Numerical Analysis for Artificial Intelligence, Week 2

UCSD Summer session II 2018

CSE 190

Jacek Cyranka

Derivatives 1 (univariate)

What is the derivative ?

Measures the rate of change of a function
(negative when function is decreasing
and positive when function is increasing)

Mathematica 1d gradient presentation

Numerical vs symbolic derivatives

Derivative of function f

Derivative symbol

at x

$$f'(x) = \frac{df}{dx}(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

with respect to x

Limit as h approaches 0

`Numerical derivative`

Estimate of the rate change of f for fixed (small) h

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

Numerical vs symbolic derivatives

Computation of derivatives symbolically using the rules of differentiation

- $\frac{d}{d(x)}(a) = 0$
- $\frac{d}{d(x)}(x) = 1$
- $\frac{d}{d(x)}(x^n) = n(x)^{n-1}$
- $\frac{d}{d(x)}[f(x) \pm g(x)] = f'(x) \pm g'(x)$
- $\frac{d}{d(x)}[c f(x)] = c f'(x)$
- $\frac{d}{d(x)}[f(x) \cdot g(x)] = f'(x) g(x) + g'(x) f(x)$ Product rule
- $\frac{d}{d(x)}\left[\frac{f(x)}{g(x)}\right] = \frac{f'(x) g(x) - g'(x) f(x)}{[g(x)]^2}$ Quotient rule
- $\frac{d}{d(x)} f[g(x)] = f'[g(x)] g'(x)$ Chain rule
- $\frac{d}{d(x)} f(x)^n = n f(x)^{n-1} \cdot f'(x)$ Power rule
- $\frac{d}{d(x)} f(kx + e) = k f'(kx + e)$
- $\frac{d}{d(x)} \ln[f(x)] = \frac{f'(x)}{f(x)}$

smtutor.com

$$[x^a]' = a \cdot x^{a-1},\ x \in \mathbb{R} \text{ for } a \in \mathbb{N},\ x \in \mathbb{R} - \{0\} \text{ for } a \in \mathbb{Z},$$
$$x \in \mathbb{R}^+ \text{ for } a \in \mathbb{R}.$$
$$[e^x]' = e^x,\ x \in \mathbb{R};$$
$$[a^x]' = \ln(a) a^x,\ x \in \mathbb{R}.$$
$$[\ln(x)]' = \tfrac{1}{x},\ x > 0;$$
$$[\log_a(x)]' = \tfrac{1}{\ln(a)}\,\tfrac{1}{x},\ x > 0.$$
$$[\sin(x)]' = \cos(x),\ x \in \mathbb{R};$$
$$[\cos(x)]' = -\sin(x),\ x \in \mathbb{R};$$
$$[\tan(x)]' = \tfrac{1}{\cos^2(x)},\ x \neq \tfrac{\pi}{2} + k\pi;$$
$$[\cot(x)]' = \tfrac{-1}{\sin^2(x)},\ x \neq k\pi.$$
$$[\arcsin(x)]' = \tfrac{1}{\sqrt{1-x^2}},\ x \in (-1,1);$$
$$[\arccos(x)]' = \tfrac{-1}{\sqrt{1-x^2}},\ x \in (-1,1);$$
$$[\arctan(x)]' = \tfrac{1}{x^2+1},\ x \in \mathbb{R};$$
$$[\mathrm{arccot}(x)]' = \tfrac{-1}{x^2+1},\ x \in \mathbb{R}.$$
$$[\sinh(x)]' = \cosh(x),\ x \in \mathbb{R};$$
$$[\cosh(x)]' = \sinh(x),\ x \in \mathbb{R};$$
$$[\tanh(x)]' = \tfrac{1}{\cosh^2(x)},\ x \in \mathbb{R};$$
$$[\coth(x)]' = \tfrac{-1}{\sinh^2(x)},\ x \neq 0.$$
$$[\mathrm{argsinh}(x)]' = \tfrac{1}{\sqrt{x^2+1}},\ x \in \mathbb{R};$$
$$[\mathrm{argcosh}(x)]' = \tfrac{1}{\sqrt{x^2-1}},\ x \in (1,\infty);$$
$$[\mathrm{argtanh}(x)]' = \tfrac{1}{1-x^2},\ x \in (-1,1);$$
$$[\mathrm{argcoth}(x)]' = \tfrac{1}{1-x^2},\ x \in (-\infty,-1) \cup (1,\infty).$$

See the comparison of numerical and symbolic derivatives in [week1_2.ipynb]

Chain rule

The rule for differentiating compositions of functions

inner function

$$\text{If } f = g(h) \text{ and } h = h(x), \text{ then } \frac{df}{dx} = \frac{dg}{dh} \times \frac{dh}{dx}$$

f is a composed function

outer function

Derivative of the outer function

Derivative of the inner function

Example for the chain rule in practice

$$f(x) = \arctan x^3$$

1. Decomposition of f into the outer (g) / inner (h) functions.

$$f(x) = g(h(x))$$

$g(h) = \arctan h$
$h(x) = x^3$

2. Differentiate g and h.

$$\frac{d}{dx}\arctan x = \frac{1}{1+x^2}$$

$$\frac{d}{dx}x^3 = 3x^2$$

$g(h) = \arctan h$	$g'(h) = \frac{1}{1+h^2}$
$h(x) = x^3$	$h'(x) = 3x^2$

3. Compose the final result

$$\frac{df}{dx} = \frac{dg}{dh} \times \frac{dh}{dx} = \frac{1}{1+h^2} \cdot 3x^2 = \frac{1}{1+x^6} \cdot 3x^2.$$

See the comparison of this derivative with symbolic in week1_2.ipynb

Partial derivatives and gradients

Partial derivative with respect to x

$$\frac{\partial f}{\partial x}(x, y)$$

When computing, treat y as a constant

Partial derivative with respect to y

$$\frac{\partial f}{\partial y}(x, y)$$

When computing, treat x as a constant

gradient $$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

[Mathematica 2d gradient presentation]

Computing partial derivatives analytically

$$f(x, y) = \sin xy$$

Compute $\frac{\partial f}{\partial x}$ applying the univariate chain rule (treating y as constant)
we have $f(x, y) = g(h), \quad h = h(x)$

$g(h) = \sin h$	$g'(h) = \cos h$
$h(x) = xy$	$h'(x) = y$

$$\frac{\partial f}{\partial x} = \frac{dg}{dh} \times \frac{dh}{dx} = \cos(xy) \cdot y.$$

And analogously

$$\frac{\partial f}{\partial y} = \frac{dg}{dh} \times \frac{dh}{dy} = \cos(xy) \cdot x.$$

Computing 2D gradients numerically

Estimate the rate of change of $f(x, y)$
for fixed (small) h in x and y direction

$$\frac{f(x+h, y) - f(x, y)}{h}$$

$$\frac{f(x, y+h) - f(x, y)}{h}$$

Those two quantities form a numerical gradient of $f(x, y)$

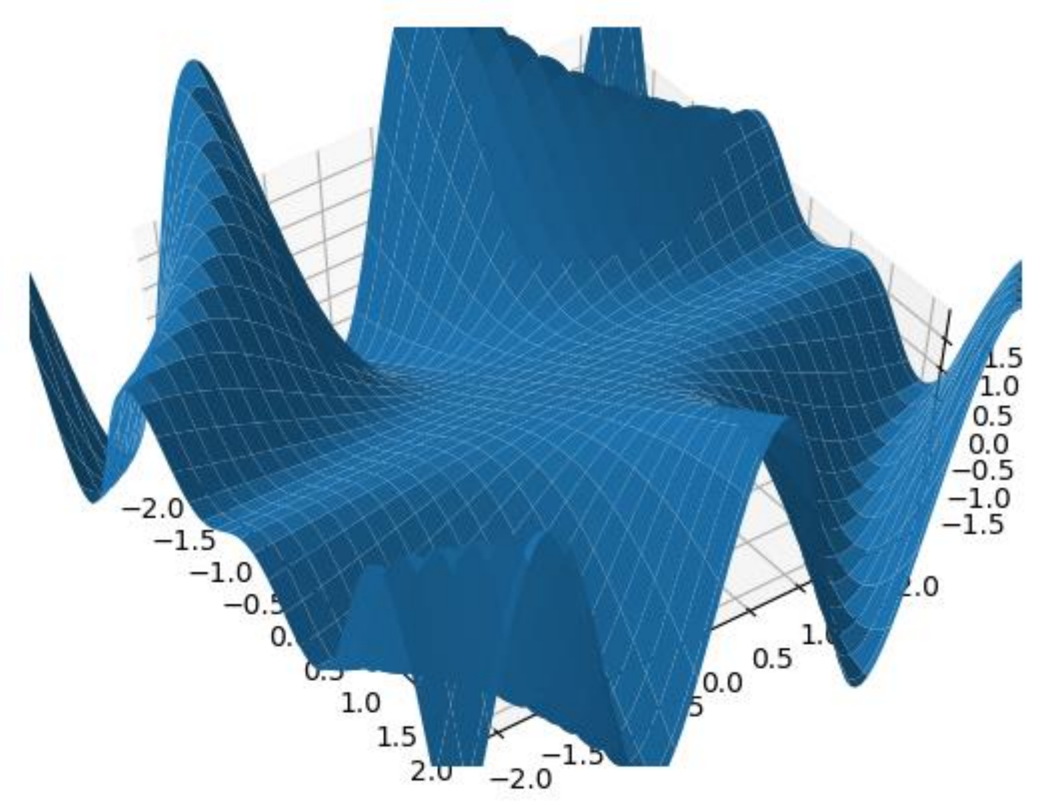
Differentiating the sigmoid function

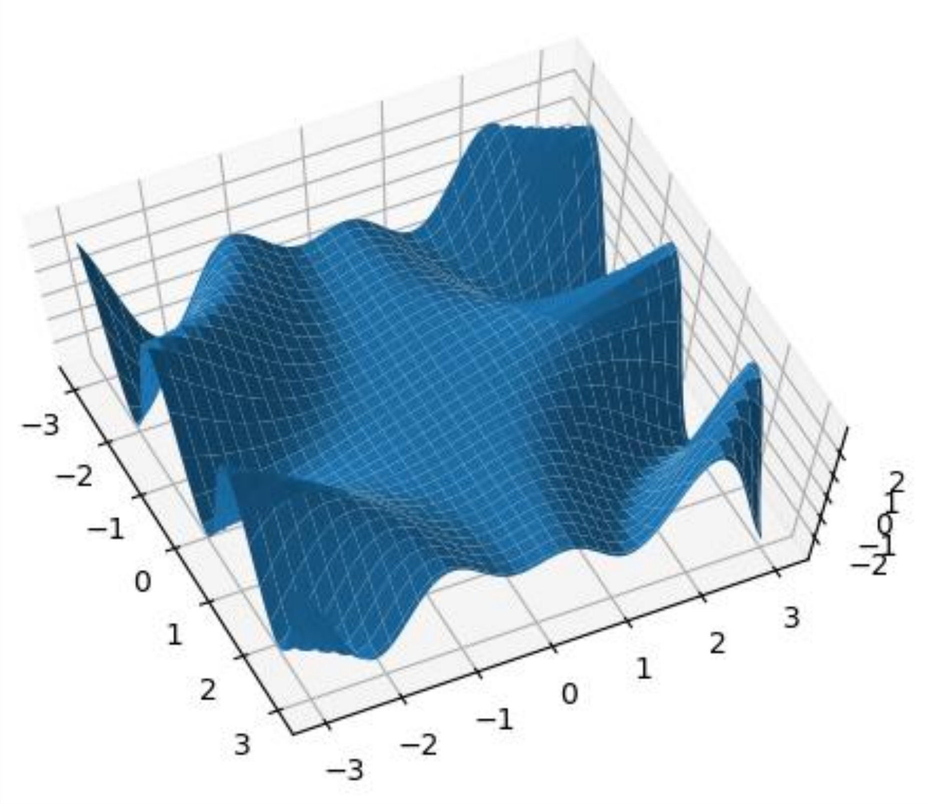
…Show on the blackboard…

Plotting gradient components

```
#import matplotlib 3d plotting
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(xsf, ysf, gradx)
plt.show()
```

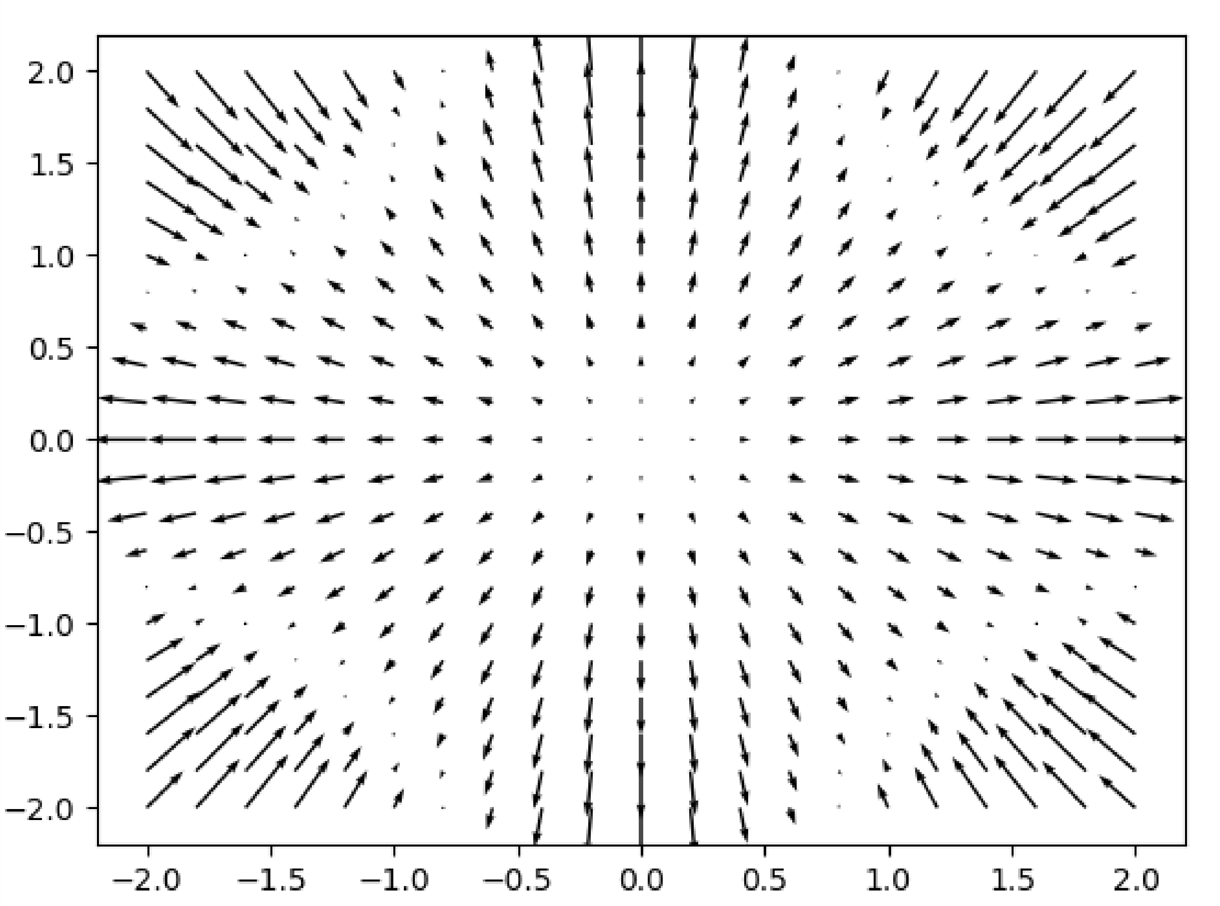




See 3d plotting in week2_1.ipynb

Plotting gradients as a vector field

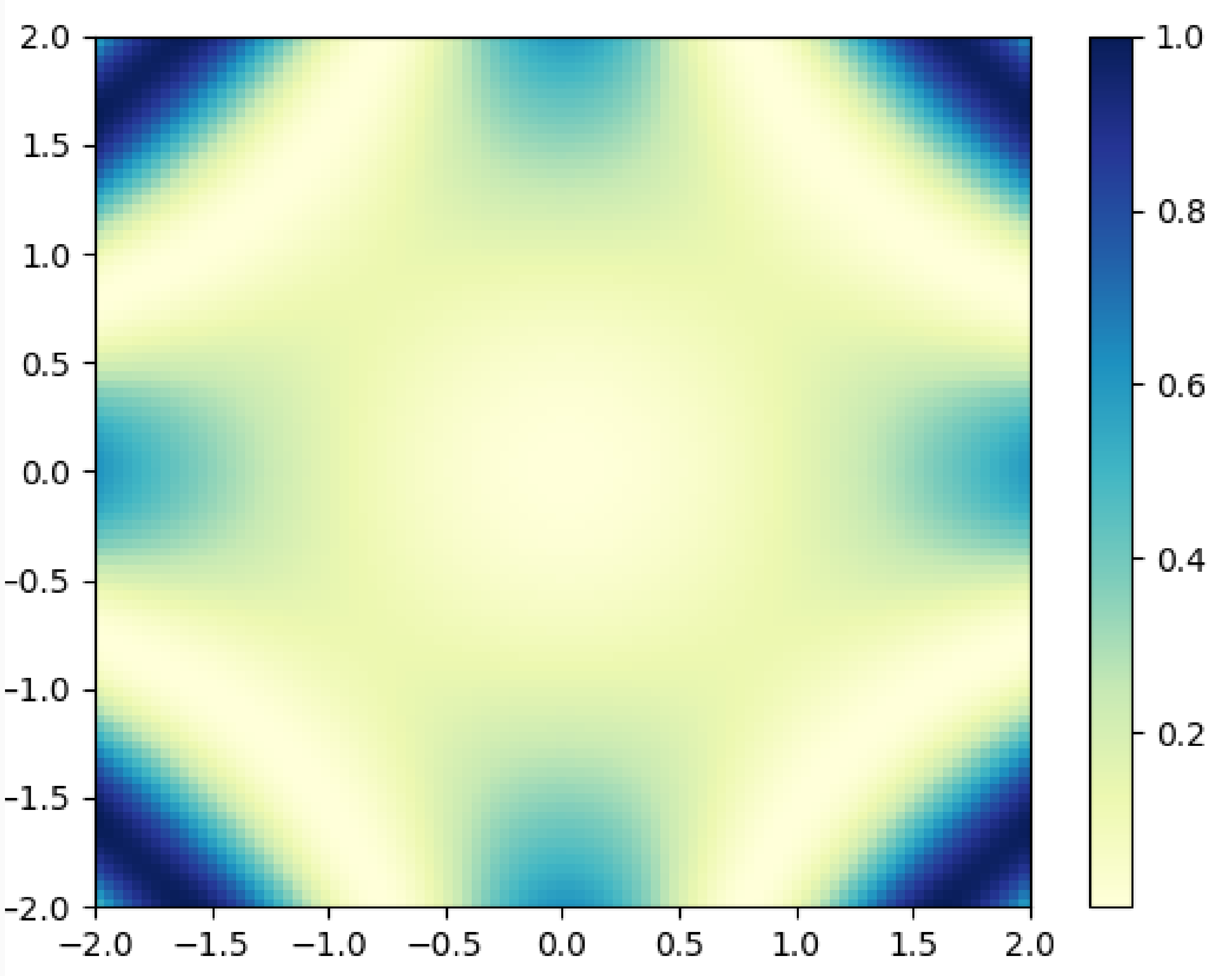
Two components of the gradient can be plotted on a single figure as a so-called **vector field**



See the gradient vector field plotting in [week2_1.ipynb]

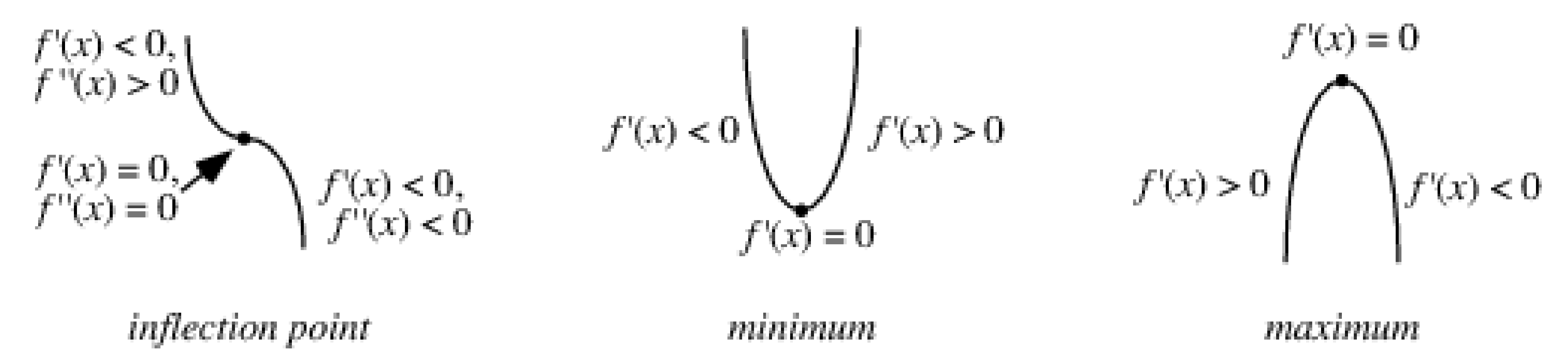
Yet another way of plotting the gradient

Heat map



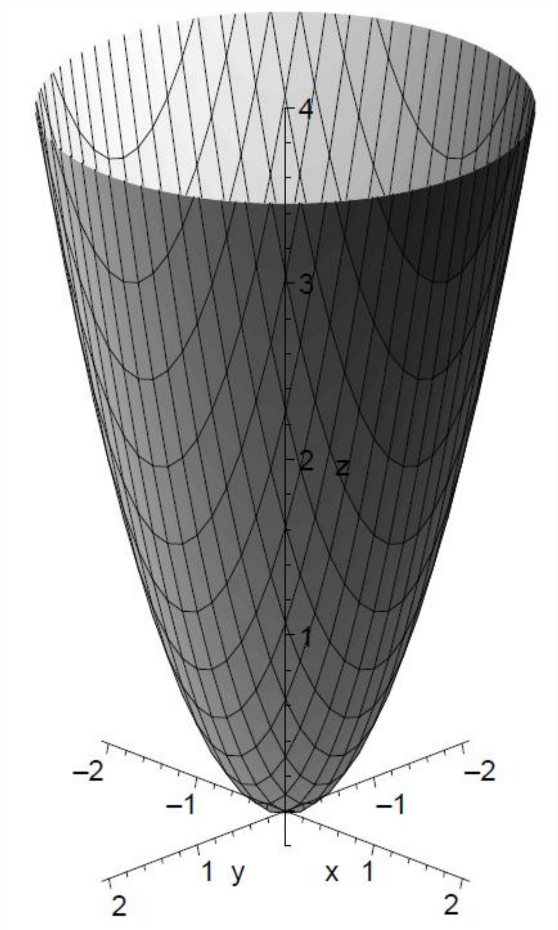
See the gradient heatmap plotting in [week2_1.ipynb]

Critical points in 1D

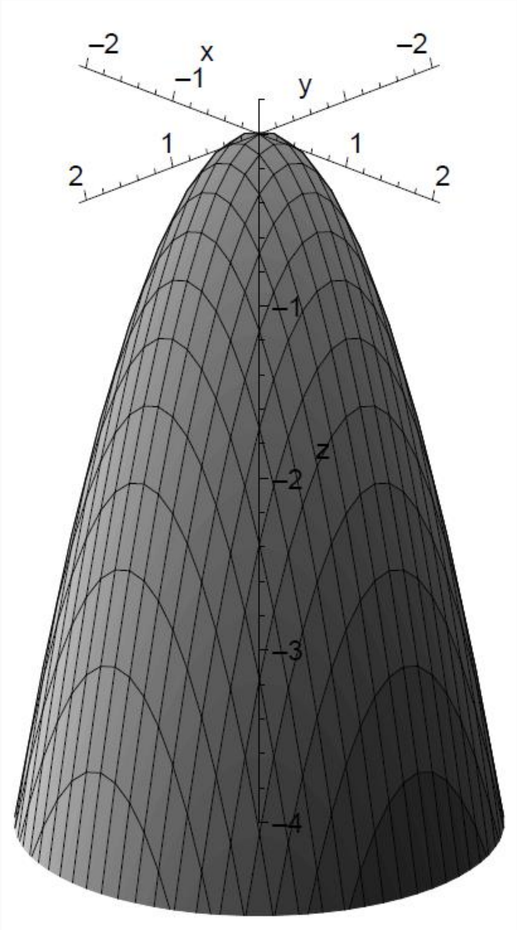


[Mathematica 1d derivative presentation]

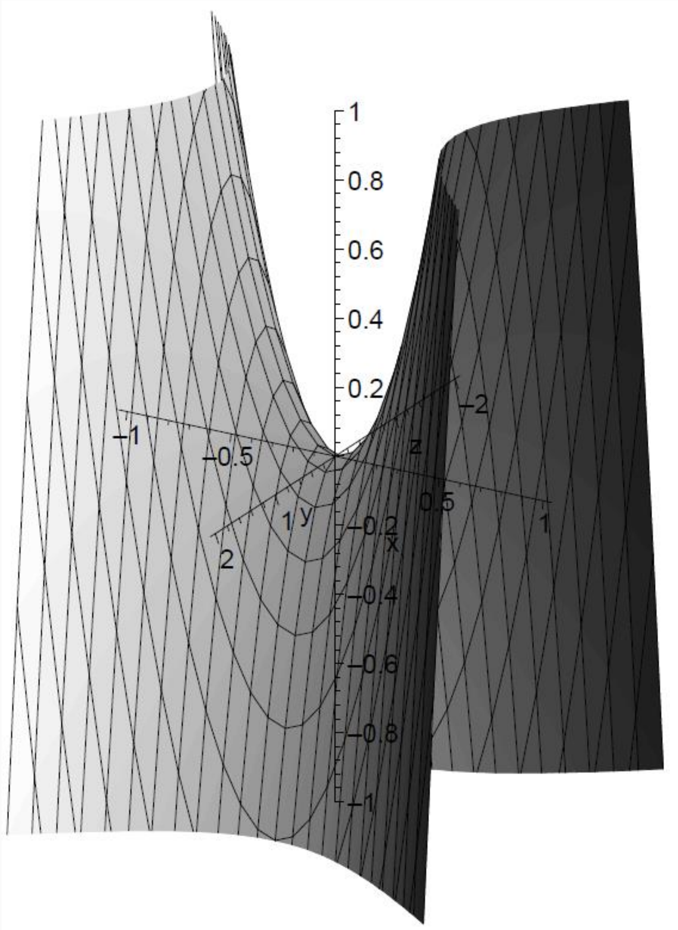
Critical points in multi-D



$f(x,y) = x^2 + 3y^2$



$f(x,y) = -(x^2 + y^2)$



$f(x,y) = -x^2 + 3y^2$

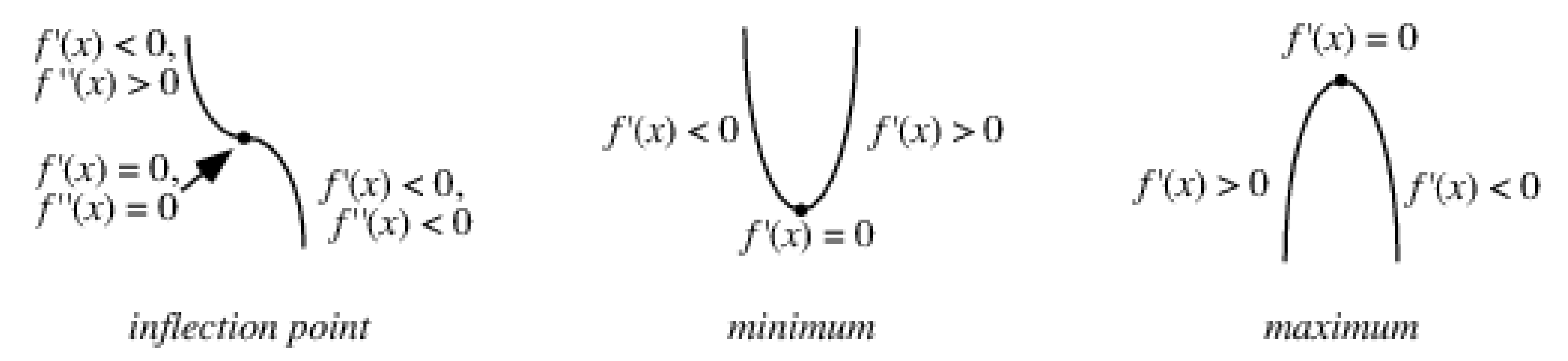
(x, y) is a critical point iff $\nabla f(x,y) = 0$ (in the sense the gradient is zero vector)

Characterization of critical points in 1D

Question: how to characterize a critical point as **min / max / saddle ???**

Apply calculus methods !!!

The first derivative test for critical points in 1D



Algorithm for finding and classifying critical points of a 1D function using the 1st derivative test

IN: function f, search interval $[min, max]$**,** h 'resolution parameter', ε 'precision parameter'
OUT: set of n critical points locations $\{x_i\}_1^n$ and their characterization as min/max/saddle.

1. Compute the numerical derivative (with resolution h in interval $[min, max]$).
2. Find critical points locations , i.e. x values for which $|f'(x)| < \epsilon$.
3. Check the sign of the derivative on the left-hand and right-hand side of the found x values.
4. If $-, -$ or $+, +$ then classify as inflection point, if $-, +$ then classify as minimum, and if $+, -$ then classify as maximum.

Computing the second derivative numerically

The second derivative of function f

at x

$$f''(x) = \frac{d^2 f}{d\,x^2}(x)$$

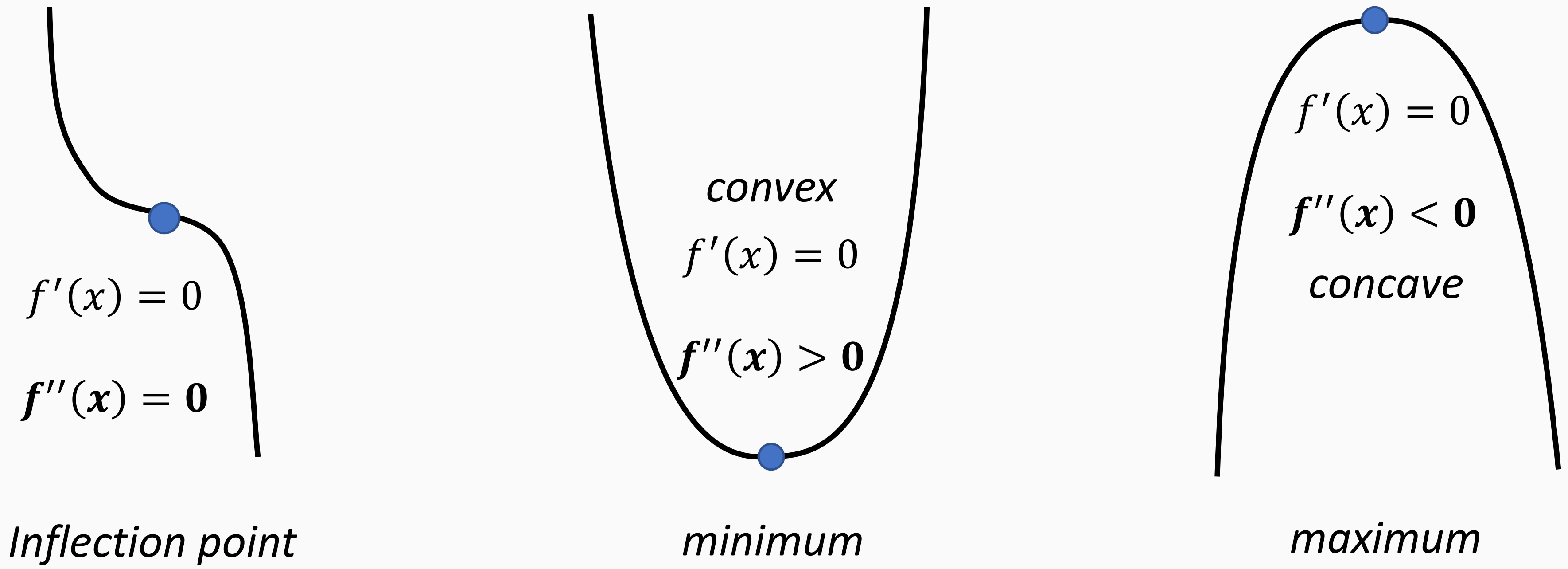
taken twice with respect to x

Interpretation: 1st derivative measures the rate of change, 2nd derivative measures convexity(when+)/concavity(when-)

Numerical formula for the second derivative in 1D

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

The second derivative test for critical points in 1D



Algorithm for finding and classifying critical points of a 1D function using the 2nd derivative test

IN: function f, search interval $[min, max]$**,** h 'resolution parameter', ε 'precision parameter'
OUT: set of n critical points locations $\{x_i\}_1^n$ and their characterization as min/max/saddle.

1. Compute the numerical derivative (with resolution h in interval $[min, max]$).
2. Find critical points locations , i.e. x values for which $|f'(x)| < \epsilon$.
3. Compute the second derivative at the found x values.
4. If 2nd derivative has value $< \varepsilon$ then classify as inflection point, if $-$ then classify as maximum, if $+$ then classify as minimum.

Second order partial derivatives

The second derivative of f

usually same

$$\frac{\partial f^2}{\partial x \partial x}(x,y) \qquad \frac{\partial f^2}{\partial x \partial y}(x,y) = \frac{\partial f^2}{\partial y \partial x}(x,y) \qquad \frac{\partial f^2}{\partial y \partial y}(x,y)$$

taken first with respect to y and then w.r.t. x

Altogether they form the so-called 'Hessian' matrix

$$f''(x,y) = \begin{bmatrix} \frac{\partial f^2}{\partial x \partial x}(x,y) & \frac{\partial f^2}{\partial x \partial y}(x,y) \\ \frac{\partial f^2}{\partial y \partial x}(x,y) & \frac{\partial f^2}{\partial y \partial y}(x,y) \end{bmatrix}$$

Also use a different notation

$$\partial_{xx} f(x,y) \quad \partial_{xy} f(x,y) \quad \partial_{yx} f(x,y) \quad \partial_{yy} f(x,y)$$

Eigenvalues and Eigenvectors

Eigenvalues and Eigenvectors

For an $n \times n$ matrix $\mathbf{A}$, scalars λ and vectors $\mathbf{x}_{n\times 1} \neq \mathbf{0}$ satisfying $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$ are called ***eigenvalues*** and ***eigenvectors*** of $\mathbf{A}$, respectively, and any such pair, $(\lambda, \mathbf{x})$, is called an ***eigenpair*** for $\mathbf{A}$. The set of *distinct* eigenvalues, denoted by $\sigma(\mathbf{A})$, is called the ***spectrum*** of $\mathbf{A}$.

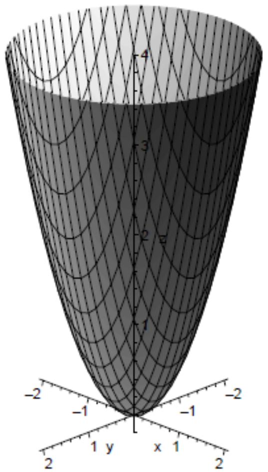
Postive/negative definiteness of a symmetric matrix

Given symmetric matrix $M \in \mathbb{R}^{n \times n}$ is

- Positive-definite if all its eigenvalues are positive (strictly) or equivalently $x^T \cdot M \cdot x > 0$ for all vectors $x \neq 0$,
- Negative-definite if all its eigenvalues are negative (strictly) or equivalently $x^T \cdot M \cdot x < 0$ for all vectors $x \neq 0$,
- Otherwise, it is *undefinite,*

Characterization of 2D (and multiD) critical points

EXAMPLE: **Standard minimum** $f(x,y) = x^2 + 3y^2$



Find critical points:

$$\partial_x f(x,y) = 2x, \qquad \partial_y f(x,y) = 6y$$

so the only critical point is the origin, $(0,0)$.

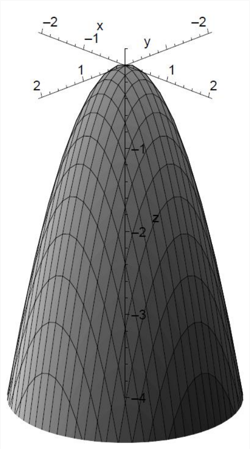
Second derivative test:

$$\partial_{xx} f(x,y) = 2, \qquad \partial_{xy} f(x,y) = 0, \qquad \partial_{yy} f(x,y) = 6$$

$f''(0,0)$ is the diagonal matrix

$$f''(0,0) = \begin{pmatrix} 2 & 0 \\ 0 & 6 \end{pmatrix}.$$

This is *positive definite* so the origin is a local minimum.



$$f(x,y) = -(x^2 + y^2)$$

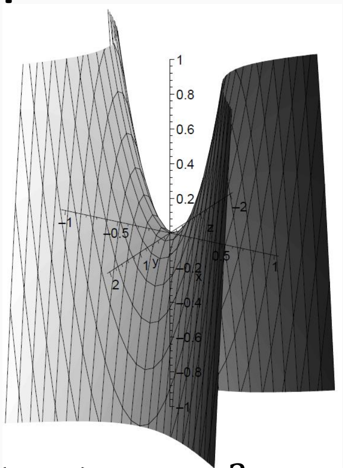
$$\partial_x f(x,y) = -2x, \qquad \partial_y f(x,y) = -2y,$$

The critical point is $(0,0)$,
Second derivative test:

$$\partial_{xx} f(x,y) = -2,$$
$$\partial_{yy} f(x,y) = -2,$$
$$\partial_{xy} f(x,y) = 0.$$

$$f''(0,0) = \begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix}$$

This is *negative-definite*,
so $(0,0)$ is a *local maximum.*



$$f(x,y) = -x^2 + 3y^2$$

$$\partial_x f(x,y) = -2x, \qquad \partial_y f(x,y) = 6y,$$

The critical point is $(0,0)$,
Second derivative test:

$$\partial_{xx} f(x,y) = -2,$$
$$\partial_{yy} f(x,y) = 6,$$
$$\partial_{xy} f(x,y) = 0.$$

$$f''(0,0) = \begin{bmatrix} -2 & 0 \\ 0 & 6 \end{bmatrix}$$

This is *undefined*,
and no zero eigenvalues,
so $(0,0)$ is a *saddle*.

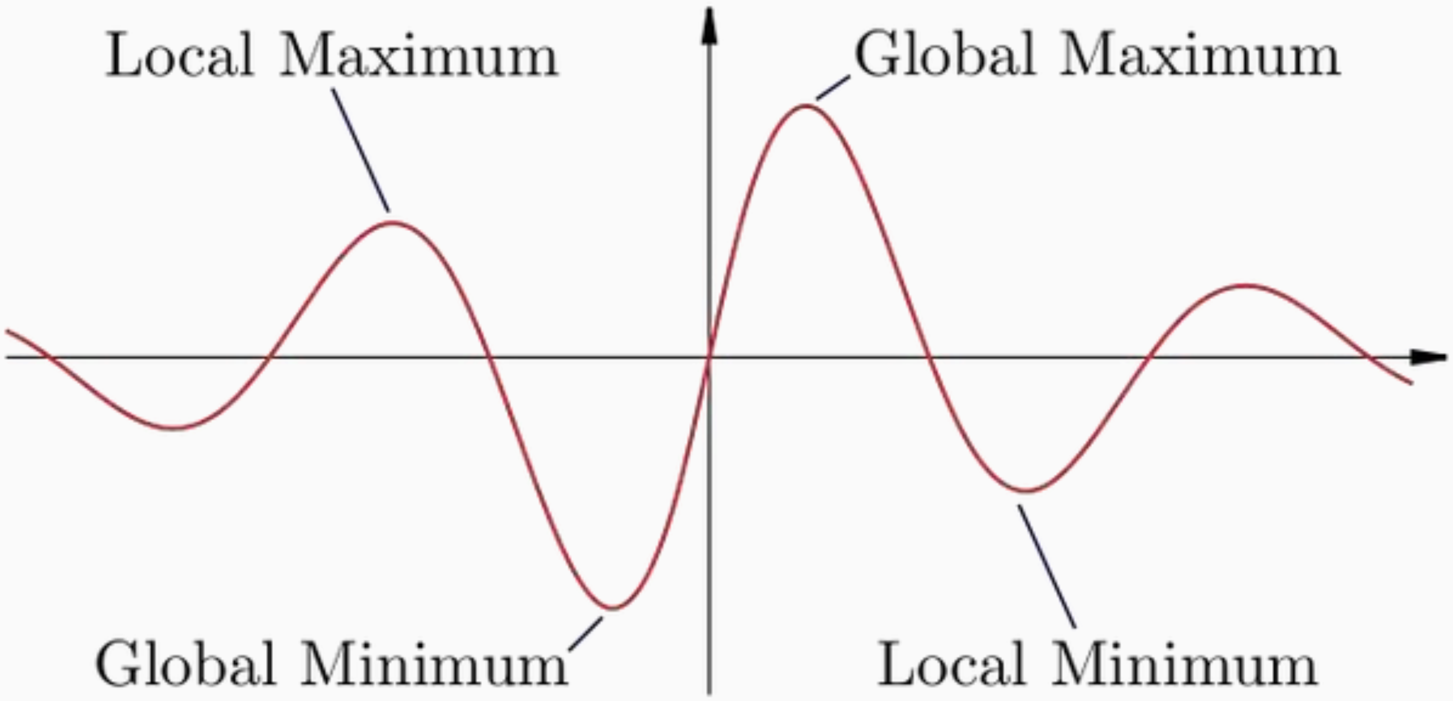
Computing 2nd order derivatives numerically

$$f_{xx}(x,y) \approx \frac{f(x+h,y) - f(x,y) + f(x-h,y)}{h^2},$$

$$f_{yy}(x,y) \approx \frac{f(x,y+h) - 2f(x,y) + f(x,y-h)}{h^2},$$

$$f_{xy}(x,y) \approx \frac{f(x+h,y+h) - f(x+h,y-h) - f(x-h,y+h) + f(x-h,y-h)}{4h^2}$$

Global / local minima

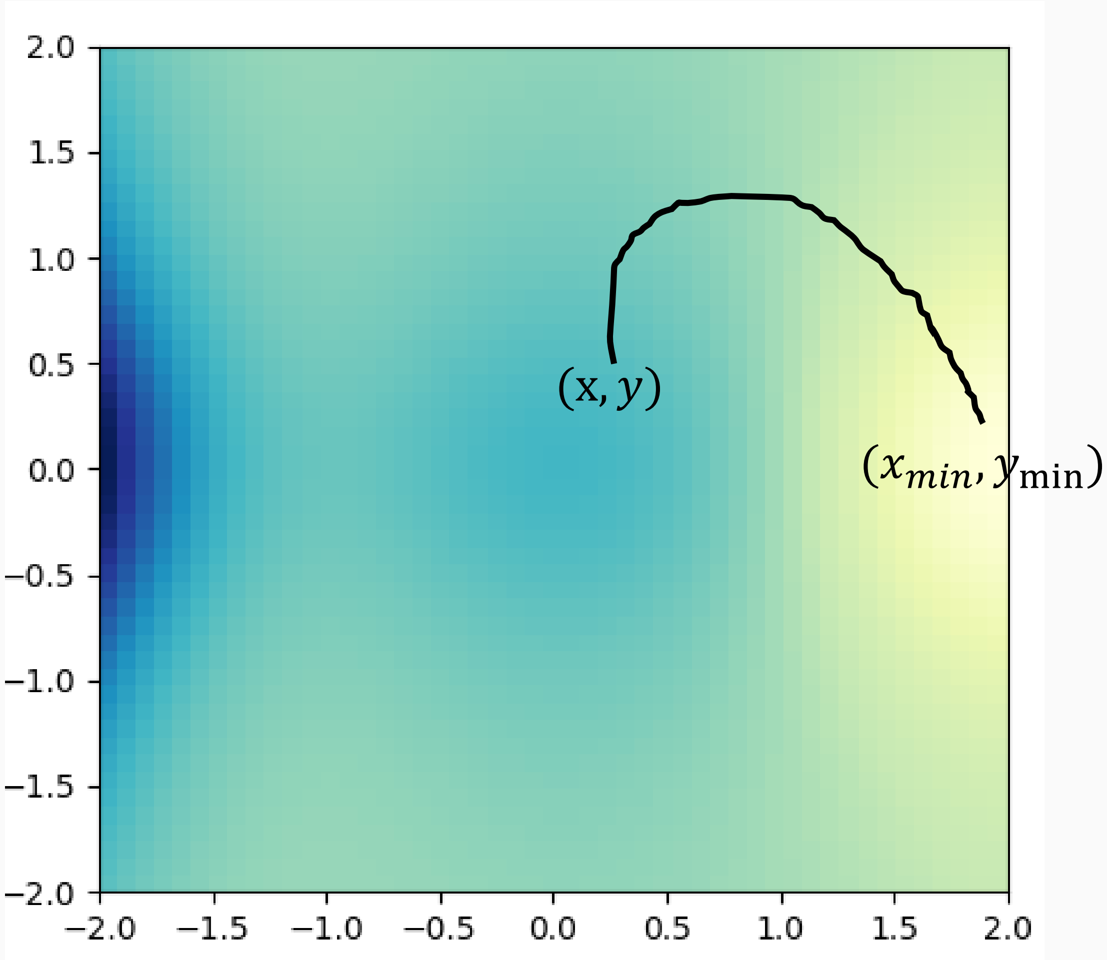


- Maximum = local maximum or global maximum,
- Minimum = local minimum or global minimum,
- There is only one global max and global min,
- There can be a lot of local min and local max.

How to find a minimum of a 2D function?

Big question: How to find minima of a multiD function, without checking the numerical derivative of all points on a multiD grid ???
(The first optimization algorithm)

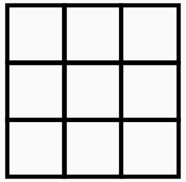
Algorithm for descent without gradient method



Algorithm for descent without gradient method

IN: function f, search box $[min, max]x[min, max]$, h 'resolution parameter'
OUT: a minimum position $(x_{min}, y_{\min})$

1. Choose the initial point (x, y) by random,
2. Considering the current point(x, y), investigate f values for all nodes in the neighborhood
   $(x + h, y), (x, y + h),$
   $(x + h, y + h), (x - h, y - h),$
   $(x + h, y - h), (x - h, y + h),$
   $(x - h, y), (x, y - h)$
3. Set new (x, y) to the point in the neighborhood attaining the lowest f value (also lower than current $f(x, y)$)
4. If there is no such point terminate, otherwise go to Step 2.



WARNING:
This Optimization method is impractical,
We are implementing it to see advantages of the gradient descent method.

Introduction to the gradient descent algorithm

Gradient descent of a 2D function

NEW x OLD x LEARNING RATE OLD x OLD y

$$x := x - \alpha \frac{\partial}{\partial x} f(x, y),$$

GRADIENT

$$y := y - \alpha \frac{\partial}{\partial y} f(x, y)$$

Update simultaneously x and y.

$\alpha > 0$ – *learning rate* parameter.

Repeat until convergence

Problem when using numerical derivatives , they generate error at each step.
Better to use analytic derivatives of backprop algorithm for computing gradients.

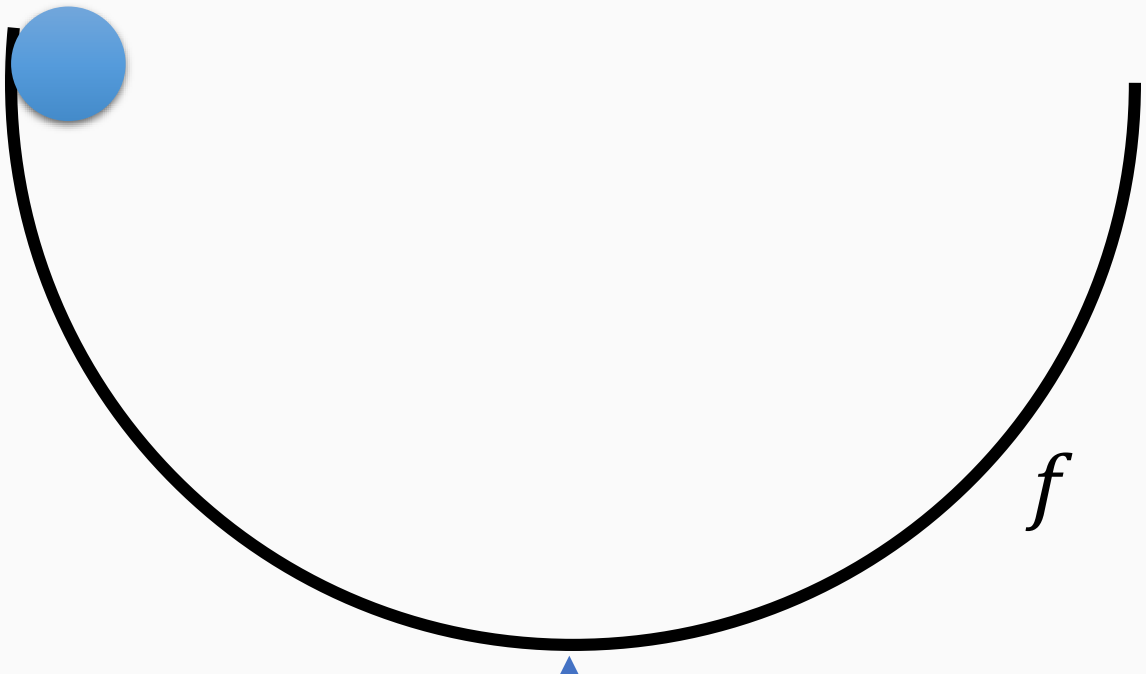
Simultaneous update

$$\texttt{temp0} := x - \alpha \frac{\partial}{\partial x} f(x,y)$$
$$\texttt{temp1} := y - \alpha \frac{\partial}{\partial y} f(x,y)$$
$$x := \texttt{temp0}$$
$$y := \texttt{temp1}$$

INCORRECT !

$$x := x - \alpha \frac{\partial}{\partial x} f(x,y)$$
$$y := y - \alpha \frac{\partial}{\partial y} f(x,y)$$

Convex Optimization



Show optimization using gradient descent of a quadratic function [mathematica demonstrations\ConvergenceOfMinimizationMethods.cdf]

1. Given a convex function f
2. Locate the minimizer by following the descent

It follows that there is a unique minimizer

The first convex optimization problem

Minimizing a quadratic function, where Q is a <u>square symmetric positive definite</u> matrix, and b is a vector

$$f(x) = \frac{1}{2} x^T \cdot Q \cdot x + b^T \cdot x$$

Observe that now <u>x denotes a vector $x \in R^n$</u>, we use interchangeably x as a vector and a number (x is a number on slide 30)

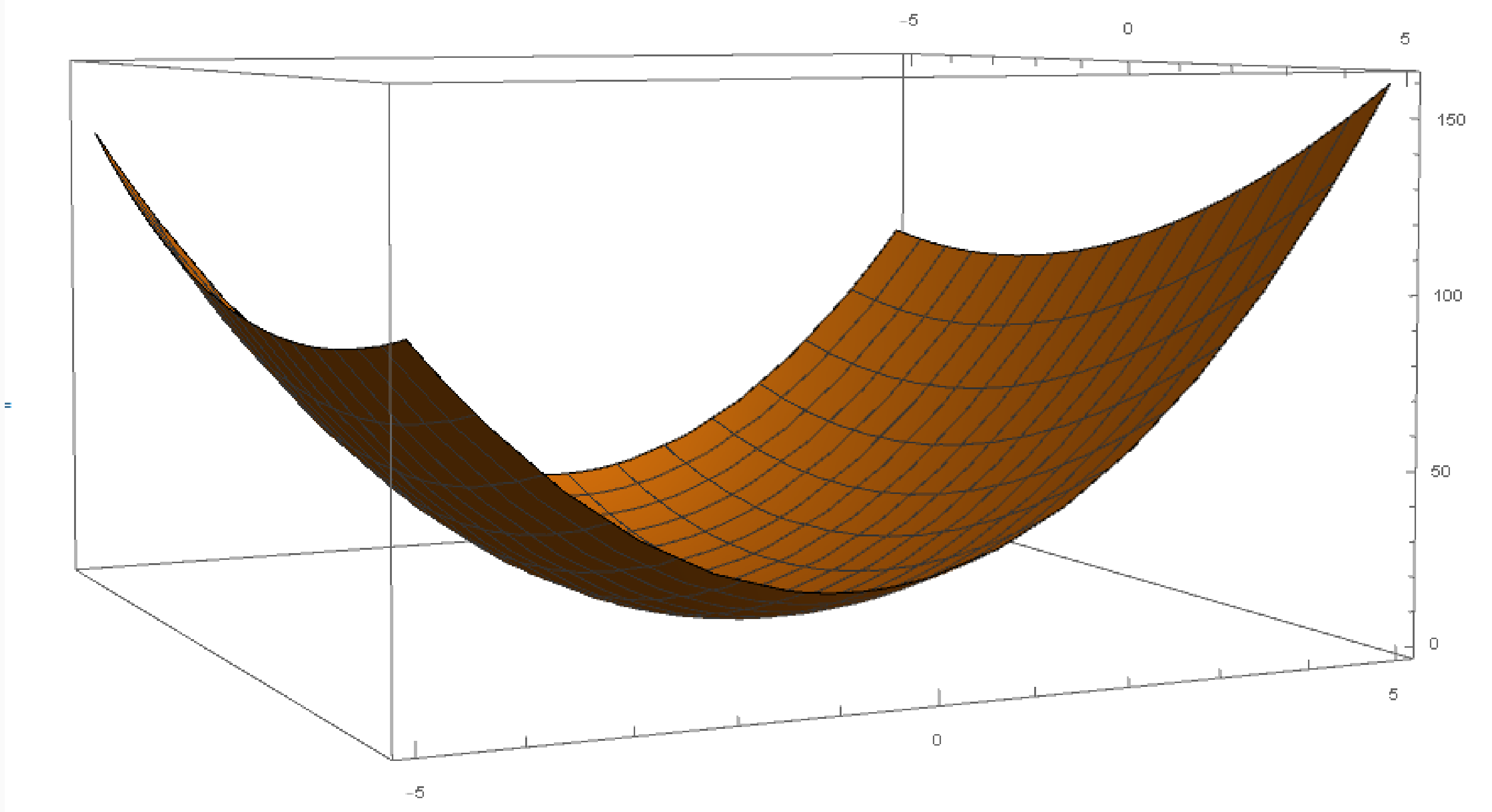
Symmetric matrix satisfies $Q = Q^T$

The minimal point can be computed analytically by the formula $x_{min} = -b^T \cdot Q^{-1}$

The formula for the gradient $\nabla_x f(x)$ is simple $\nabla_x f(x) = x^T Q + b^T$

Example of a quadratic function

For example take $Q = \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & 2 \end{bmatrix}$, and $b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.



Gradient descent for a quadratic function

$$\text{Consider example } Q = \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & 2 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

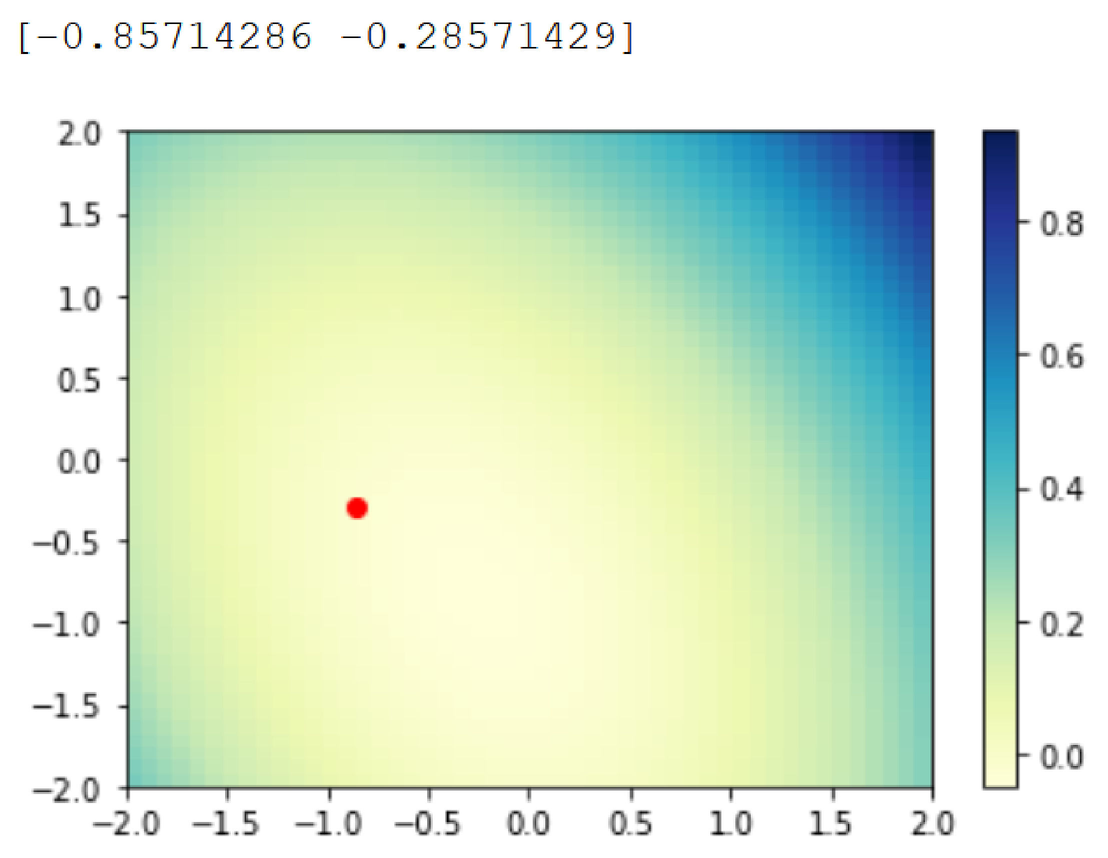
$$x^\star = [-0.85714286, -0.28571429]$$

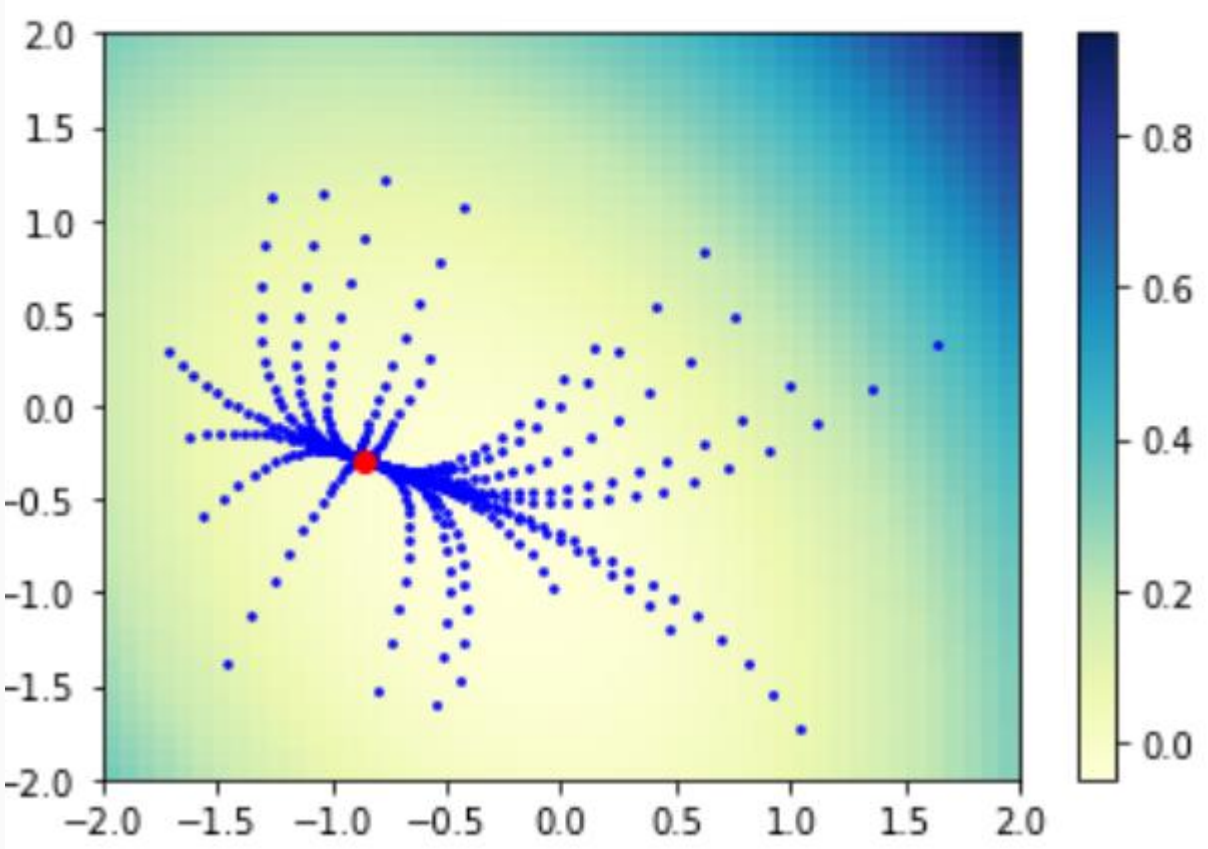
$$f(x) = \frac{1}{2} x^T \cdot Q \cdot x + b^T \cdot x$$

Gradient descent $x := x - \alpha \nabla f(x)$
(compactified notation)
$x \epsilon R^2$ is a two-dimensional vector,
x_t denotes the t-th iteration.

See example computations and plots in [jupyter_notebooks\week2_2.ipynb](jupyter_notebooks\week2_2.ipynb)

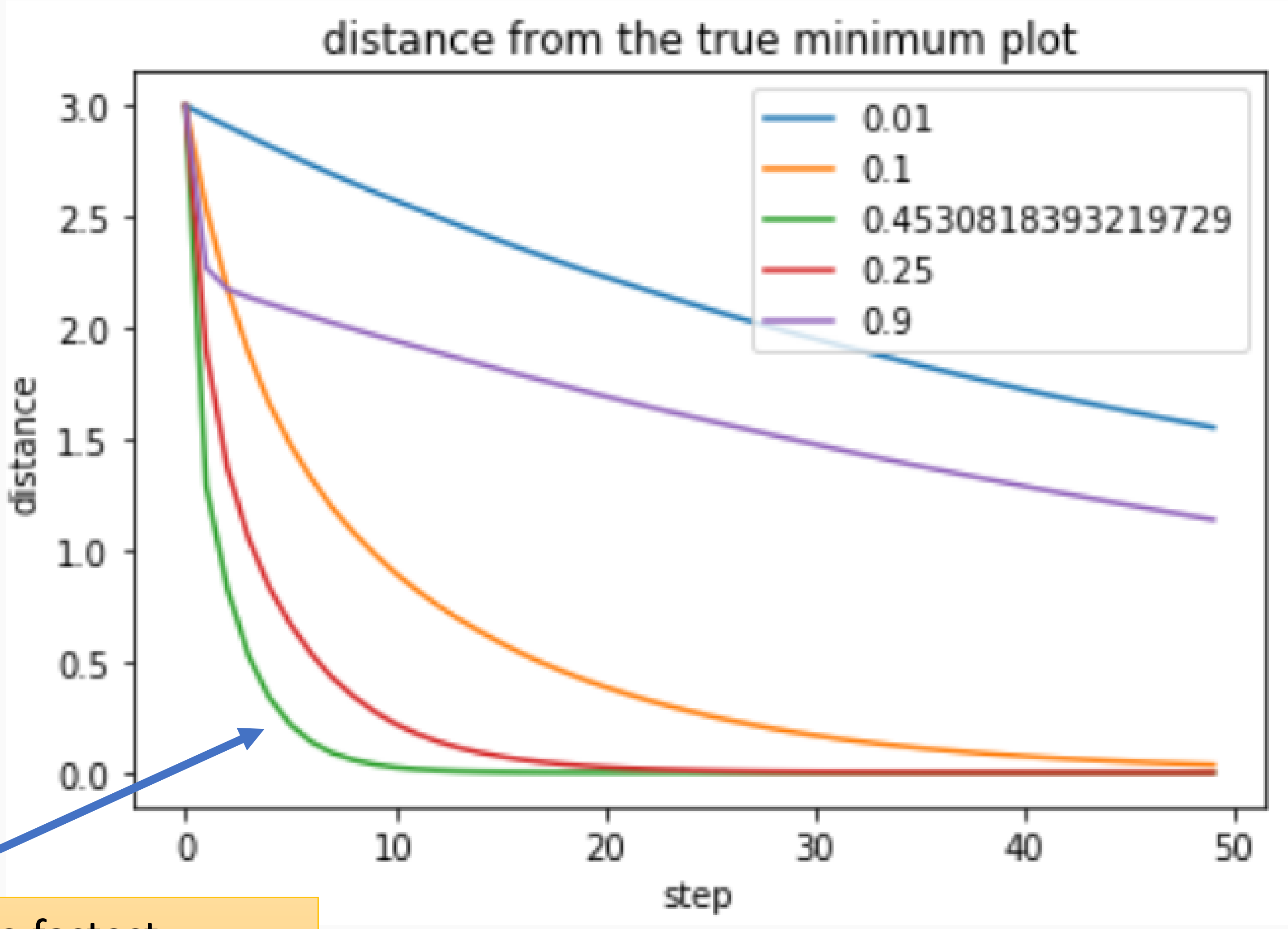
Gradient descent for the quadratic function





Behavior for different learning rates

the plot shows
$dist = ||x^* - x_t||$



We observe the fastest convergence for $\alpha = 0.453$ ... **Why ?**

A Glimpse at gradient descent convergence theory

Let $L > 0$ be the *Lipschitz constant* of the gradient, i.e.

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \text{ for all } x, y \in \mathbb{R}^d.$$

The following theorem holds (informal statement below)

Theorem 1. *Let $x^\star$ be a global minimum of f. ∇f is* Lipschitz *with a constant $L > 0$ (see above). Choosing*

$$\alpha = \frac{1}{L}.$$

Then, gradient descent *with any x_0 satisfies*

1. *Function values are monotone decreasing*

$$f(x_{t+1}) \leq f(x_t) - \frac{1}{2L}\|\nabla f(x_t)\|^2$$

2.

$$\|f(x_T) - f(x^\star)\| \leq \frac{L}{2T}\|x_0 - x^\star\|^2, \quad T > 0.$$

Remark on the Theorem

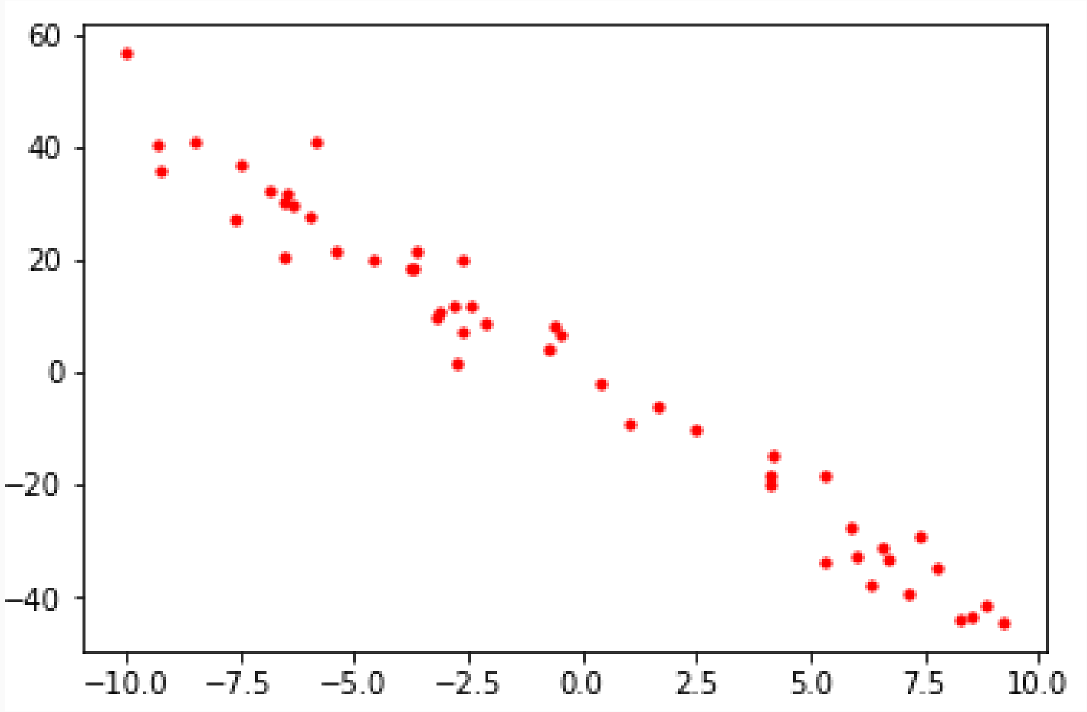
The Theorem provides us with a good guess for the learning rate.
Informally, it guarantees that for the given choice of learning rate the function value along the gradient descent path is decreasing at least like $\frac{1}{T}$. It is provided as an example of mathematical statement about gradient descent applied to convex problems.

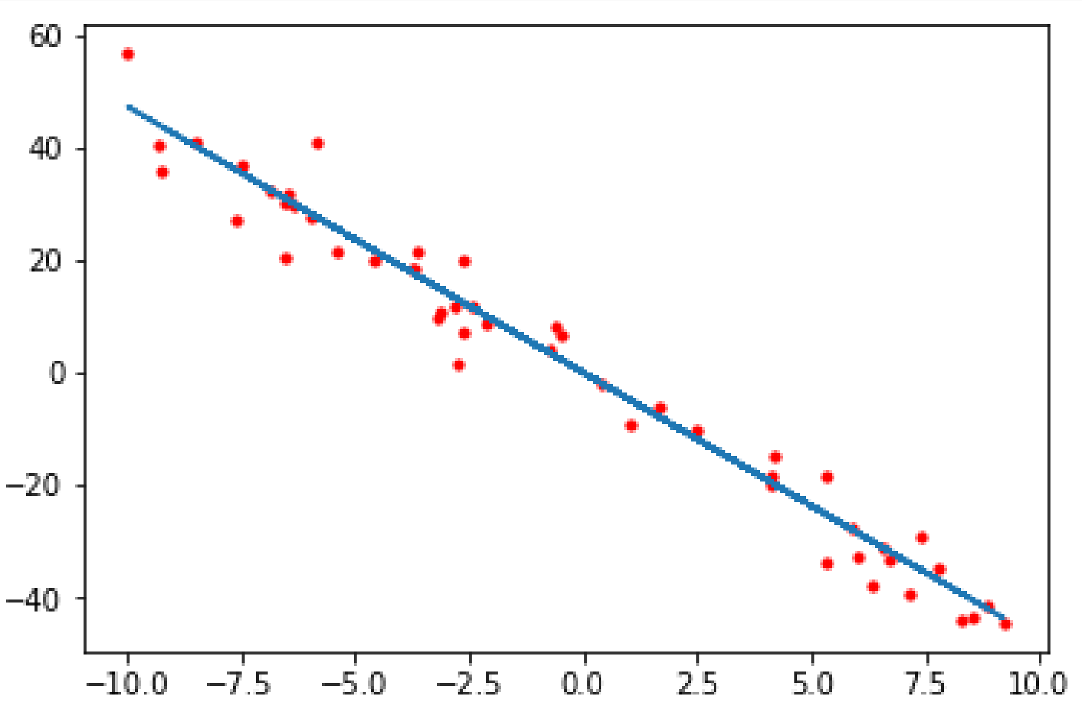
However, for most practical problems Theorem cannot be applied literally, it serves as a guide exclusively

- There is no global Lipschitz constant, only local, so the learning rate is usually being adapted from step to step,
- Lipschitz constant is computationally expensive to estimate,
- Practical optimization problems are nonconvex.

Linear Regression

Statement of the *linear regression problem*:
given data , find a linear function which is the best approximator





Given a set of N data-points $\{(x_i, y_i)\}$

Best approximator = the line $mx + b$ that minimizes $\frac{1}{N}\sum(y_i - (mx_i + b))^2$

Linear regression using gradient descent

The task is

Find $\boldsymbol{m}, \boldsymbol{b}$ that minimize the '*mean square error*' $MSE = \frac{1}{N}\sum(y_i - (mx_i + b))^2$

1. Compute the gradient

$$\frac{\partial MSE}{\partial m} = \frac{2}{N}\sum_{i=1}^{N} -x_i(y_i - (mx_i + b)),$$

$$\frac{\partial MSE}{\partial b} = \frac{2}{N}\sum_{i=1}^{N} -(y_i - (mx_i + b)),$$

It can be written using vectors (NumPy way)

$$\frac{\partial MSE}{\partial m} = \frac{2}{N}\left(-x^T y + m x^T x + x^T \hat{b}\right),$$

$$\frac{\partial MSE}{\partial b} = \frac{2}{N}(-1^T y + m 1^T x + 1^T \hat{b}),$$

where $1 = [1, 1, \ldots, 1]$, i.e. is a vector of ones, and $\hat{b} = [b, b, \ldots, b]$ is a vector of b's.

Linear regression using gradient descent

2. Perform the simultaneous update

$$\texttt{temp0} := m - \alpha \frac{\partial}{\partial m} MSE(m, b)$$

$$\texttt{temp1} := b - \alpha \frac{\partial}{\partial b} MSE(m, b)$$

$$m := \texttt{temp0}$$

$$b := \texttt{temp1}$$

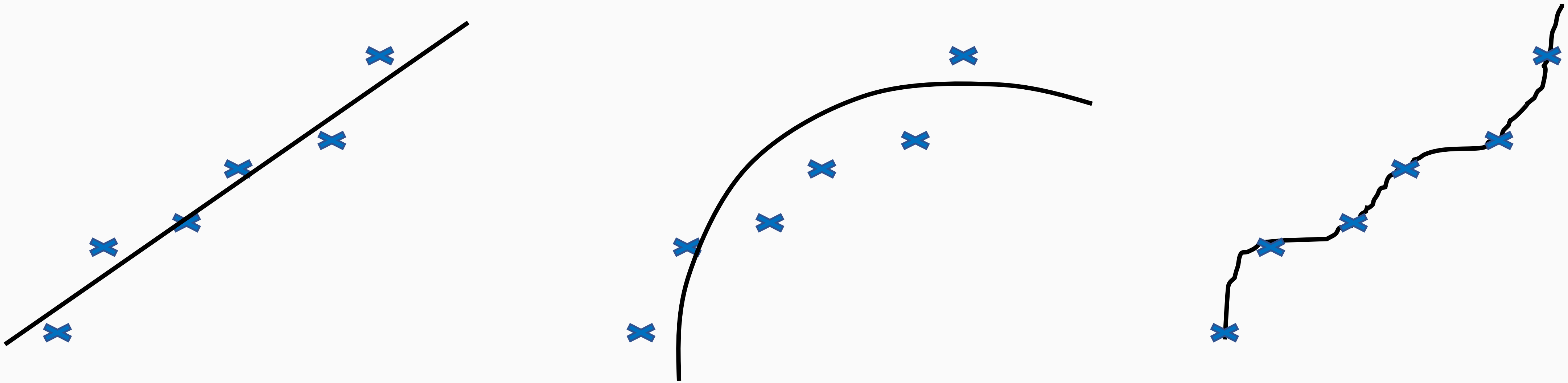
3. Until convergence

The analytic solution is given by

$$mx^T x - x^T y + x^T \hat{b} = 0,$$

$$m1^T x - 1^T y + 1^T \hat{b} = 0,$$

Nonlinear polynomial regression – fitting a curve



The task is of nonlinear regression for 2D datapoints
Find $\boldsymbol{b}, \{\boldsymbol{m_i}\}_{\boldsymbol{i=1}}^{\boldsymbol{p}}$ that minimize the '*mean square error*'
(fit a curve that approximates the data the best)

MSE for the nonlinear regression using p-th order polynomials

$$MSE = \frac{1}{N}\sum(y_i - (m_1 x_i^1 + m_2 x_i^2 + \cdots + m_p x_i^p + b))^2$$

Nonlinear polynomial regression cont.

Nonlinear regression for 2D datapoints

Using the compact polynomial notation

$m_1 x_i^1 + m_2 x_i^2 + \cdots + m_p x_i^p + b$ = $\sum_{j=1}^{p} m_j x_i^j + b$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(\sum_{j=1}^{p} m_j x_i^j + b\right)\right)^2$$

Solving nonlinear regression using gradient descent

We can solve nonlinear regression using gradient descent like we solved linear regression

For the quadratic regression we have the following gradients

$$\frac{\partial MSE}{\partial m_1} = \frac{2}{N}\sum_{i=1}^{N} -x_i(y_i - (m_1 x_i + m_2 x_i^2 + b)),$$

$$\frac{\partial MSE}{\partial m_2} = \frac{2}{N}\sum_{i=1}^{N} -x_i^2(y_i - (m_1 x_i + m_2 x_i^2 + b)),$$

$$\frac{\partial MSE}{\partial b} = \frac{2}{N}\sum_{i=1}^{N} -(y_i - (m_1 x_i + m_2 x_i^2 + b)),$$

And perform simultaneous update (like on slide 41) of the three parameters $m_{1,} m_2, b$

Nonlinear regression of a general polynomial

We can solve nonlinear regression for a arbitrary polynomial

For the quadratic regression we have the following gradients

$$\frac{\partial MSE}{\partial m_1} = \frac{2}{N}\sum_{i=1}^{N} -x_i\left(y_i - \left(\sum_{j=1}^{p} m_j x_i^j + b\right)\right),$$

$$\frac{\partial MSE}{\partial m_2} = \frac{2}{N}\sum_{i=1}^{N} -x_i^2\left(y_i - \left(\sum_{j=1}^{p} m_j x_i^j + b\right)\right),$$

$$\ldots$$

$$\frac{\partial MSE}{\partial m_j} = \frac{2}{N}\sum_{i=1}^{N} -x_i^j\left(y_i - \left(\sum_{j=1}^{p} m_j x_i^j + b\right)\right),$$

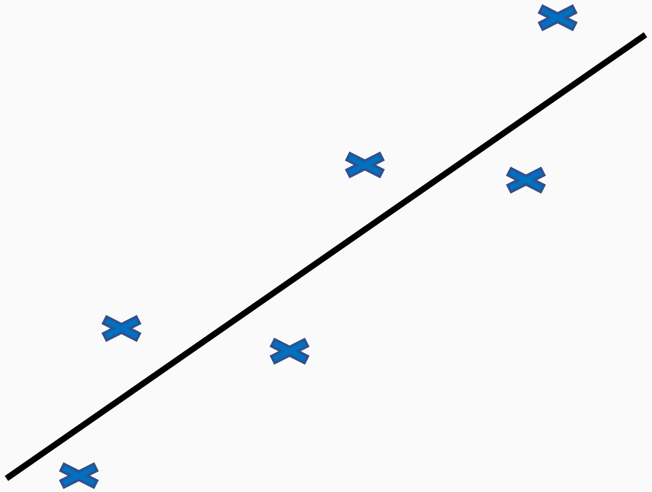
$$\ldots$$

$$\frac{\partial MSE}{\partial b} = \frac{2}{N}\sum_{i=1}^{N} -(y_i - (\sum_{j=1}^{p} m_j x_i^j + b)),$$

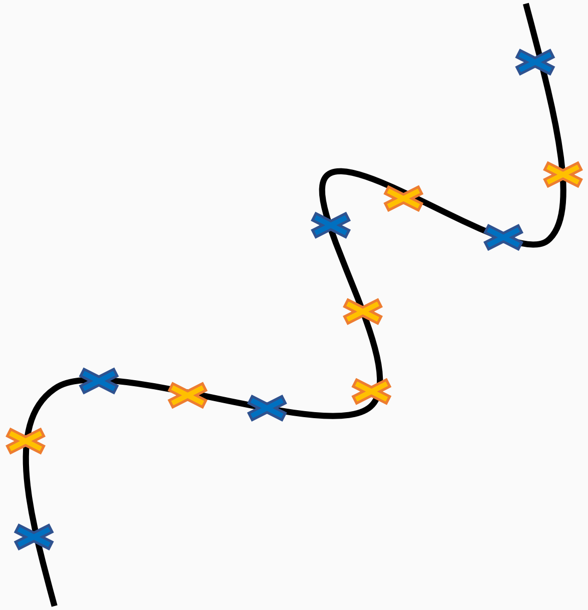
If doing nonlinear regression with a p-th order polynomial, then there are $p+1$ parameters for gradient descent (can store them in a vector).

Overfitting / Underfitting

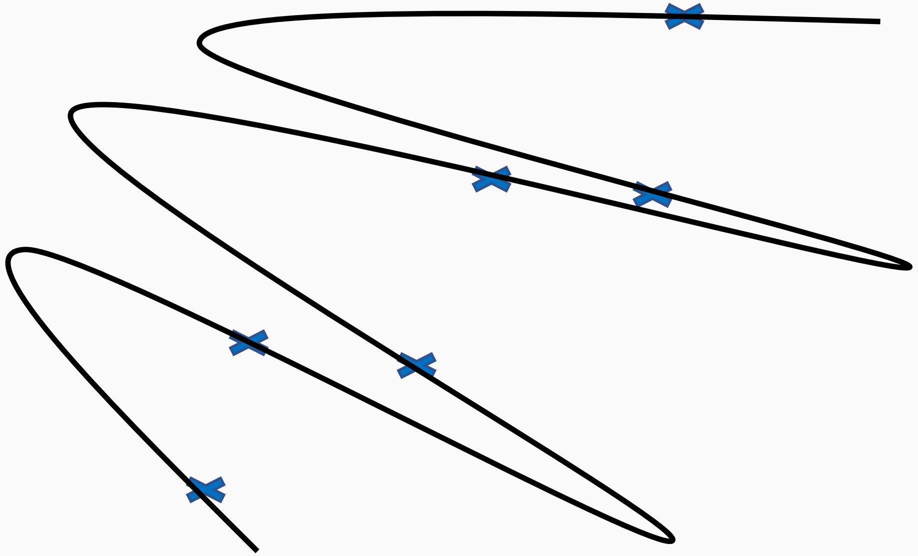
There is no a general recipe of choosing appropriate polynomials for nonlinear regression



Underfitted
(order too low)



Good generalization
order fits data well
Test datapoints



Overfitted
(order too large)

Glimpse of the modern machine learning research

Generalization of Deep Neural Networks is an open problem of machine learning research.

Why DNN seem to generalize well despite having enormous number of parameters???