Lecture 4:Linear Algebra and Optimization

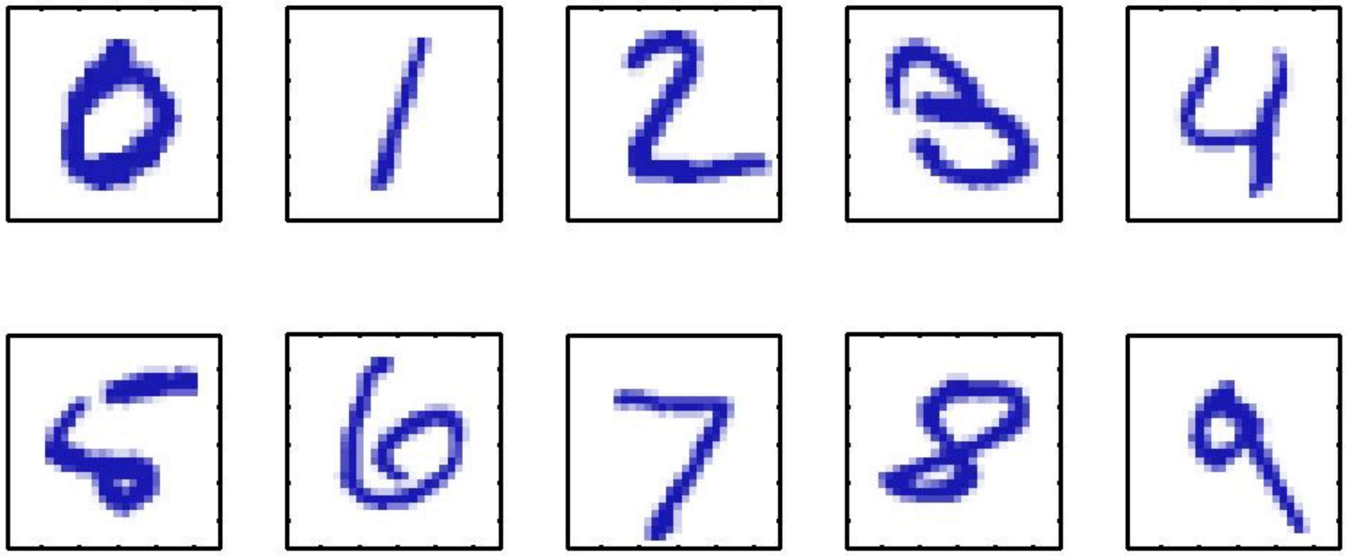
Andreas Wichert

Department of Computer Science and Engineering

Técnico Lisboa

Example

- Descriptors of objects like patterns are mostly represented as feature vectors of a fixed dimension



Norm

A norm is a function given a vector space V that maps a vector into a real number with

$$\|v\| \geq 0$$

and with α scalar

$$\|\alpha \cdot \mathbf{x}\| = |\alpha| \cdot \|\mathbf{x}\|$$

$$\|\mathbf{x} + \mathbf{y}\| \leq |\mathbf{x}\| + |\mathbf{y}\|.$$

The l_p norm is defined as the following (for $p = 2$ it is the Euclidean norm):

$$\|\mathbf{x}\|_p = (|x_1|^p + |x_2|^p + \cdots + |x_m|^p)^{\frac{1}{p}}$$

l_p norms are equivalent and the following relation holds for $0 < q < p$

$$\|\mathbf{x}\|_p \leq \|\mathbf{x}\|_q \leq m^{\frac{1}{q} - \frac{1}{p}} \cdot \|\mathbf{x}\|_p$$

Distance function

Metric defines a distance between two vectors with

$$d(\mathbf{x}, \mathbf{y}) \geq 0$$

symmetry

$$d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$$

and the triangle inequality

$$d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y}).$$

The l_p norm induces the distance between two points (metric)

$$d_p(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_p = (|x_1 - y_1|^p + |x_2 - y_2|^p + \cdots + |x_m - y_m^p)^{\frac{1}{p}}$$

The most popular metrics are the Taxicab or Manhattan metric d_1 with

$$d_1(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_1 = |x_1 - y_1| + |x_2 - y_2| + \cdots + |x_m - y_m|$$

and the Euclidean metric

$$d_2(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{|x_1 - y_1|^2 + |x_2 - y_2|^2 + \cdots + |x_m - y_m|^2}.$$

- Euclidean norm is induced by the inner product (scalar product)

$$\|\mathbf{x}\|_2 = \sqrt{\langle \mathbf{x} | \mathbf{x} \rangle}$$

- It defines a Hilbert space, which extends the two or three dimensional Euclidean space to spaces with any finite or infinite number of dimensions. A scalar product exists in l_2 but not in l_1 space.
- A normed vector space (does not need to have a scalar product) is called a Banach space. Without a scalar product there is no orthogonality.

Often one writes for Euclidean distance function and norm simply

$$d(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\| = \sqrt{|x_1-y_1|^2 + |x_2-y_2|^2 + \cdots + |x_m-y_m|^2}.$$

By normalising the vector to the length one the Euclidean distance function is constrained to the unit sphere

$$0 \leq d\left(\frac{\mathbf{x}}{\|\mathbf{x}\|}, \frac{\mathbf{y}}{\|\mathbf{y}\|}\right) = \left\|\frac{\mathbf{x}}{\|\mathbf{x}\|} - \frac{\mathbf{y}}{\|\mathbf{y}\|}\right\| \leq \sqrt{2}$$

and corresponds to the angle ω between the vectors

$$\cos\omega = \frac{\langle \mathbf{x}|\mathbf{y}\rangle}{\|\mathbf{x}\|\cdot\|\mathbf{y}\|}$$

with a similarity function

$$0 \leq sim(\mathbf{x},\mathbf{y}) = \cos\omega \leq 1$$

If we do not normalise the vectors, then we get the simple scalar product also called the dot product

$$\langle \mathbf{x} | \mathbf{w} \rangle = \cos \omega \cdot \|\mathbf{x}\| \cdot \|\mathbf{w}\|,$$

it is measure of the projection of one vector onto another.

The dot product of a vector with a unit vector is the projection of that vector in the direction given by the unit vector. The dot product is a linear representation represented by the value *net*,

$$y = net := \langle \mathbf{x} | \mathbf{w} \rangle = \sum_{i=1}^{D} w_j \cdot x_j,$$

David Hilbert, one of the most famous German mathematicians, attended a banquet in 1934, and he was seated next to the new minister of education, Bernhard Rust [Reid (1996)]. Rust asked, "How is mathematics in Göttingen now that it has been freed of the Jewish influence?" Hilbert replied, "Mathematics in Göttingen? There is really none any more." David Hilbert died in 1943. On his tombstone, at Göttingen, one can read his epitaph:

- Wir müssen wissen (We have to know)
- Wir werden wissen (We shall know!)

K-Nearest Neighbor

- In nearest-neighbor learning the target function may be either discrete-valued or real valued
- Learning a discrete valued function

- $f : \Re^d \rightarrow V$, *V* is the finite set *{v₁,......,vₙ}*

- For discrete-valued, the *k*-NN returns the most common value among the k training examples nearest to *x_q*.
- $Data = \{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots ,, (\mathbf{x}_N, t_N)\}$

$$f(\mathbf{x}_\eta) = t_\eta = v_\eta$$

K-Nearest Neighbor

$$Data = \{(\mathbf{x}_1, f(\mathbf{x}_1)), (\mathbf{x}_2, f(\mathbf{x}_1)), \cdots , , f(\mathbf{x}_N, (\mathbf{x}_N))\}$$

- Training algorithm
  - For each training example ***(x**,f(**x**))* add the example to the list

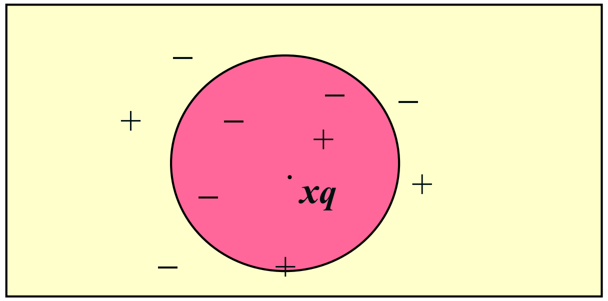
- Classification algorithm
  - Given a query instance x_q to be classified
    - Let $x_1,..,\mathbf{x}_k$ *k* instances which are nearest to $\mathbf{x}_q$

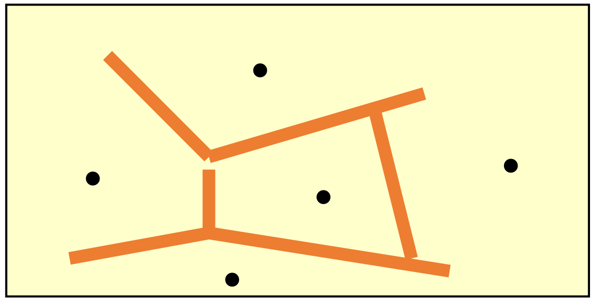
$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\arg\max} \sum_{i=1}^{k} \delta(v, f(x_i))$$

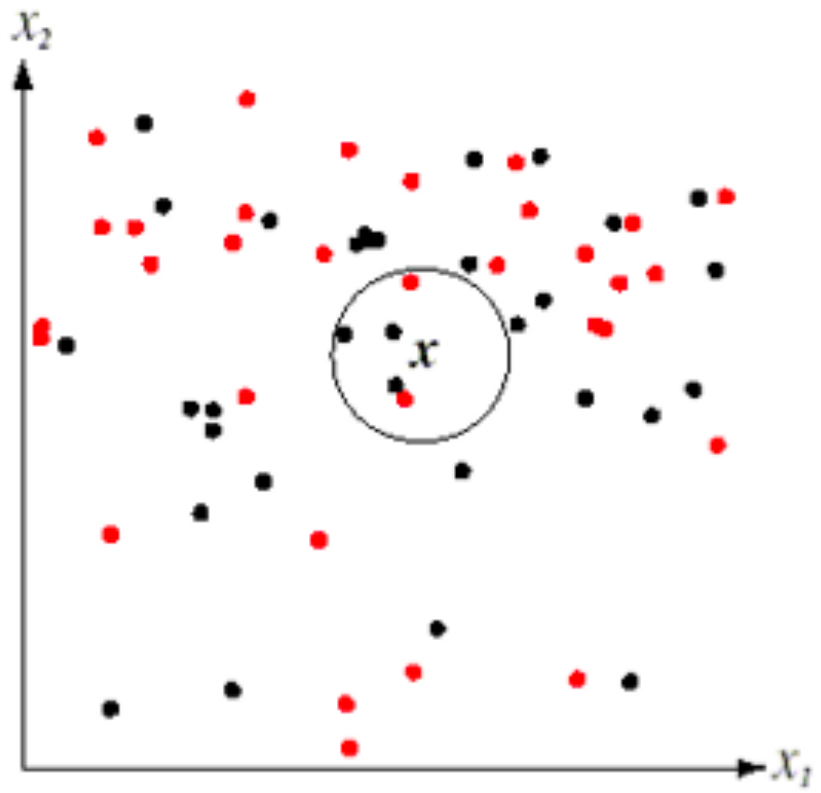
    - Where $\delta(a,b)=1$ if $a=b$, else $\delta(a,b)= 0$ (Kronecker function)

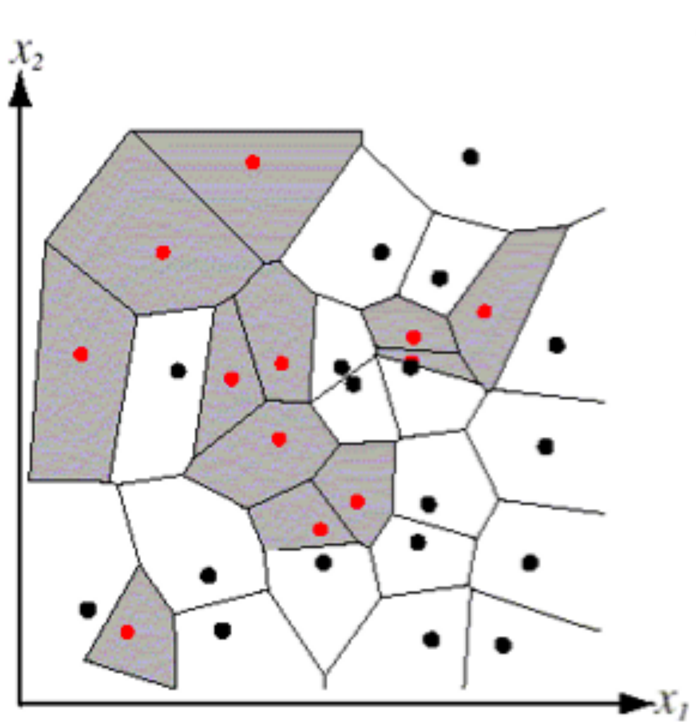
Definition of Voronoi diagram

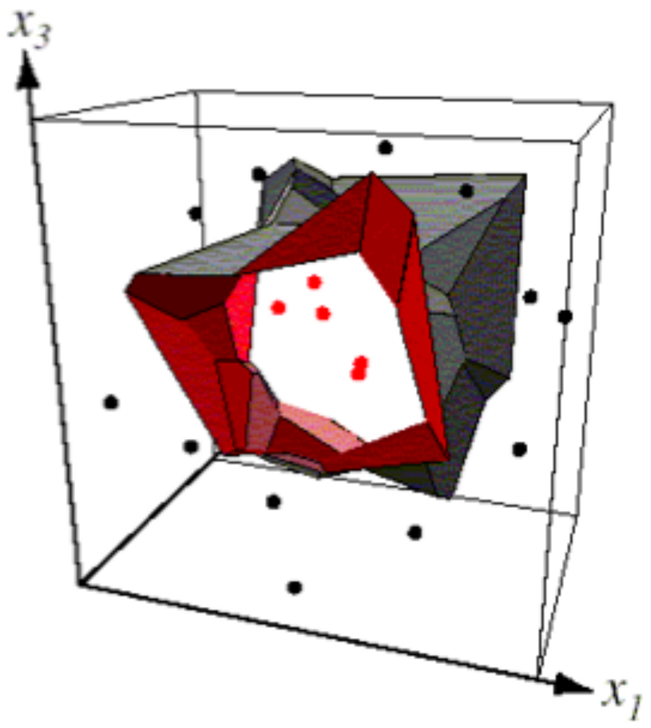
- The decision surface induced by 1-NN for a typical set of training examples.






x_2
x
x_1



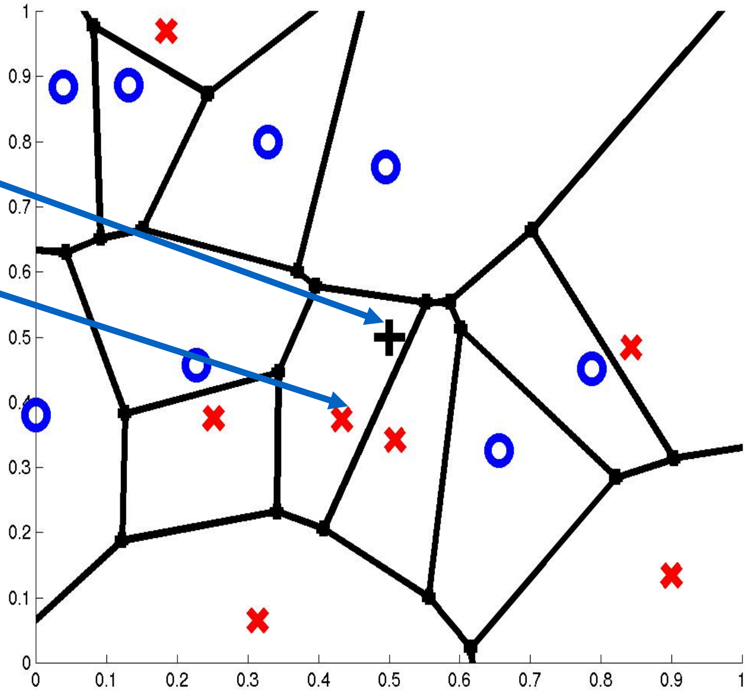


- kNN rule leeds to partition of the space into cells (Vornoi cells) enclosing the training points labelled as belonging to the same class
- The decision boundary in a Vornoi tessellation of the feature space resembles the surface of a crystall

1-Nearest Neighbor

query point x_q

nearest neighbor x_i

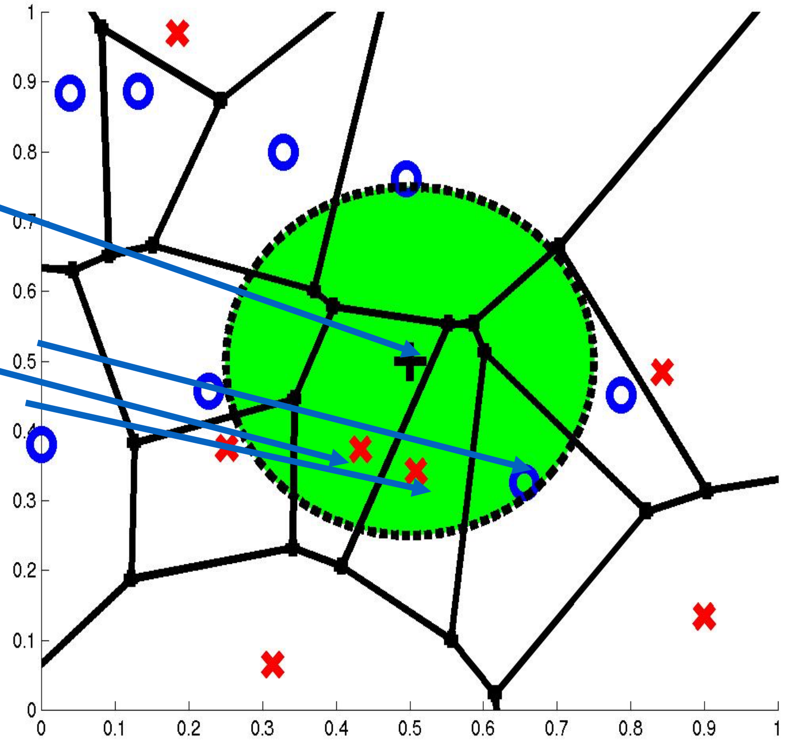


3-Nearest Neighbors

query point x_q

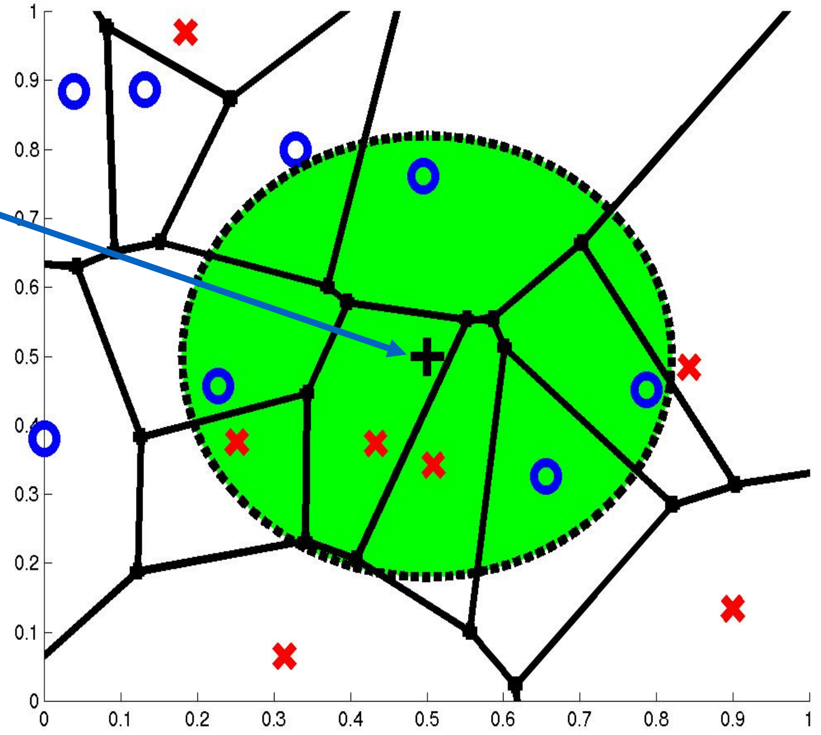
3 nearest neighbors

2x,1o



7-Nearest Neighbors

query point x_q



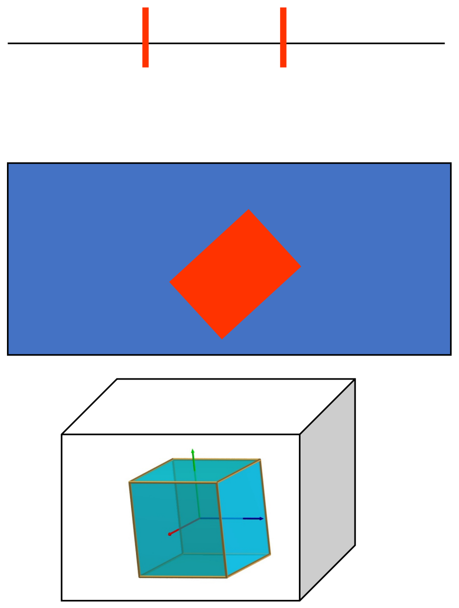
7 nearest neighbors

3x,4o

The Curse of Dimensionality

- A broad variety of mathematical effects can be observed when one increases the dimensionality of the data space
- One cannot think about these effects, by simply extending two- or three-dimensional experiences
- Rather, one has to think for example, at least *10*-dimensional to even see the effect occurring
- Furthermore, some are pretty nonintuitive

Curse of dimensionality
Some intuition



2

2^2

2^3

2^d

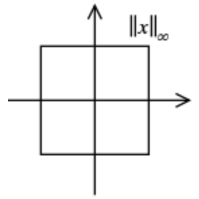
- 100 points cover the one-dimensional unit interval [0,1] on the real line quite well
- If one now considers the corresponding 10-dimensional unit hypersquare, 100 points are now isolated points in a vast empty space
- To get similar coverage to the one-dimensional space would now require 1020 points

- The volume of a high-dimensional cube approaches its surface with an in- crease in dimension
- In high-dimensional spaces, a partition is only performed in a few dimensions, which touch the boundary of the data space in most dimensions.

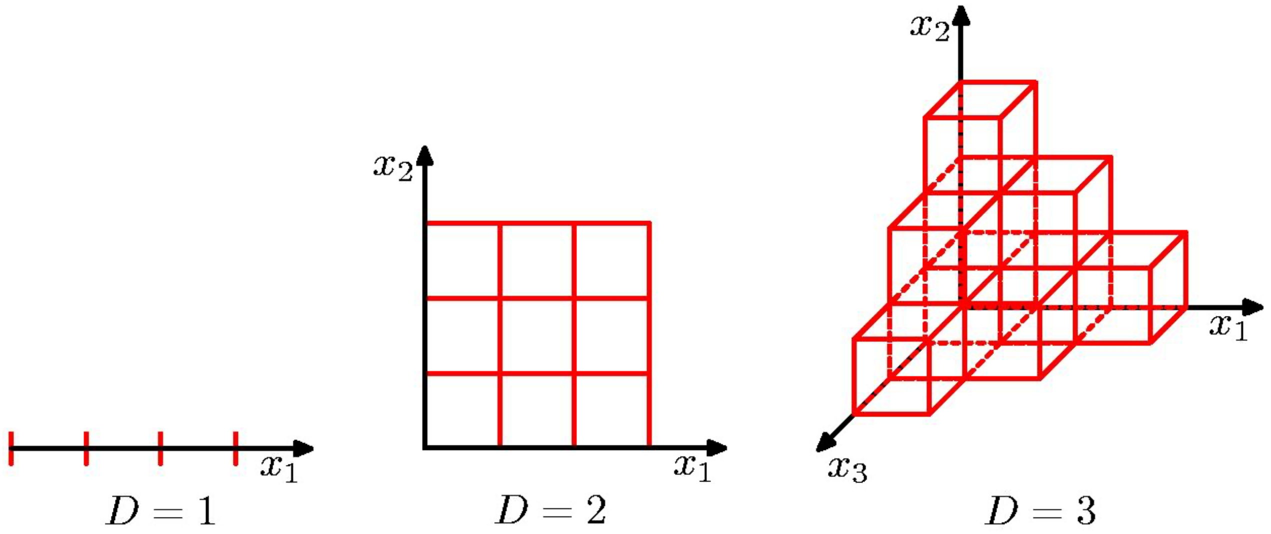
$$\|\mathbf{x}\|_\infty = max\left(|x_1|, |x_2|, \cdots, |x_m|\right).$$

- A hypercube corresponds to a sphere in l_∞ space; the volume v of a Euclidean ball of radius *r* in n-dimensional l_∞ space can be indicated by

$$v_n^\infty = (2 \cdot r)^n$$



Curse of Dimensionality



- Surprisingly the volume **decreases** for a sphere with fixed radius *r* in l_1 and l_2 space with the growing dimension
- The volume *v* of a sphere with l_1 norm and with radius *r* in n - dimensional space is given by

$$v_n^1 = \frac{2^n}{n!} \cdot r^n,$$

Suppose that r is fixed, then the volume v_n^1 approaches zero as n tends to infinity because

$$\lim_{n\to\infty} \frac{(2 \cdot r)^n}{n!} = 0.$$

On the other hand the radius r of a sphere with l_1 norm and with the volume v_n^1 in n-dimensional space is given by

$$r = \left(\frac{v_n^1 \cdot n!}{2^n} \right)^{1/n} = \frac{(v_n^1 \cdot n!)^{1/n}}{2},$$

Suppose that v_n^1 is fixed, then the radius r approaches infinity as n tends to infinity because

$$\lim_{n \to \infty} \frac{(v_n^1 \cdot n!)^{1/n}}{2} = \infty.$$

Consequences

- The same relation between the volume, the radius and the dimension are as well true for sphere with l_2 norm.
- Such a sphere with l_2 norm in dimension n is called a n ball, the volume can be indicated by explicit formulas

$$v_{2\cdot n}^2 = \frac{\pi^n}{n!} \cdot r^{2\cdot n},$$

$$v_{2\cdot n+1}^2 = \frac{2 \cdot n! \cdot (4 \cdot \pi)^n}{(2 \cdot n + 1)!} \cdot r^{2\cdot n+1}$$

- This relation have **serious consequences** for the similarity in popular l_1 and l_2 spaces!
  - In high dimensions the value of *r* that describes the radius of the sphere is **mostly larger** than the extension of the data space in most dimensions

Matrix Operations

Scalar multiplication is commutative

$$\langle \mathbf{x} | \mathbf{w} \rangle = \mathbf{x}^T \mathbf{w} = \mathbf{w}^T \mathbf{x} = \langle \mathbf{w} | \mathbf{x} \rangle$$

$$\begin{pmatrix} w_0 & w_1 \end{pmatrix} \cdot \begin{pmatrix} x_o \\ x_1 \end{pmatrix} = w_0 \cdot x_0 + w_1 \cdot x_1$$

but

$$\begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \cdot \begin{pmatrix} x_0 & x_1 \end{pmatrix} = \begin{pmatrix} w_0 \cdot x_0 & w_0 \cdot x_1 \\ w_1 \cdot x_0 & w_1 \cdot x_1 \end{pmatrix}$$

Matrix multiplication is not commutative

$$A \cdot B \neq B \cdot A$$

$$(A \cdot B)^T = B^T \cdot A^T$$

it is

$$A \cdot B = (B^T \cdot A^T)^T$$

A symmetric matrix is

$$A^T = A$$

An identity matrix of dimension D is defined as

$$I = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

An orthogonal matrix is

$$A^T \cdot A = A \cdot A^T = I$$

it implies that

$$A^{-1} = A^T$$

Tensor Product

The tensor product is defined for vector and matrices as

$$\begin{pmatrix} \omega_0 \\ \omega_1 \end{pmatrix} \otimes \begin{pmatrix} \omega_0 \\ \omega_1 \end{pmatrix} = \begin{pmatrix} \omega_0 \cdot \omega_0 \\ \omega_0 \cdot \omega_1 \\ \omega_1 \cdot \omega_0 \\ \omega_1 \cdot \omega_1 \end{pmatrix}$$

and

$$A \otimes B = \begin{pmatrix} a_{11} \cdot B & a_{12} \cdot B \\ a_{21} \cdot B & a_{22} \cdot B \end{pmatrix} = \begin{pmatrix} a_{11} \cdot b_{11} & a_{11} \cdot b_{12} & a_{12} \cdot b_{11} & a_{12} \cdot b_{12} \\ a_{11} \cdot b_{21} & a_{11} \cdot b_{22} & a_{12} \cdot b_{21} & a_{12} \cdot b_{22} \\ a_{21} \cdot b_{11} & a_{21} \cdot b_{12} & a_{22} \cdot b_{11} & a_{22} \cdot b_{12} \\ a_{21} \cdot b_{21} & a_{21} \cdot b_{22} & a_{22} \cdot b_{21} & a_{22} \cdot b_{22} \end{pmatrix}$$

Gradient

Consider a continuously differentiable function $f(\mathbf{x})$.

$$f : \mathbb{R}^D \to \mathbb{R} : f(\mathbf{x}) = y$$

It maps the vector $\mathbf{x}$ into a real value.
The gradient operator for $\mathbf{x}$ is

$$\nabla = \left[\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \cdots, \frac{\partial}{\partial x_D}\right]^T$$

or

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_D} \end{pmatrix}$$

Jacobian matrix

Then the Jacobian matrix J of $\mathbf{f}$ is an $K \times D$ matrix

$$J = \left(\frac{\partial \mathbf{f}}{\partial x_1}, \frac{\partial \mathbf{f}}{\partial x_2}, \cdots \frac{\partial \mathbf{f}}{\partial x_D} \right)$$

$$J = \begin{pmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_D} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f(_2\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_D} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_K(\mathbf{x})}{\partial x_1} & \frac{\partial f_K(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_K(\mathbf{x})}{\partial x_D} \end{pmatrix}$$

- If $K = 1$ f is a **scalar fiel**d and the Jacobian matrix is reduced to a row vector of partial derivatives of f .
- It is the ***transpose*** of the gradient of f, when denoted as column vector.

Rules

$$\nabla(a \cdot f(\mathbf{x}) + b \cdot g(\mathbf{x})) = a \cdot \nabla f(\mathbf{x}) + b \cdot \nabla g(\mathbf{x}), \quad a, b \in \mathbb{R}$$

$$\nabla\left((\mathbf{x}^T \cdot A \cdot \mathbf{x}\right) = \left(A + A^T\right) \cdot \mathbf{x}$$

If A symmetric, then

$$\nabla\left((\mathbf{x}^T \cdot A \cdot \mathbf{x}\right) = 2 \cdot A) \cdot \mathbf{x}$$

$$\nabla_x \left(\mathbf{y}^T \mathbf{x}\right) = \mathbf{y}$$

Second Derivative

Consider a continuously twice differentiable function $f(\mathbf{x})$.

$$f : \mathbb{R}^D \to \mathbb{R} : f(\mathbf{x}) = y$$

Then the Hessian matrix with respect to $\mathbf{x}$, written $\nabla^2 f(\mathbf{x})$ or simply as H is the $D \times D$ matrix of partial derivatives, it is the Jacobian of the gradient

$$\nabla^2 f(\mathbf{x}) = J \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_D} \end{pmatrix}^T = \begin{pmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial x_1^2} & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_D} \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_D} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_D \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_D \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_D^2} \end{pmatrix}$$

Be careful, we cannot take the gradient of a vector but only the Jacobian of the gradient transpose, the relatiion is

$$H(f(\mathbf{x}) = J(\nabla(f(\mathbf{x}))^T$$

Numerical Solution

- Consider a continuously differentiable function $y = f(x)$
- The function has extreme points for $0 = f'(x)$
- We can determine the solution numerically, by approximating the real zeros $f'(x) = 0$

Gradient Descent for one Dimension

The gradient descent for one dimension is

$$x_{n+1} = x_n - \eta \cdot f'(x_n)$$

$$-\eta \cdot f'(x_n) = (x_{n+1} - x_n)$$

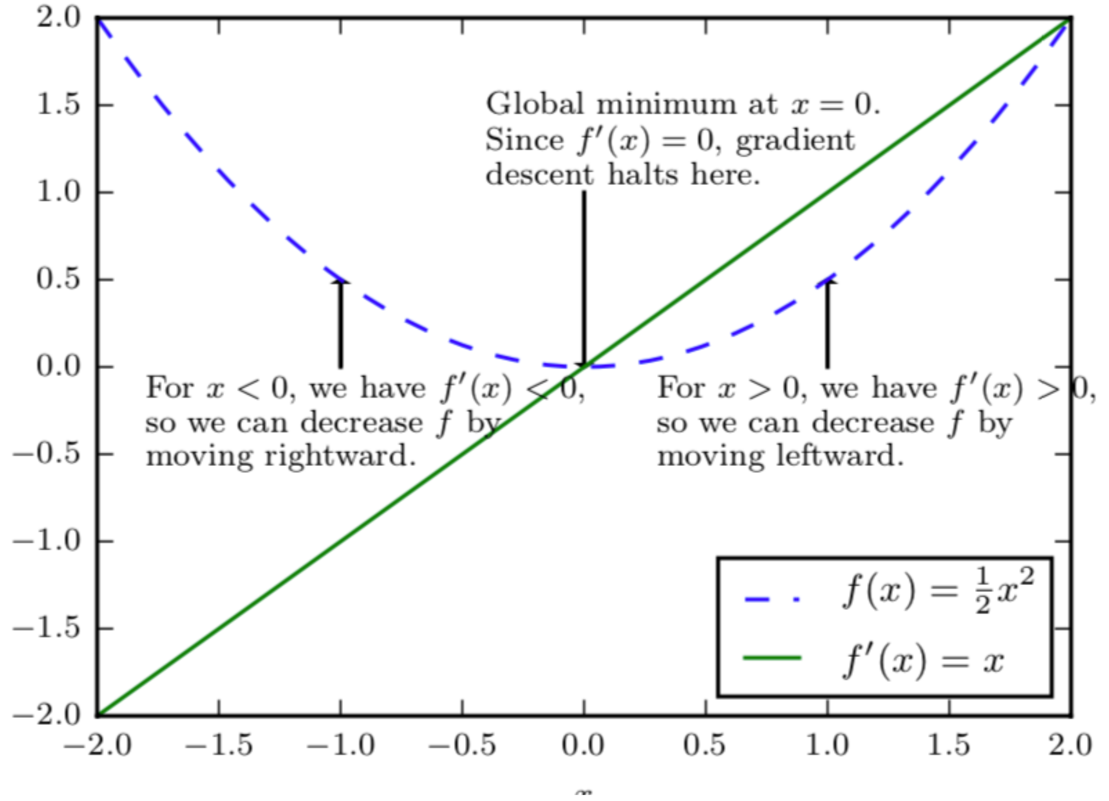
By first-order Taylor series expansion we have

$$f(x_{n+1}) \approx f(x_n) + f'(x_n) \cdot (x_{n+1} - x_n) = f(x_n) - \eta f'(x_n)^2$$

and it follows

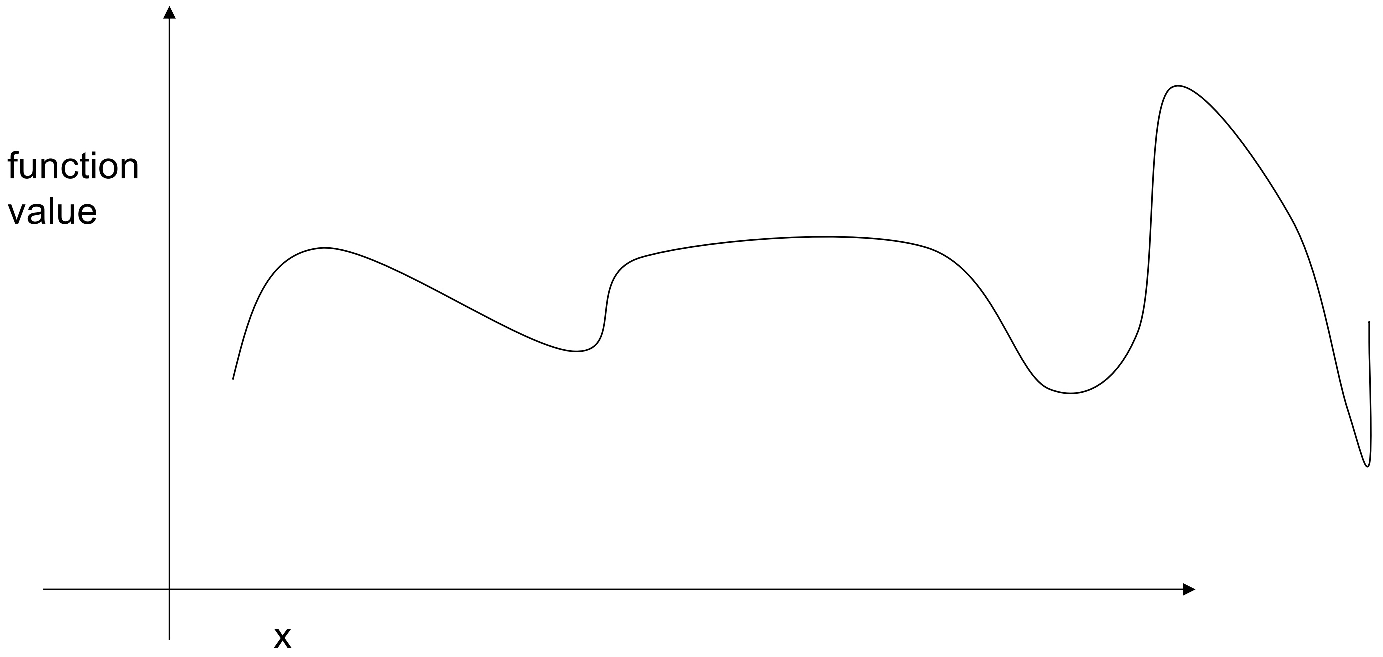
$$f(x_{n+1}) \leq f(x_n)$$

Gradient Descent



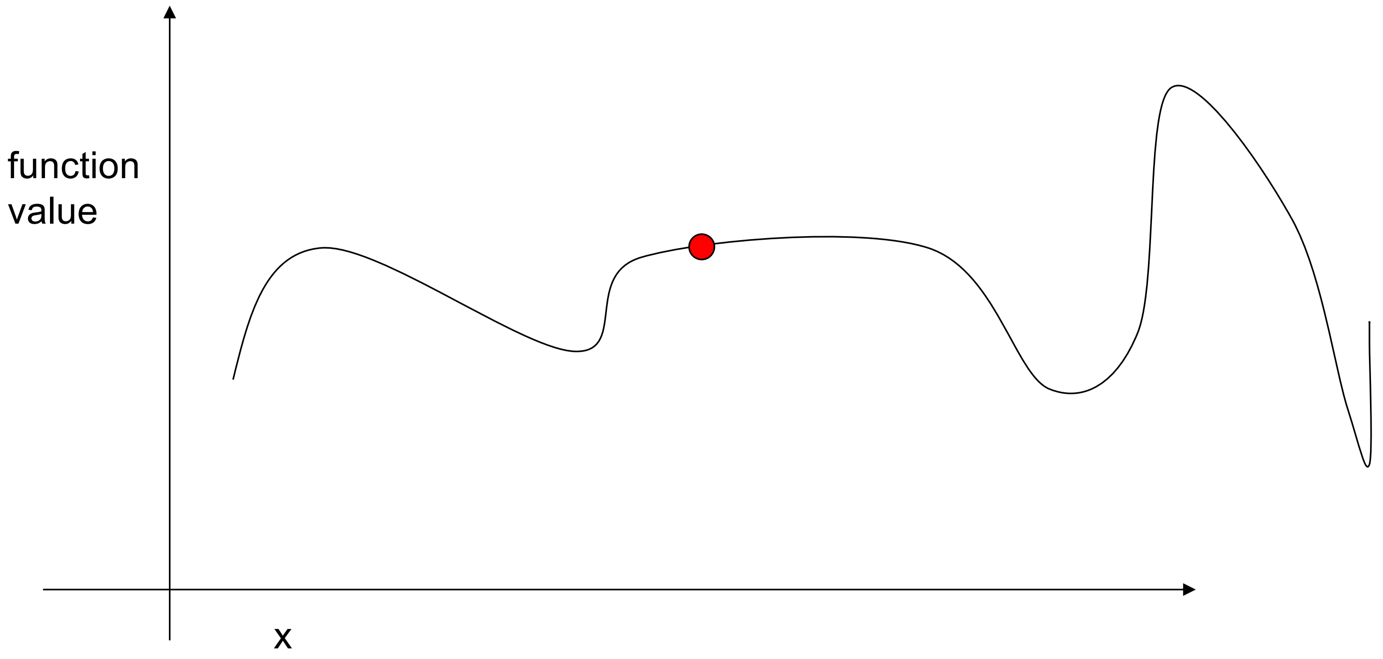
Simple Example

- One dimension

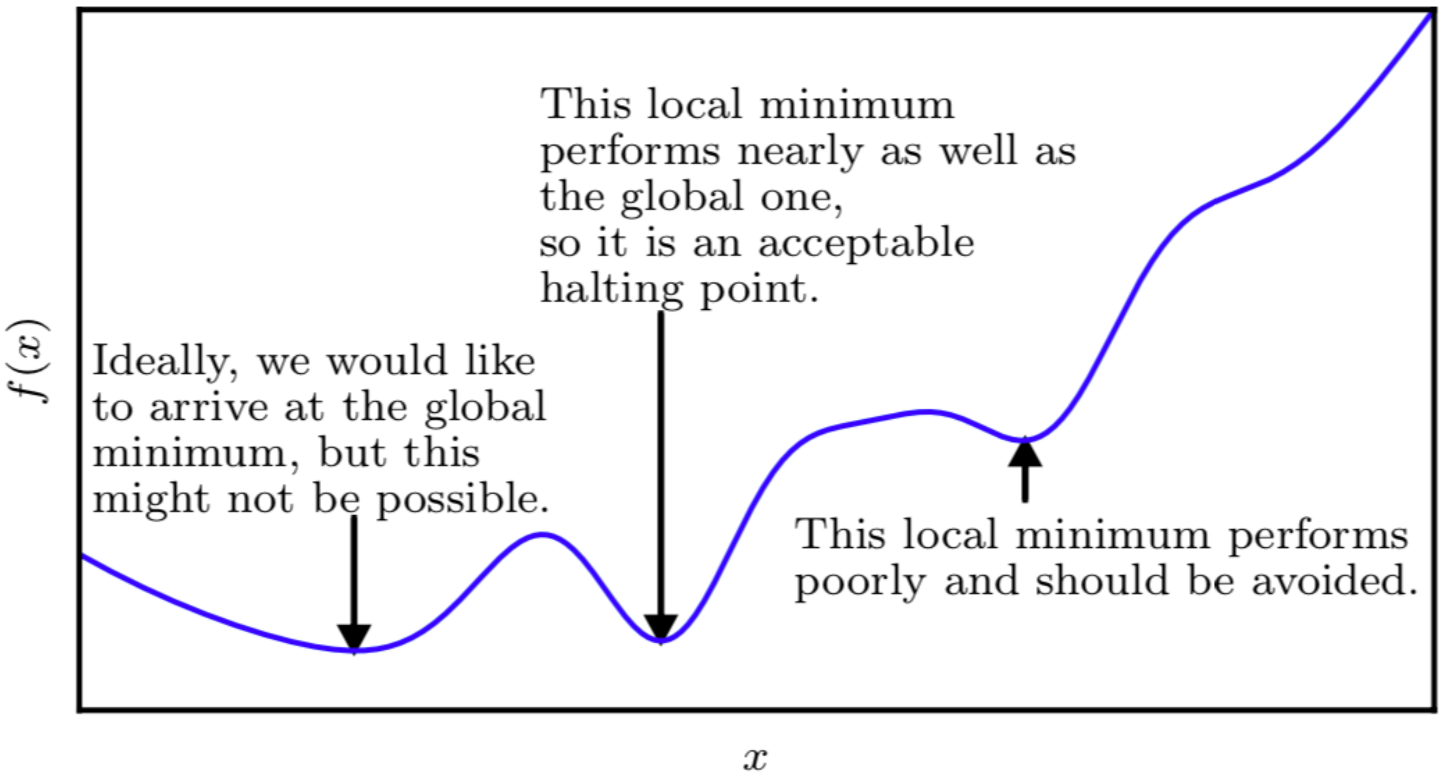


Simple Example

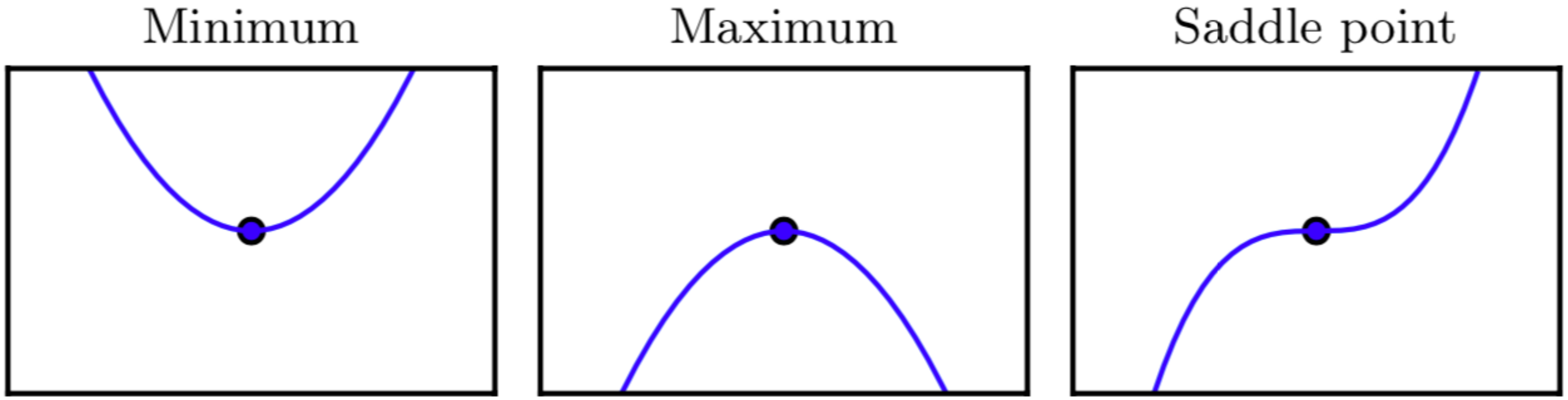
- Move to better state



Approximate Optimization



Critical Points



The gradient descent for D dimensions

Consider a continuously differentiable function $f(\mathbf{x})$.

$$f : \mathbb{R}^D \rightarrow \mathbb{R} : f(\mathbf{x}) = y$$

It maps the vector $\mathbf{x}$ into a real value.
The gradient operator for $\mathbf{x}$ is

$$\nabla = \left[\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \cdots, \frac{\partial}{\partial x_D}\right]^T$$

or

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_D} \end{pmatrix}$$

The gradient descent for D dimensions

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \eta \cdot \nabla f(\mathbf{x}_n)$$

By first-order Taylor series expansion we have

$$f(\mathbf{x}_{n+1}) \approx f(\mathbf{x}_n) + (\nabla f(\mathbf{x}_n))^T \cdot (\mathbf{x}_{n+1} - \mathbf{x}_n)$$

$$f(\mathbf{x}_{n+1}) \approx f(\mathbf{x}_n) - \eta\,(\nabla f(\mathbf{x}_n))^T\,(\nabla f(\mathbf{x}_n))$$

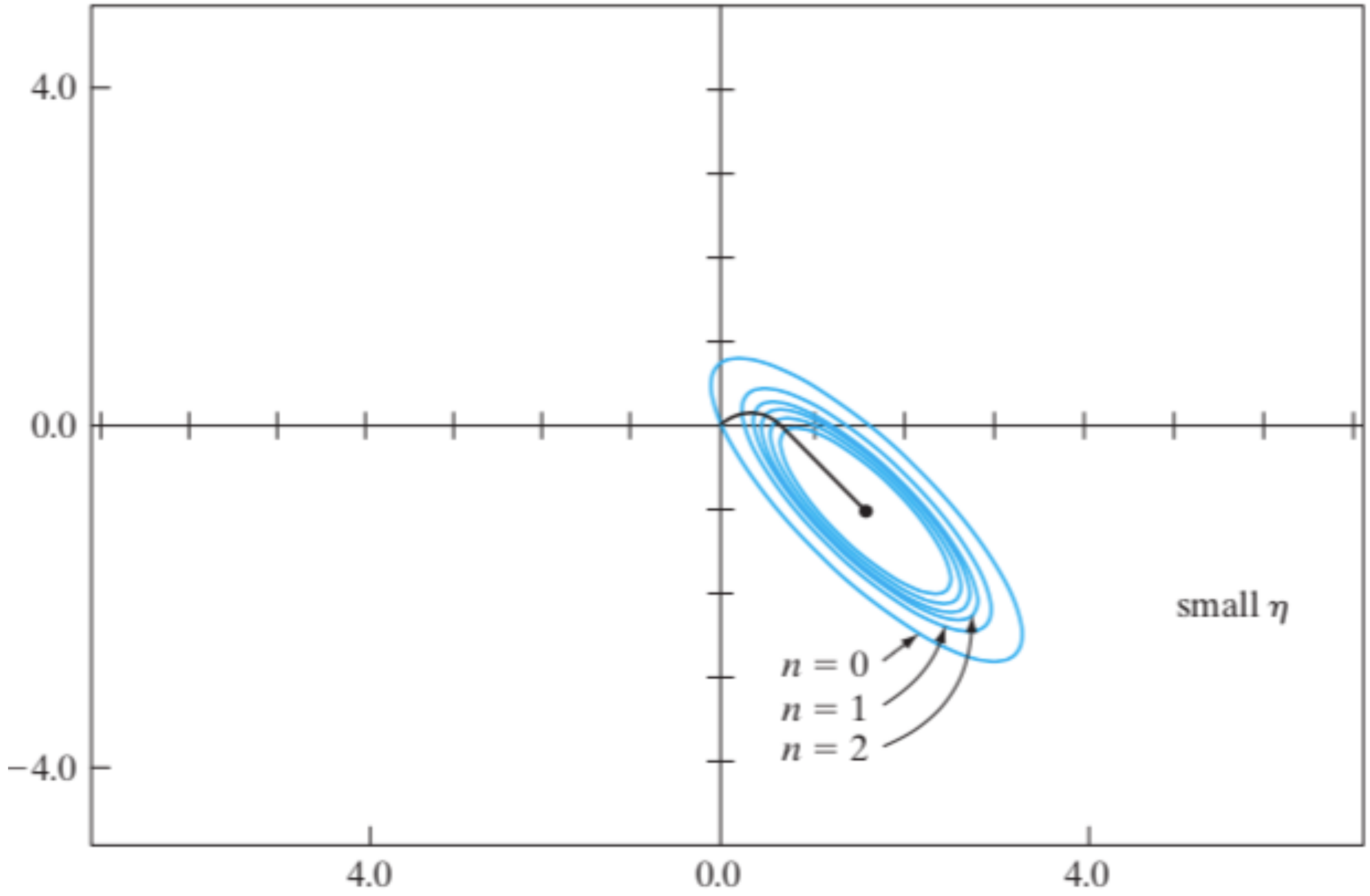
$$f(\mathbf{x}_{n+1}) \approx f(\mathbf{x}_n) - \eta \|\nabla f(\mathbf{x}_n)\|^2$$

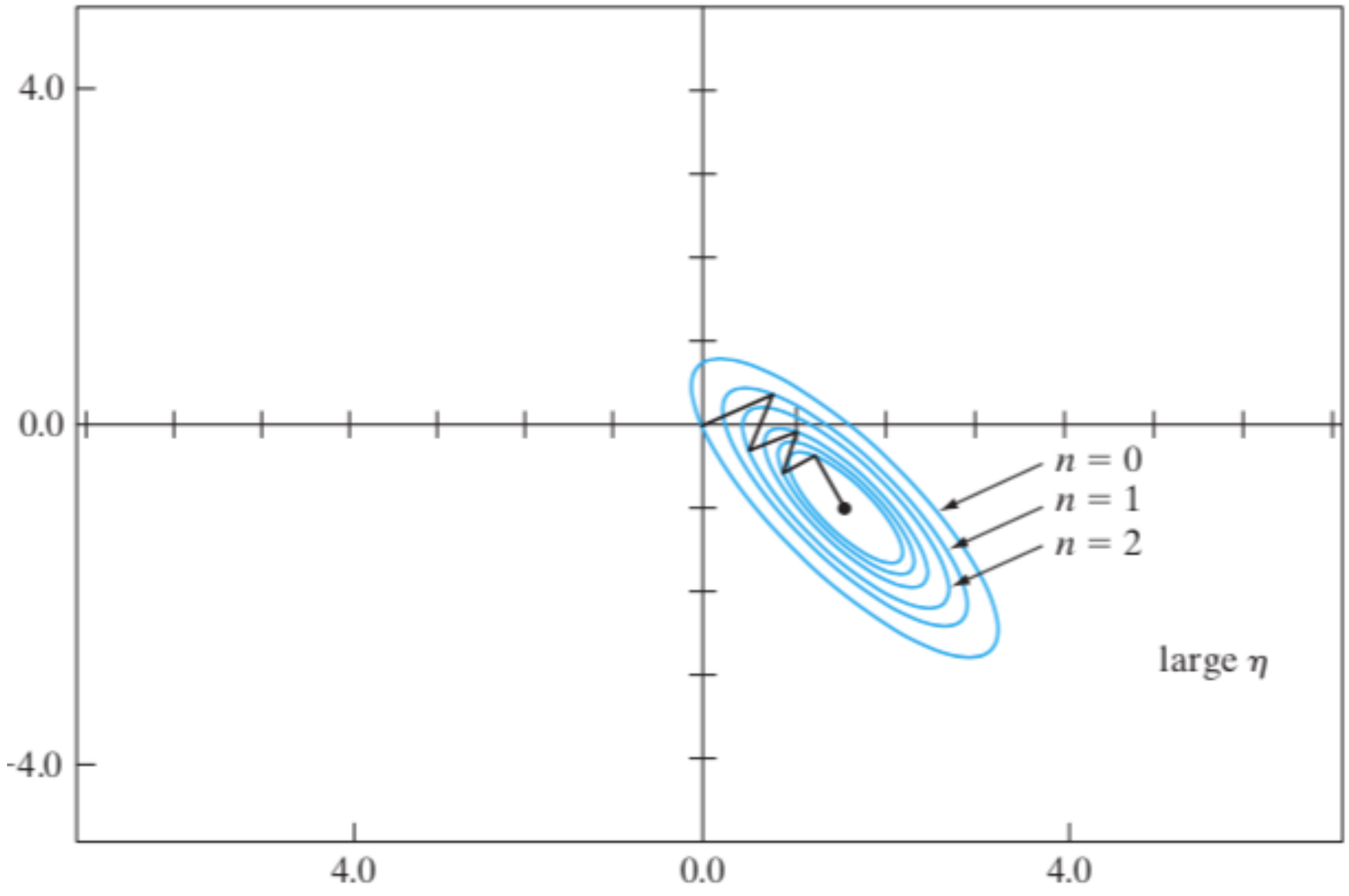
and it follows

$$f(\mathbf{x}_{n+1}) \leq f(\mathbf{x}_n)$$

Learning parameter η

- The method of steepest descent converges to the optimal solution slowly The learning-rate parameter η has a profound influence on its convergence behaviour:
  - When η is small, the transient response of the algorithm follows a smooth path (slow)
  - When η is large, the transient response of the algorithm follows a zigzagging (oscillatory) path
  - When η exceeds a certain critical value, the algorithm becomes unstable (i.e., it diverges)


4.0
0.0
−4.0
4.0
0.0
4.0
small η
$n = 0$
$n = 1$
$n = 2$



4.0
0.0
-4.0
4.0
0.0
4.0
$n = 0$
$n = 1$
$n = 2$
large η

Newton's Method

- Consider a continuously *twice* differentiable function ***f(x)***

$$f : \mathbb{R}^D \to \mathbb{R} : f(\mathbf{x}) = y$$

Then the Hessian matrix with respect to $\mathbf{x}$, written $\nabla^2 f(\mathbf{x})$ or simply as H is the $D \times D$ matrix of partial derivatives,

$$\nabla^2 f(\mathbf{x}) = \begin{pmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial x_1^2} & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_D} \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_D} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_D \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_D \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_D^2} \end{pmatrix}$$

By second-order Taylor series expansion we have

$$f(\mathbf{x}_{n+1}) \approx f(\mathbf{x}_n) + (\nabla f(\mathbf{x}_n))^T \cdot (\mathbf{x}_{n+1} - \mathbf{x}_n) + \frac{(\mathbf{x}_{n+1} - \mathbf{x}_n)^T H_n (\mathbf{x}_{n+1} - \mathbf{x}_n)}{2}$$

We can reformulate to

$$f(\mathbf{x}_{n+1}) - f(\mathbf{x}_n) \approx (\nabla f(\mathbf{x}_n))^T \cdot (\mathbf{x}_{n+1} - \mathbf{x}_n) + \frac{(\mathbf{x}_{n+1} - \mathbf{x}_n)^T H_n (\mathbf{x}_{n+1} - \mathbf{x}_n)}{2}$$

Newton’s Method

We minimise the resulting change with

$$0 = (\nabla f(\mathbf{x}_n))^T \cdot (\mathbf{x}_{n+1} - \mathbf{x}_n) + \frac{(\mathbf{x}_{n+1} - \mathbf{x}_n)^T H_n (\mathbf{x}_{n+1} - \mathbf{x}_n)}{2}$$

$$0 = (\nabla f(\mathbf{x}_n))^T + \frac{(\mathbf{x}_{n+1} - \mathbf{x}_n)^T H_n}{2}$$

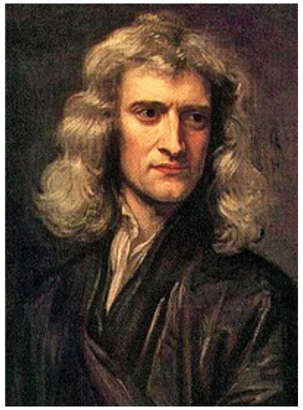
we minimise when

$$0 = \nabla f(\mathbf{x}_n) + H_n (\mathbf{x}_{n+1} - \mathbf{x}_n)$$

$$(\mathbf{x}_{n+1} - \mathbf{x}_n) = -H_n^{-1} \cdot \nabla f(\mathbf{x}_n)$$

and we get the update rule as

$$\mathbf{x}_{n+1} = \mathbf{x}_n - H_n^{-1} \cdot \nabla f(\mathbf{x}_n)$$



- Compared to the previous rule the is no learning parameter

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \eta \cdot \nabla f(\mathbf{x}_n)$$

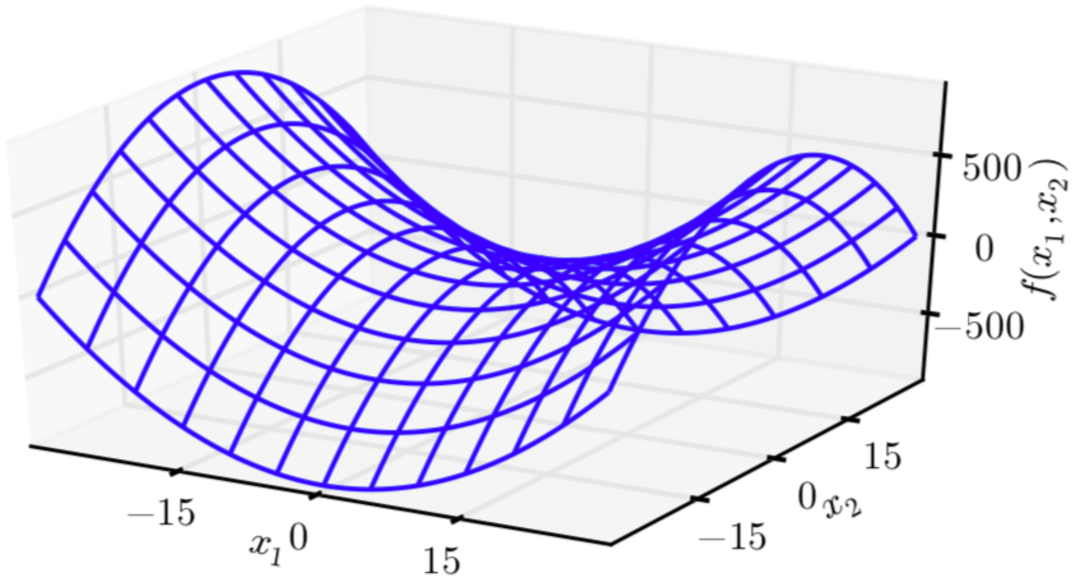
  - Optimization algorithms that use only the gradient, such as gradient descent, are called **first-orde**r optimisation algorithms.

$$\mathbf{x}_{n+1} = \mathbf{x}_n - H_n^{-1} \cdot \nabla f(\mathbf{x}_n)$$

  - Optimization algorithms that also use the Hessian matrix, such as Newton’s method, are called **second-order** optimisation algorithms

- Newton’s method converges quickly asymptotically and does not exhibit the zigzagging behaviour that sometimes characterises the method of steepest descent
- Iteratively updating the approximation and jumping to the minimum of the approximation can reach the critical point much faster than gradient descent
- When f(x) is a positive definite quadratic function, Newton’s method consists of once to jump to the minimum of the function directly.

- However: This is a useful property near a local minimum, but it can be a harmful property near a saddle point



- A major limitation of Newton’s method is its computational complexity
- Newton’s method to work, the Hessian H_n has to be a positive definite matrix for all n.

$$\boldsymbol{x}^T H_n \boldsymbol{x} > 0$$

- Unfortunately, in general, there is no guarantee that H_n is positive definite at every iteration of the algorithm.

Numerical Computation

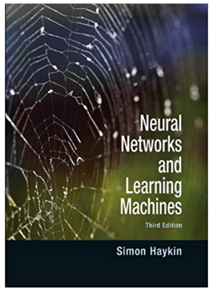
- Machine learning algorithms require a high amount of numerical computation
- Evaluating a mathematical function on computer can be difficult when the function involves real numbers, which cannot be represented precisely
- We need to represent real numbers with a finite number of bits.
- Rounding errors can cause algorithms that work in theory to fail in practice

- Underflow
  - Numbers near zero are rounded to zero
  - Problem, we cannot **divide** by zero. Probabilities become zero.
- Overflow
  - Overflow occurs when huge numbers become negative. No negative **logarithm.**

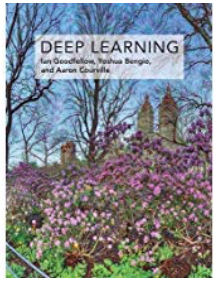
What’s next?

- Machine Learning as Optimization Problem
- Example, Linear and Non-Linear Regression

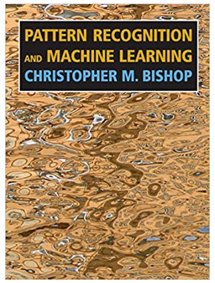
Literature



- Simon O. Haykin, Neural Networks and Learning Machine, (3rd Edition), Pearson 2008
  - Chapter 3

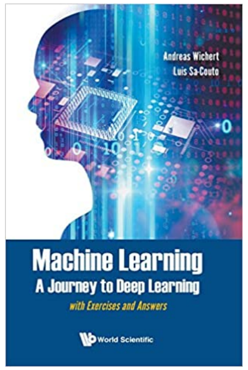


- Deep Learning, I. Goodfellow, Y. Bengio, A. Courville MIT Press 2016
  - Chapter 2, Chapter 4



- Christopher M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer 2006
  - Section *1.4*

Literature



- Machine Learning - A Journey to Deep Learning, A. Wichert, Luis Sa-Couto, World Scientific, 2021
  - Chapter 3