + b^T x + c$ for a symmetric matrix $A \in R^{n \times n}$ and $b \in R^n$ and $c \in R$
- Nonnegative weighted sums of convex functions $f(x) = \sum_{i=1}^{K} w_i f_i(x)$
  - Where $f_1, f_2, \ldots f_K$ are convex functions and $w_1, w_2, \ldots w_K$ are nonnegative real numbers

Affine functions are both convex and concave

Convex optimization

Convex optimization problems:

$$\text{Minimize } f(x)$$
$$\text{subject to } x \in C$$

Where C is a convex set.

We can often write it as

$$\text{Miminize } f(x) \quad \text{objective}$$
$$\text{s.t. } g_i(x) \leq 0, i = 1, \ldots, m \quad \text{constraints}$$
$$h_j(x) = 0, j = 1, \ldots, p$$

Where f is a convex function, g_i's are convex functions, and h_j's are affine functions

Solving unconstrained convex optimization

How to solve convex optimization problems

Assume no constraints, then the problem

$\arg\min_x \ f(x)$, where $f(x)$ is a convex function

Is equivalent to

$$\nabla f(x) = 0$$

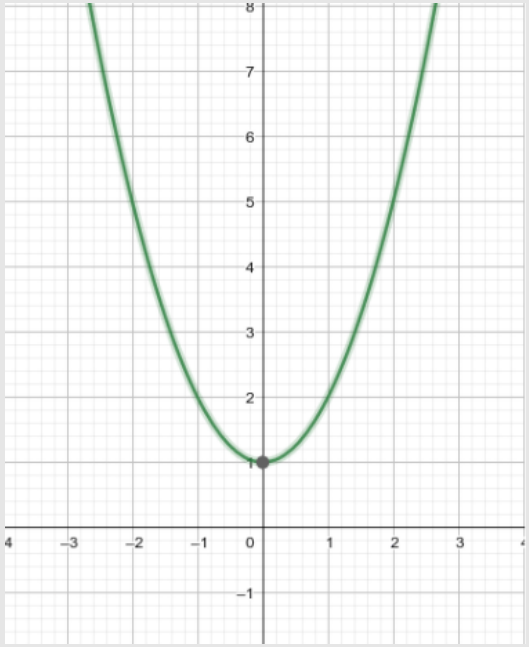
$$\nabla f(x) = \left(\frac{\partial}{x_0} f(x), \frac{\partial}{x_1} f(x), \ldots \frac{\partial}{x_p} f(x)\right)$$

Example

Objective: $f(x) = x^2 + 1$

Solve: $\frac{df(x)}{dx} = 2x = 0$

$x = 0$



Example in 2D

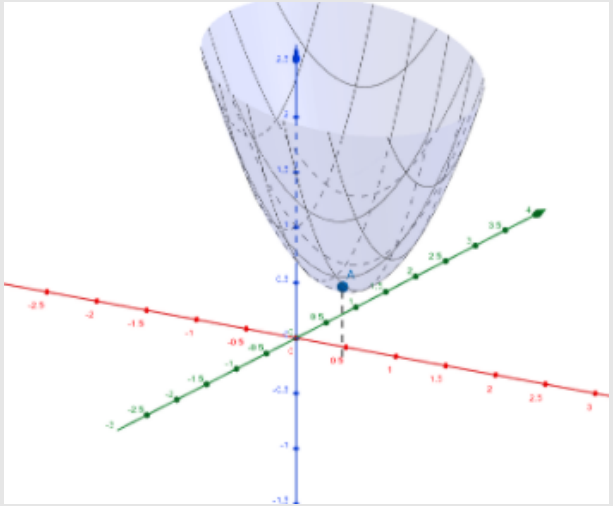
Objective: $f(x) = x_1^2 + x_2^2 + x_1 x_2 - x_1 + 1$

Solve:

$$\frac{\partial f(x)}{dx_1} = 2x_1 + x_2 - 1$$

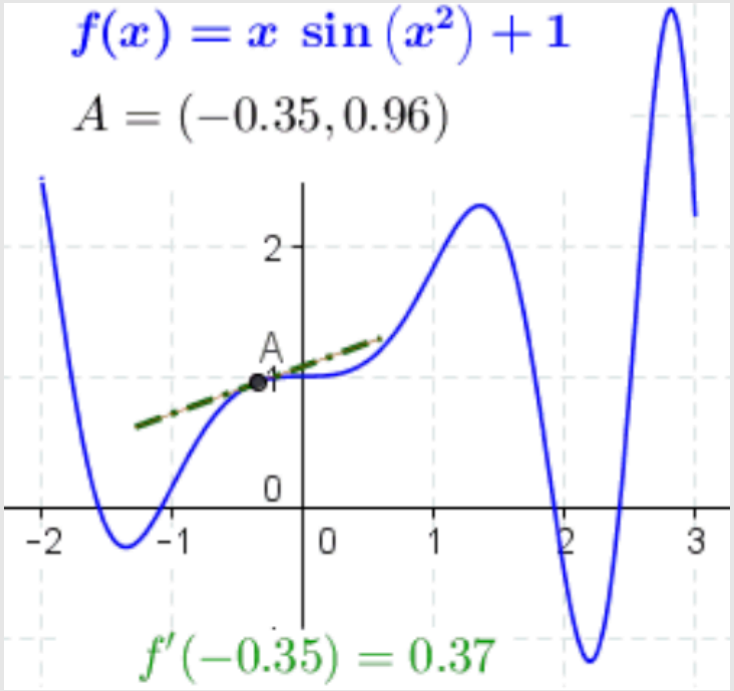
$$\frac{\partial f(x)}{dx_2} = 2x_2 + x_1$$

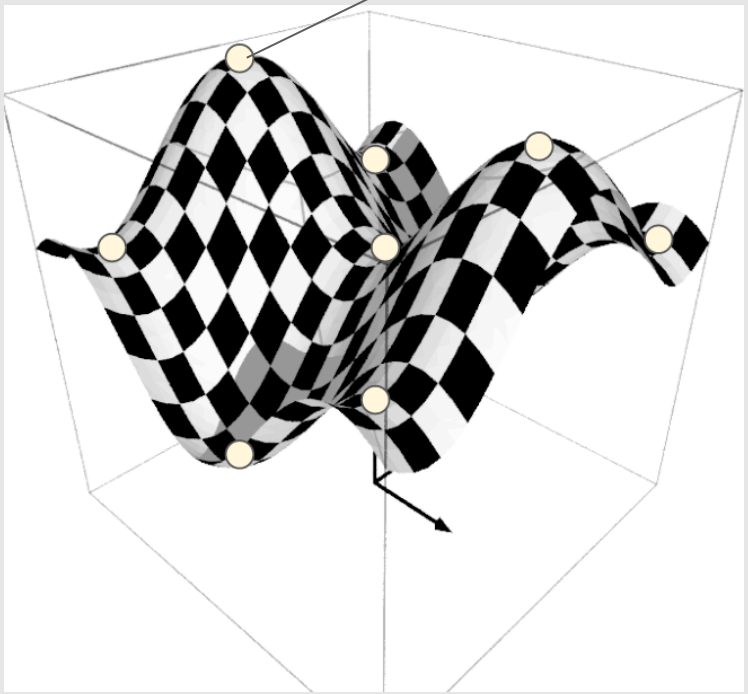
$$\nabla f(x) = \left(\frac{\partial}{x_1} f(x), \frac{\partial}{x_2} f(x)\right) = (0,0)$$



Caveat: saddle points

Check the Hessian. If Hessian is p.d./p.s.d. for x, then it's local minimum/maximum



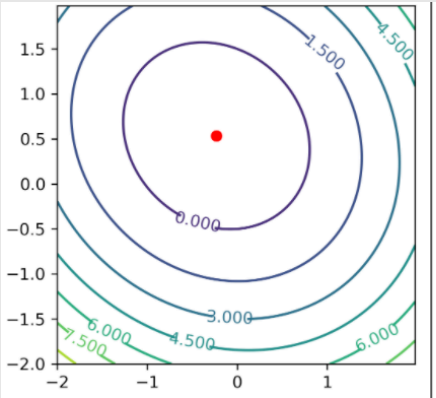


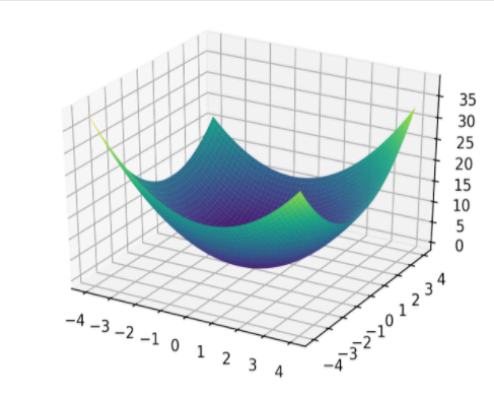
Higher dimensional: use scipy

Optimization using Scipy

```python
import numpy as np
import scipy.optimize as opt


#f0(x)
def f(x):
    return x[0]**2+(x[1]-.5)**2-1. + 0.3*x[0]*x[1] + 0.3*x[0]


# solve using scipy optimization toolbox
sol = opt.minimize(f,x0=np.random.rand(2))


# yields [-0.23, 0.53]
sol.x
```

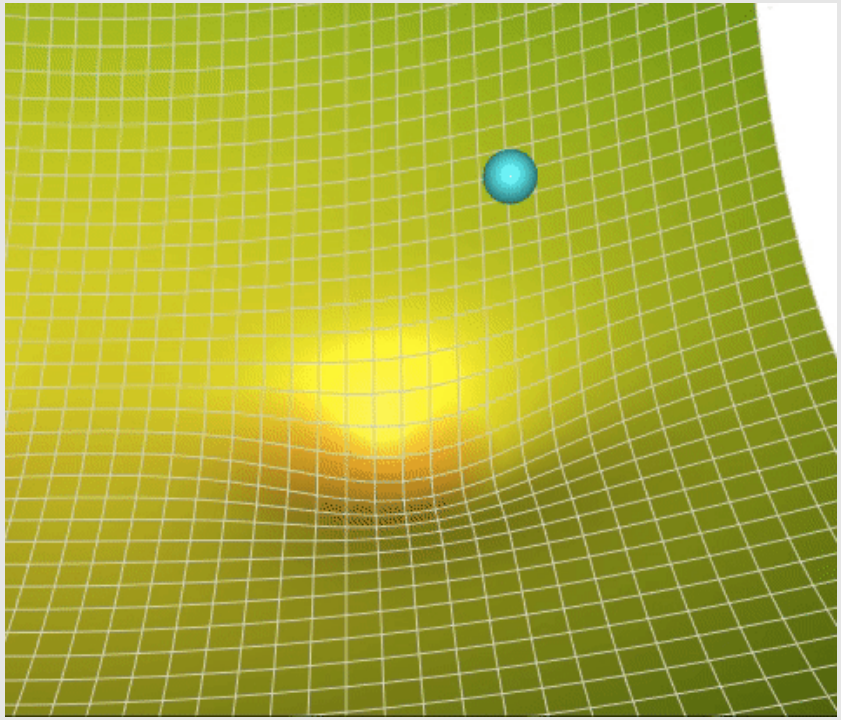
$$f(x) = x_0^2 + (x_1 - 0.5)^2 - 1 + 0.3x_0x_1 + 0.3x_0$$



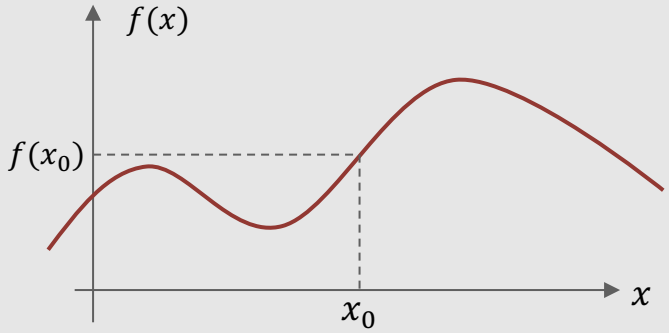


Gradient Descent

Gradient Descent



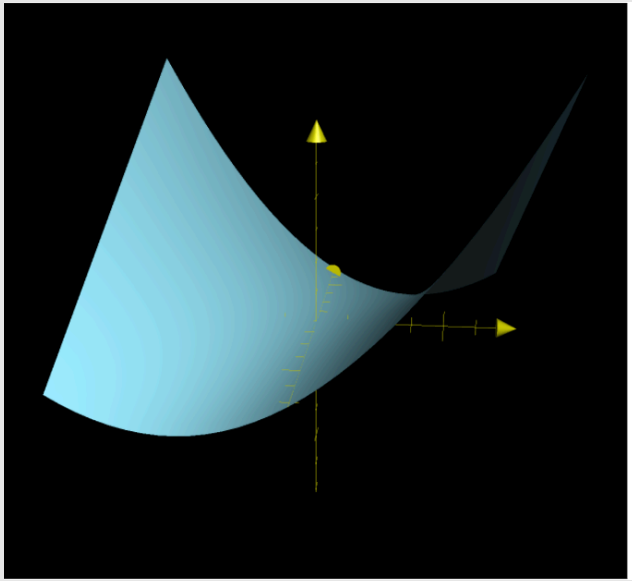
1-D Optimization

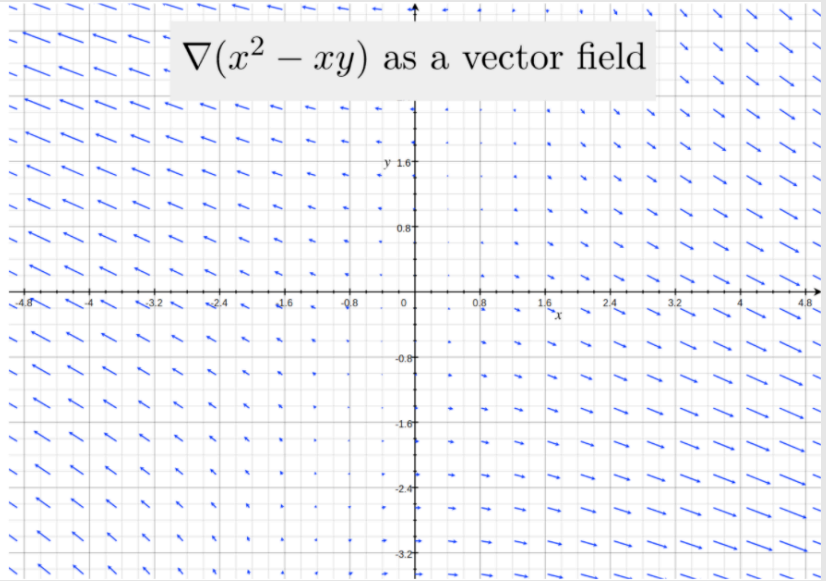


- Could evaluate $f(x_0 + h)$ and $f(x_0 - h)$, then step toward the best direction
- Or equivalently, evaluate the derivative which tells the best direction

$$\frac{d}{x} f(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0 - h)}{2h}$$

Gradient



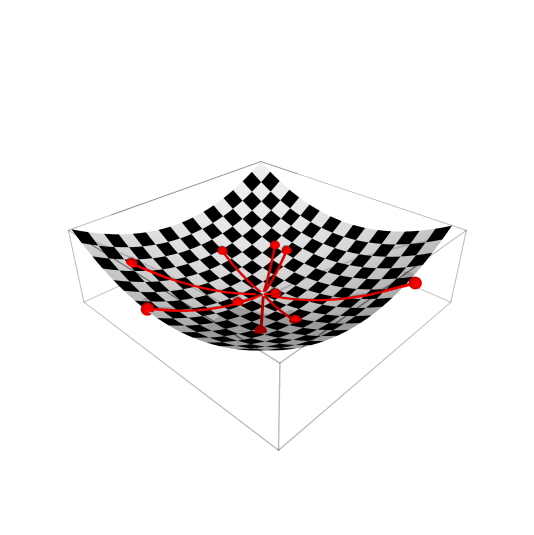


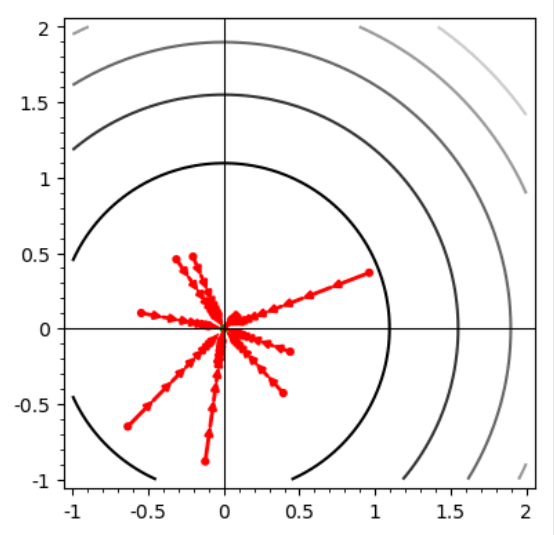
Each arrow is the gradient evaluated at that point. The gradients point towards the direction of steepest ascent.

Gradient Descent Algorithm

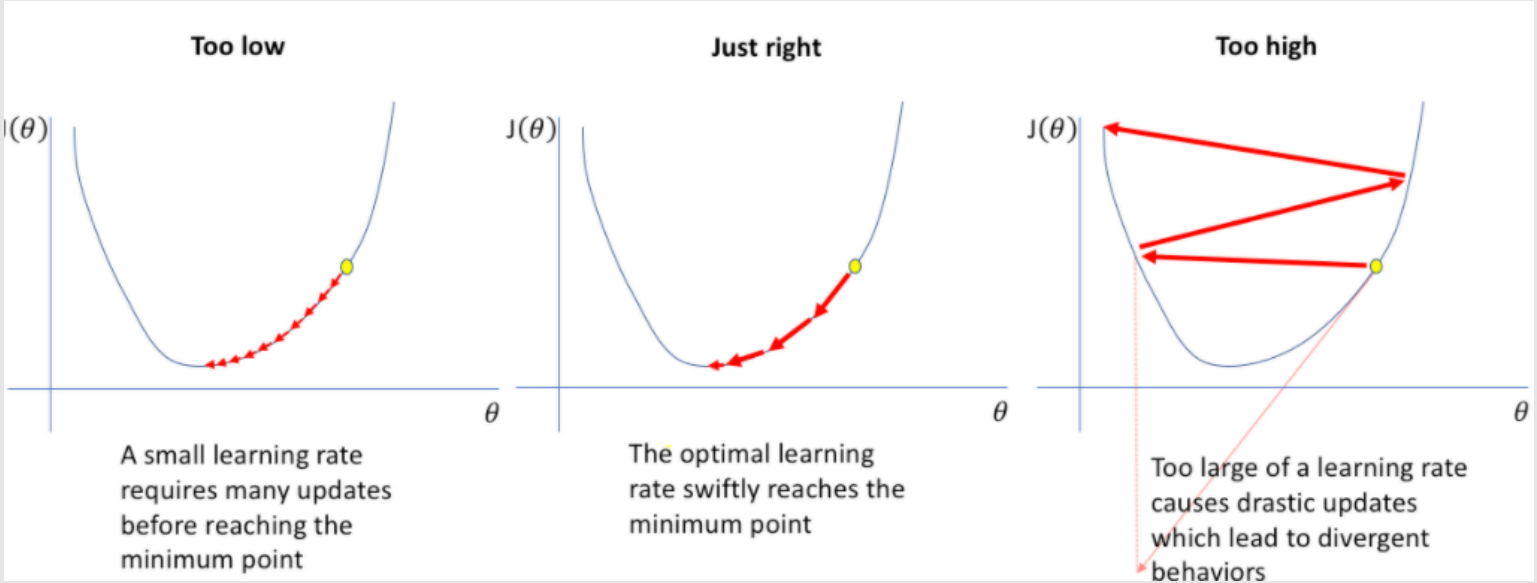
- Choose initial guess x_0
- While not converged

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

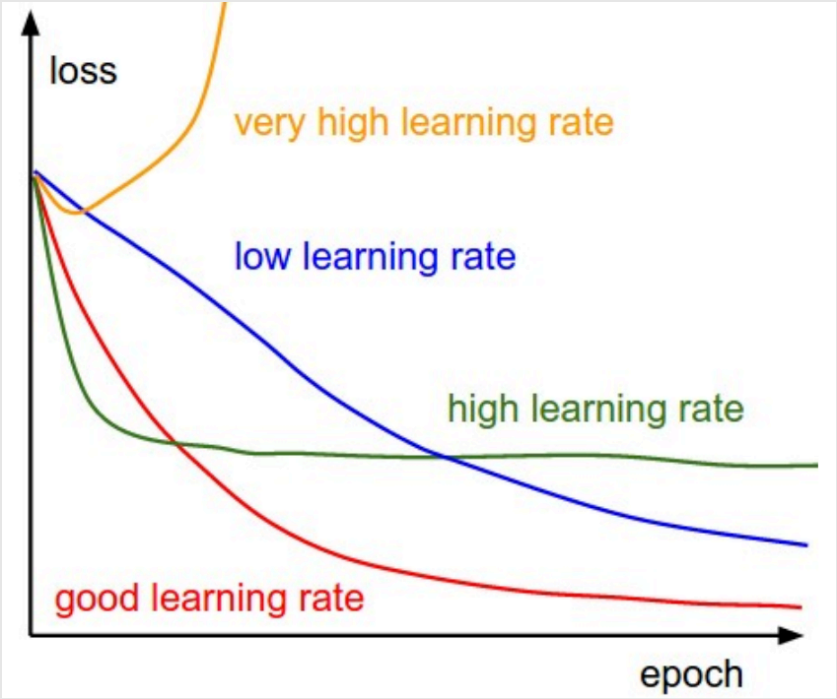




Choice of learning rate



Debugging gradient descent



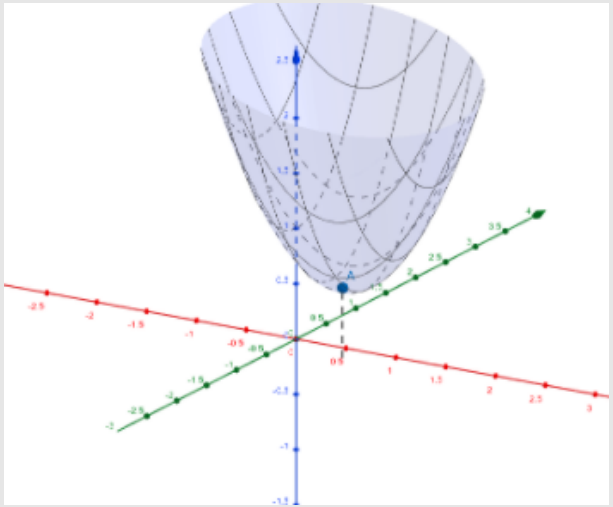
Example in 2D (revisited)

Objective: $f(x) = x_1^2 + x_2^2 + x_1 x_2 - x_1 + 1$

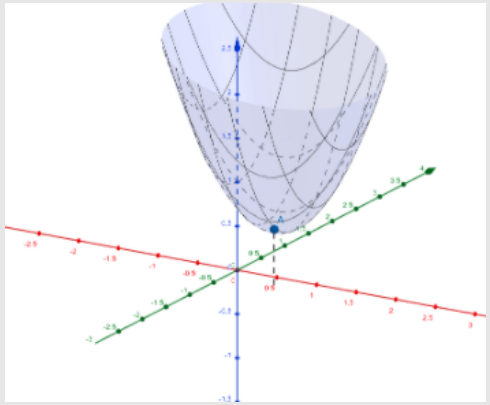
Solve: $\frac{\partial f(x)}{\partial x_1} = 2x_1 + x_2 - 1$

$\frac{\partial f(x)}{\partial x_2} = 2x_2 + x_1$

$\nabla f(x) = \left(\frac{\partial}{\partial x_1} f(x), \frac{\partial}{\partial x_2} f(x) \right)$



Example in 2D (revisited)



Algorithm:

$$\nabla f(x) = \begin{pmatrix} 2x_1 + x_2 - 1 \\ 2x_2 + x_1 \end{pmatrix}$$

$$x_{t+1} \leftarrow x_t - \alpha \nabla f(x)$$

Try it for $\alpha = 0.3$

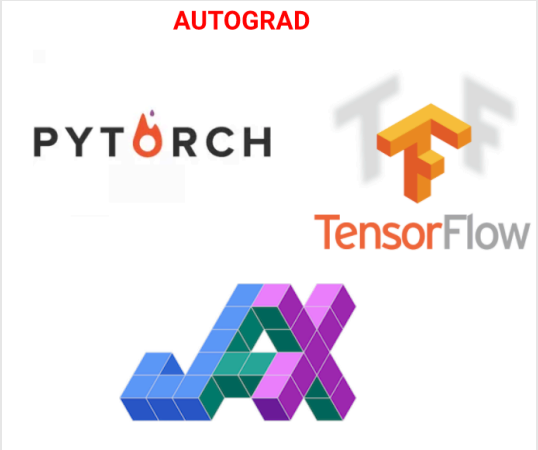
Iteration t	$\mathbf{x}_t = [\, x_{1,t}, x_{2,t} \,]$	**Gradient**
0	[3, 2]	[7,7]
1	[0.9, -0.1]	[0.7,0.7]
2	[0.69,-0.31]	[0.07,0.07]
3	[⅔,-⅓]	

Special cases

What if we can't even calculate gradient?

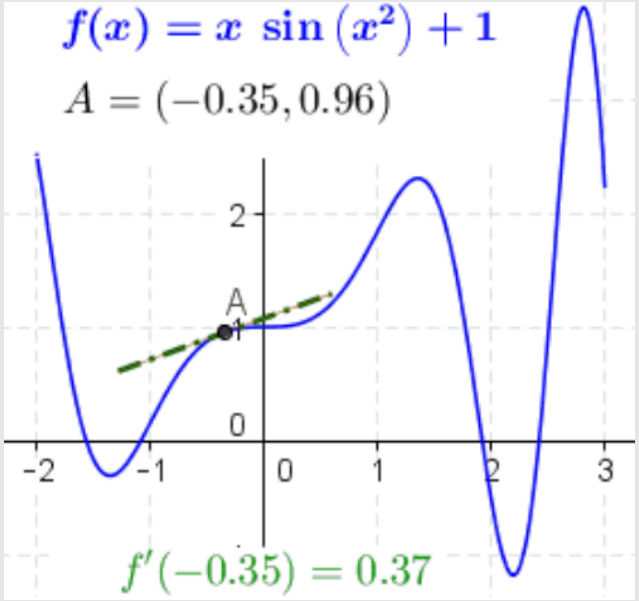
$$\nabla f(x) \approx \frac{f(x) - f(x-h)}{h}$$

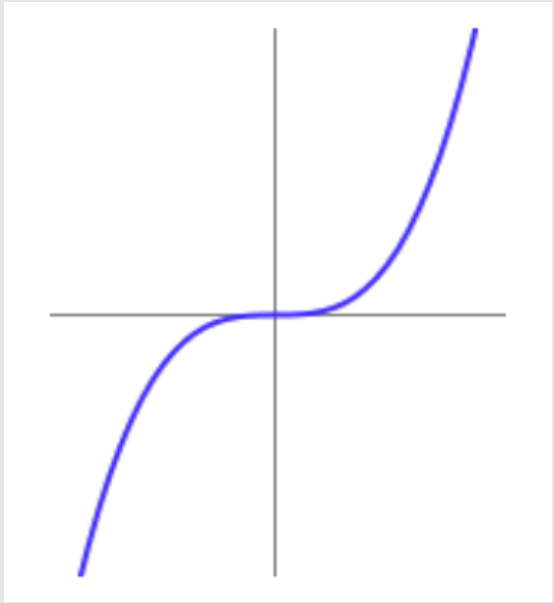
$$\nabla f(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$



This is called **Tangent Line Approximation** and is a direct result of the Taylor expansion.

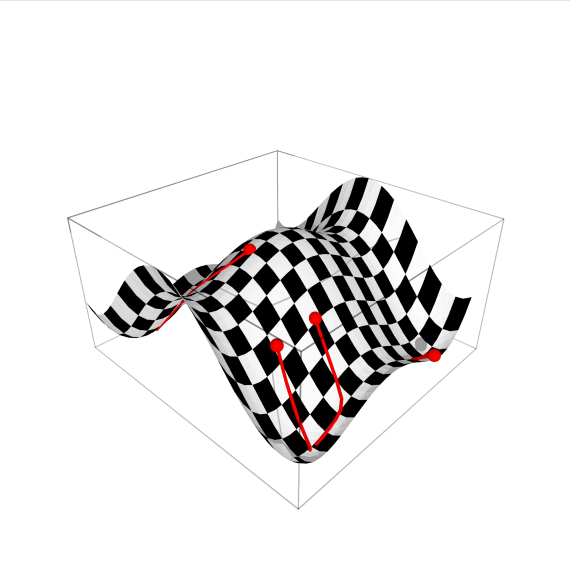
Caveat: Perils of non-convex functions

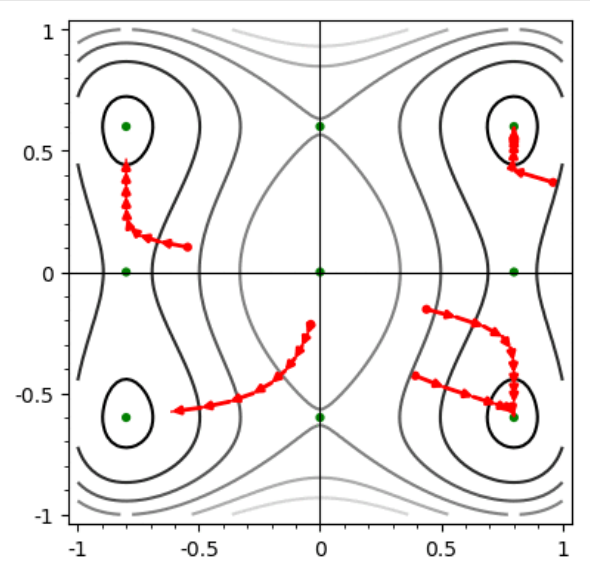




Non convex functions can lead to local minima or singularities

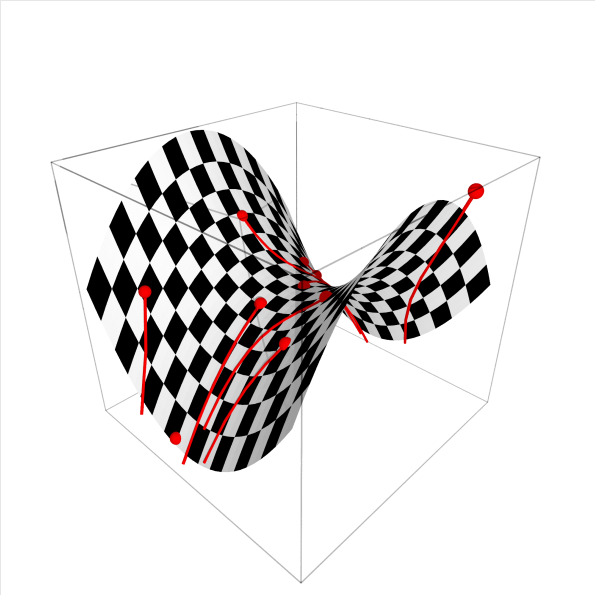
Caveat: local minima

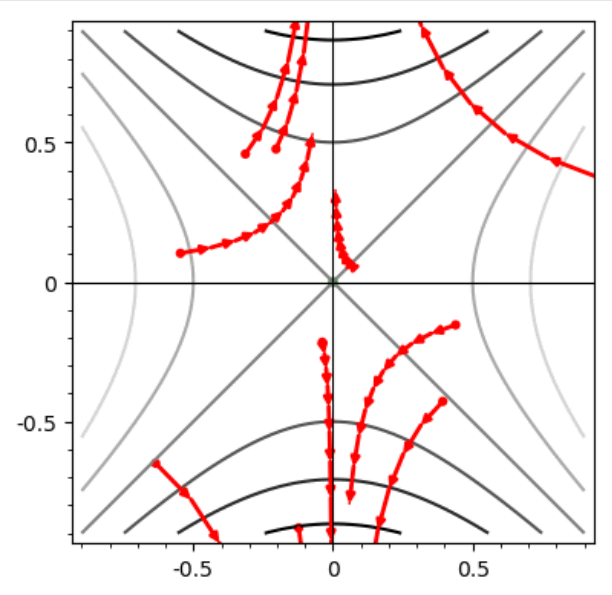




Non convex functions can lead to local minima

Caveat: singularities





Example

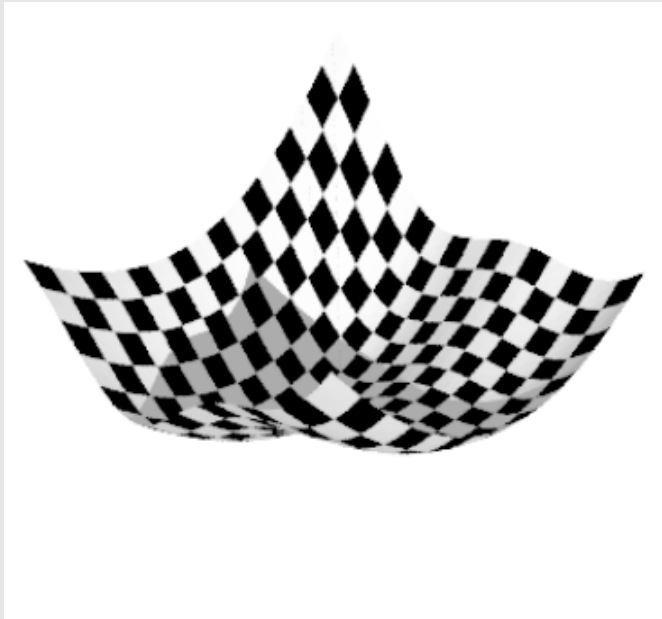
Find the minimum of function $g(u, v) = (u + 2v^2 + 10\sin(u) + 2u^2 - uv - 2)/30$

```python
import numpy as np
import scipy.optimize as opt


#f0(x)
def f(x):
    return ?


# solve using scipy optimization toolbox
sol = opt.minimize(f,x0=np.random.rand(2))


# yields [-0.23, 0.53]
sol.x
```

Example

Find the minimum of function $g(u,v) = (u + 2v^2 + 10\sin(u) + 2u^2 - uv - 2)/30$

```python
import numpy as np
import scipy.optimize as opt

#f0(x)
def g(x):
    u,v = x
    return (u + 2*v**2 + 10*np.sin(u)+2*u**2 - u*v - 2 )/30

# solve using scipy optimization toolbox
sol = opt.minimize(g,x0=np.random.rand(2))

# yields [-1.20944045, -0.30236448]
sol.x
```



Example

Find the minimum of function $g(u,v) = (u + 2v^2 + 10\sin(u) + 2u^2 - uv - 2)/30$

```python
import numpy as np
# import autograd
from autograd import elementwise_grad as egrad
# for special functions, you have to import autograd's
from autograd.numpy import sin

# f0(x)
def g(x):
    u,v = x
    return (u + 2*v**2 + 10*sin(u)+2*u**2 - u*v - 2 )/30
# gradient descent
alpha = 2
x     = np.array([1.,1.])
for i in range(0,20):
    grad =  egrad(g)(x)
    x    = x - alpha * grad
    print(i,"\t\t",x,"\t\t",grad)
```