Introduction to Linear Algebra and Convex Optimization

Dr Amit Sethi, IIT Bombay

Module objectives

- Become familiar with linear algebra concepts

- Understand how dimensions play a role in linear algebra

- Understand types of functions

- Appreciate optimization objectives

- Understand the basics of convex optimization

Outline

- Scalars, vectors, matrices, tensors

- Linear operations on vectors and matrices

- Vector and matrix properties

- Functions and derivatives

- Optimizing a continuous function

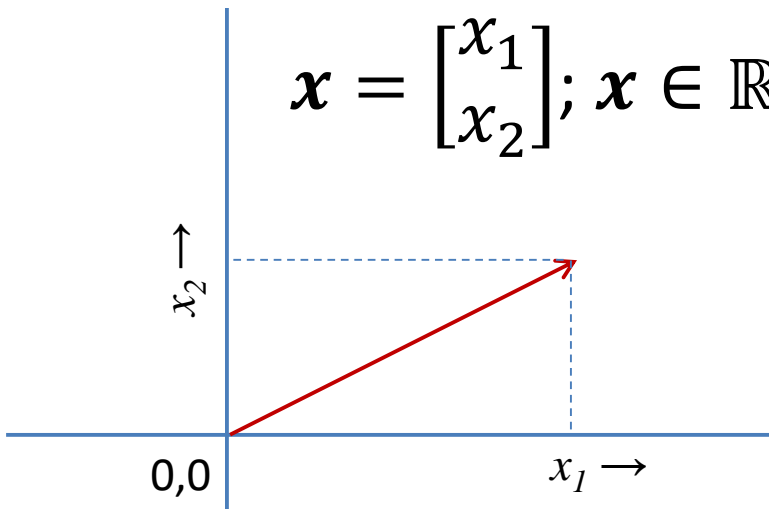# Scalars, vectors, matrices and tensors

$$x; \quad x \in \mathbb{R}$$

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{bmatrix}$$

$$\boldsymbol{X} \in \mathbb{R}^{M \times N}$$

Vectors, matrices, and tensors represent
multi-dimensional ordered data.
Bold-face is used for vectors and matrices.

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}; \boldsymbol{x} \in \mathbb{R}^N$$

$$\begin{bmatrix} x_{111} & x_{121} & x_{131} \\ x_{211} & x_{221} & x_{231} \end{bmatrix}$$
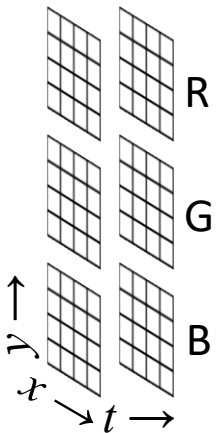
$$\begin{bmatrix} x_{112} & x_{122} & x_{132} \\ x_{212} & x_{222} & x_{232} \end{bmatrix}$$

$$\boldsymbol{X} \in \mathbb{R}^{L \times M \times N \times \cdots}$$

An example from videos

- R-value (red) of one pixel is a scalar
- RGB values of one pixel is a vector
- A row of R-values is a vector
- A frame of R-values is a matrix
- A frame of RGB values is a 3-d tensor
- A video of R-value frames is a 3-d tensor
- A video of RGB values and several frames is a 4-d tensor

N-D array

- 1-d, 2-d, and higher-d arrays are used to represent vectors, matrices, and tensors
- Sometimes n-d arrays are also used to represent scalars
- An array is a data structure, not a mathematical entity
- 1-d array is used interchangeably with a vector
- 2-d array is used interchangeably with a matrix

Outline

- Scalars, vectors, matrices, tensors

- Linear operations on vectors and matrices

- Vector and matrix properties

- Functions and derivatives

- Optimizing a continuous function

Scalar multiplication

ax

$$aX = \begin{bmatrix} ax_{11} & ax_{12} & ax_{13} \\ ax_{21} & ax_{22} & ax_{23} \end{bmatrix}$$

Scalar scales.

$$a\boldsymbol{x} = \begin{bmatrix} ax_1 \\ ax_2 \end{bmatrix}$$

$$\begin{bmatrix} ax_{111} & ax_{121} & ax_{131} \\ ax_{211} & ax_{221} & ax_{231} \end{bmatrix}$$

$$\begin{bmatrix} ax_{112} & ax_{122} & ax_{132} \\ ax_{212} & ax_{222} & ax_{232} \end{bmatrix}$$

Adding a scalar to a vector or a matrix

- Not valid in mathematics, but used in programming and ML

- How it is denoted sometimes: $\boldsymbol{x} + a$

- Mathematically correct operation:

$$\boldsymbol{x} + a \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 + a \\ x_2 + a \end{bmatrix}$$

Point-wise operations require two tensors of the same size

Addition and subtraction require two matrices (or vectors) of the same size: $\mathbb{R}^{M \times N \times \cdots} \times \mathbb{R}^{M \times N \times \cdots} \rightarrow \mathbb{R}^{M \times N \times \cdots}$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \end{bmatrix}; \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 - y_1 \\ x_2 - y_2 \end{bmatrix}$$

Point-wise multiplication is not valid in mathematics, but is important in ML, where it can be used for masking:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \odot \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 y_1 \\ x_2 y_2 \end{bmatrix}$$

Matrix transpose

Transposition is exchange of rows and columns of a matrix (or a vector)

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{bmatrix}; \qquad \boldsymbol{X}^T = \begin{bmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \\ x_{13} & x_{23} \end{bmatrix}$$

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}; \qquad \boldsymbol{x}^T = \begin{bmatrix} x_1 & x_2 \end{bmatrix}$$

Permuting dimensions for higher-dimensional tensors is defined in some python libraries

Matrix multiplication

Matrix multiplication requires matching of the columns of one matrix with the rows of another: $\mathbb{R}^{M \times N} \times \mathbb{R}^{N \times P} \to \mathbb{R}^{M \times P}$

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{bmatrix}; \quad Y = \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \\ y_{31} & y_{32} \end{bmatrix}$$

$$XY = \begin{bmatrix} x_{11}y_{11} + x_{12}y_{21} + x_{13}y_{31} & x_{11}y_{12} + x_{12}y_{22} + x_{13}y_{32} \\ x_{21}y_{11} + x_{22}y_{21} + x_{23}y_{31} & x_{21}y_{12} + x_{22}y_{22} + x_{23}y_{32} \end{bmatrix}$$

$$\boldsymbol{x} . \boldsymbol{y} = \boldsymbol{x}^T \boldsymbol{y} = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = x_1 y_1 + x_2 y_2$$

$$\boldsymbol{x} . \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1^2 + x_2^2$$

In general: $\boldsymbol{AB} \neq \boldsymbol{BA}$

Outline

- Scalars, vectors, matrices, tensors

- Linear operations on vectors and matrices

- Vector and matrix properties

- Functions and derivatives

- Optimizing a continuous function

Vector norm

- L$_p$ norm of a vector: $\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{d}|x_i|^p\right)^{\frac{1}{p}}$

- By default, norm means L$_2$ norm, which is the length of a vector (or Euclidean distance)
  - $\sqrt{x_1^2 + x_2^2 + \cdots + x_d^2}$
- L$_1$ norm is the sum of absolute values (or Manhattan distance)
  - $|x_1| + |x_2| + \cdots + |x_d|$
- L$_\infty$ norm is max absolute value of all dimensions. Why? Because max raised to the power infinity will dominate all other values in the sum

Cosine between two vectors

Cosine of the angle between x and y is $\dfrac{x.y}{\|x\|\|y\|}$
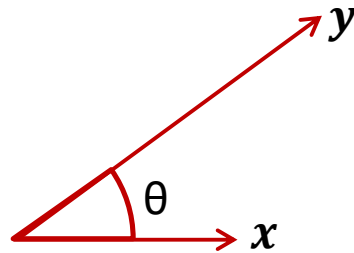
If cosine is zero then the vectors are orthogonal

That is, their dot product is zero

Cosine is a measure of similarity (direction) of vectors, when we want to ignore their magnitudes

Some special vectors and matrices

Zero vector: $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ Ones: $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

Diagonal matrix:
$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$$

One-hot bit: $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ $\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

Identity matrix:
$$\boldsymbol{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$; s.t. $x_1{}^2 + x_2{}^2 + x_3{}^2 = 1$

$$\boldsymbol{AI} = \boldsymbol{IA} = \boldsymbol{A}$$

Linear independence

- A set of vectors are linearly independent if none of them can be expressed as a linear combination of the others

- Example: $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$, and $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

- But not: $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$, and $\begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$

- And nor: $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}$

Rank of a matrix

- For a square matrix: A

- Rank is the minimum number of vectors needed to express its columns as their linear combination

- Example: $\begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 2 & 0 & 2 \end{bmatrix}$ has rank 2, and $\begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 2 \\ 2 & 0 & 0 \end{bmatrix}$

  has rank 3 (a.k.a full rank)

- Eigenvalue analysis not only finds the rank, it also finds an orthonormal set of such vectors
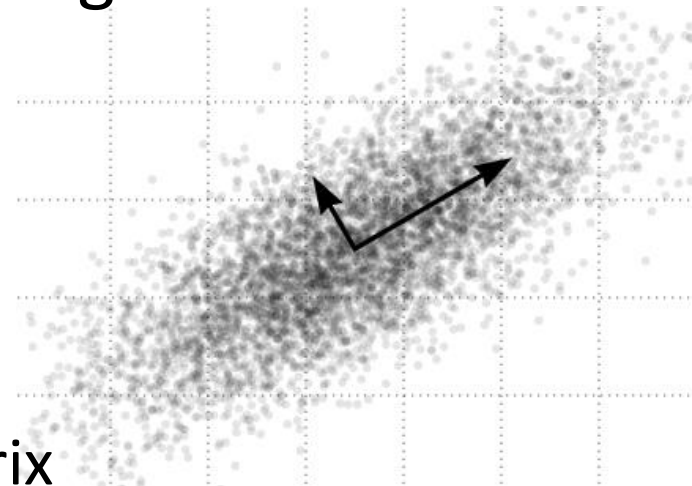 - Sort of like, modes of variation

Eigen decomposition of a square matrix

- Eigen decomposition expresses a matrix as:
$$A = U\Lambda U^{-1}$$

- Where, Λ is a diagonal matrix containing eigenvalues (usually sorted in descending order)

- And, U is a matrix of orthonormal eigenvectors $U = [u_1 \dots u_N]$ such that $u_i.u_i = 1, \ u_i.u_j = 0$, $Au_i = \lambda_i u_i, \ \ \forall i \neq j$

- Number of non-zero eigenvalues is called the rank of the matrix (number of independent dimensions)

Principal component analysis

- Finding orthogonal directions of maximum variance of a set of data points

- Rotating the reference frame

- Finding which directions can be neglected

- Steps:
  - Center points (subtract mean)
  - Compute covariance matrix
  - Eigen decompose covariance matrix
  - Find projections (dot products) along eigenvectors corresponding to highest eigenvalues

Dimension reduction using PCA

Original data

Mean subtraction

Principal components
of the covariance matrix

Projection

Projection

Dimension Reduction
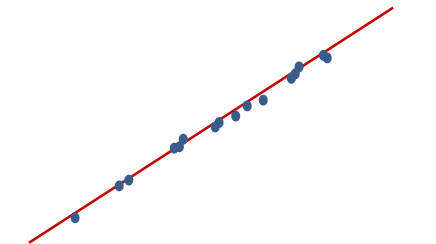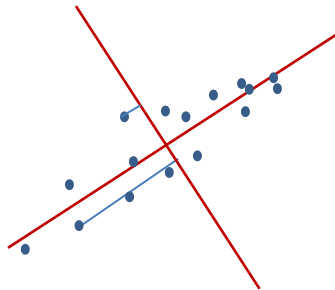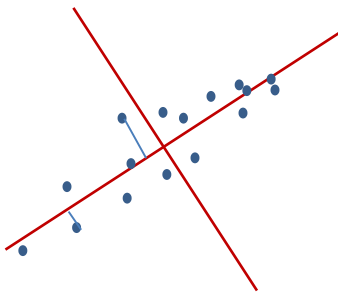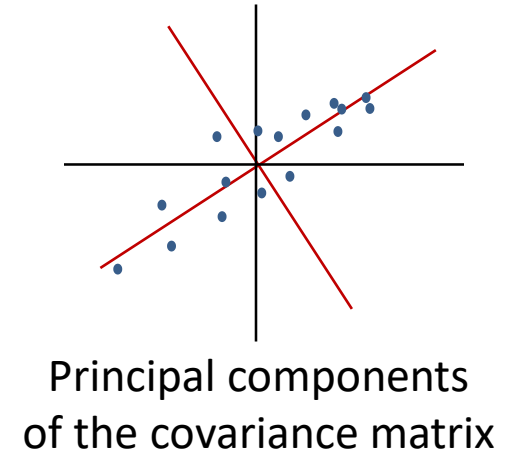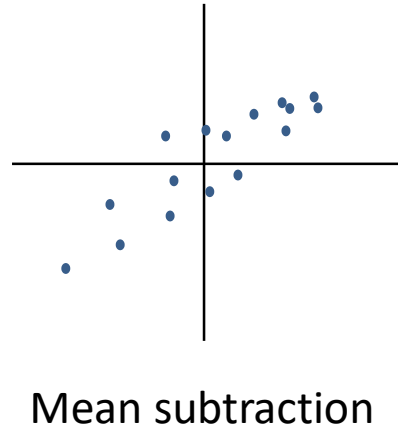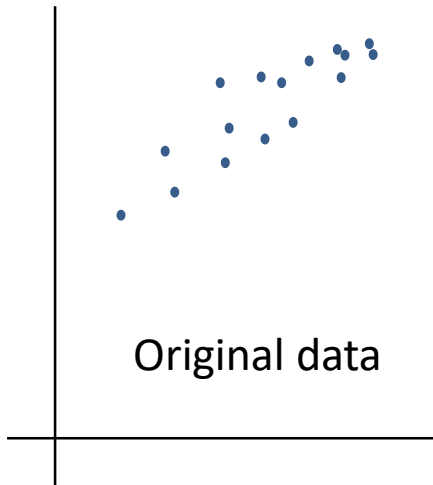
PCA FAQs

- Why perform mean-centering?
  - Co-variance is defined on mean-centered data
- What is special about principal component directions?
  - The directions are orthogonal, just like original axes
  - The data is decorrelated – in some sense independent – along these directions
  - Co-variance in the new directions is $\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$
  - Notice the zero terms in off-diagonal entries
- Which can we ignore some dimensions?
  - The ones along which the variance is the smallest
  - This will lead to the smallest mean square error between original and approximated data

Inverse and pseudo inverse

- For square matrices that are full rank (all rows and columns are linearly independent of each other) an inverse matrix exists such that:

- $AA^{-1} = A^{-1}A = I$

- Similar to:
  $$xx^{-1} = x^{-1}x = 1$$

- And just like inverse of 0 does not exist, the inverse of rank deficient matrices does not exist

- For non-square matrices of full rank, there is something called a pseudo inverse (assuming $N_{rows} > N_{columns}$ and rank is $N_{columns}$):

- $A^+ = (A^TA)^{-1}A^T$

- Note that $A^+A = I$

Singular Value Decomposition

- SVD is a generalization of eigendecomposition
- It can be applied to any sized matrix
- It expresses a matrix as: $\boldsymbol{M} = \boldsymbol{U}\,\boldsymbol{\Sigma}\,\boldsymbol{V}^*$
  - $\boldsymbol{U}$ and $\boldsymbol{V}$ are square matrices
  - $\boldsymbol{\Sigma}$ is a diagonal matrix, but need not be square
  - $\boldsymbol{V}^*$ is the conjugate transpose of $\boldsymbol{V}$
  - $\boldsymbol{U}\boldsymbol{U}^*$ and $\boldsymbol{V}\boldsymbol{V}^*$ are identity matrices of different sizes
- Non-zero singular values of $\boldsymbol{M}$ are square root of eigenvalues of $\boldsymbol{M}\boldsymbol{M}^*$ and $\boldsymbol{M}^*\boldsymbol{M}$
- It is used to find finding pseudo inverse and least square fitting

Outline

- Scalars, vectors, matrices, tensors

- Linear operations on vectors and matrices

- Vector and matrix properties

- Functions and derivatives

- Optimizing a continuous function

Function usually is a mapping from a vector to a scalar

- Usual definition of a function:
  - Input $\boldsymbol{x}$
  - Output $y = f(\boldsymbol{x})$
- Examples:
  - $f(x) = w\,x + b$ or $w\,x + b\,1$
  - $f(x) = w_2\,x^2 + w_1\,x^1 + w_0\,x^0$
  - $f(x) = \boldsymbol{w}^T\boldsymbol{x}+b$
  - $f(x) = g(\boldsymbol{w}^T\boldsymbol{x}+b)$, where g is a nonlinear function

# Derivative of a function of a scalar
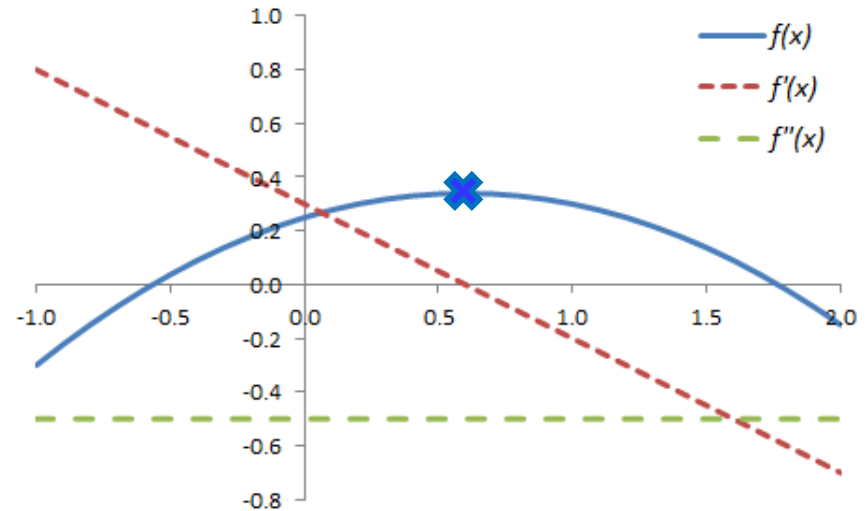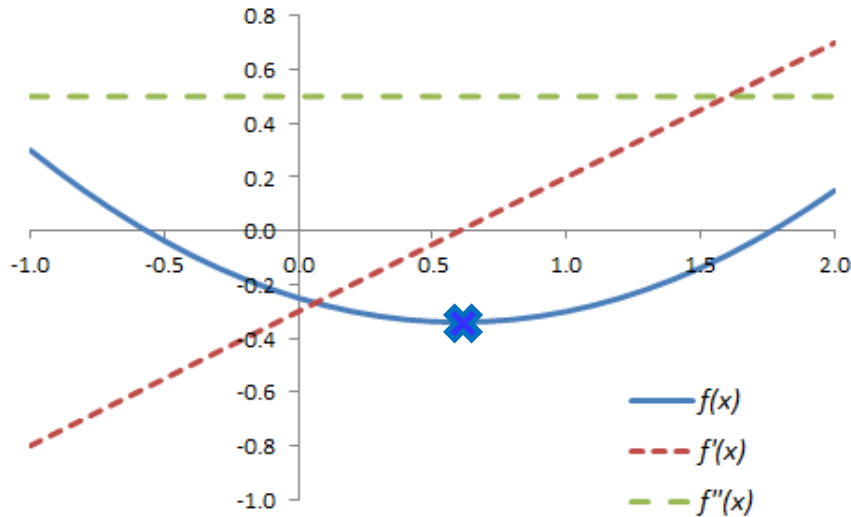
E.g. $f(x) = ax^2 + bx + c$, $f'(x) = 2ax + b$, $f''(x) = 2a$

- Derivative $f'(x) = \dfrac{d\,f(x)}{d\,x}$ is the rate of change of $f(x)$ with x
- It is zero when then function is flat (horizontal), such as at the minimum or maximum of $f(x)$
- It is positive when $f(x)$ is sloping up, and negative when $f(x)$ is sloping down
- To move towards the maxima, taking a small step in a direction of the derivative

Chain rule of differentiation

- Very handy for complicated functions
  - Especially functions of functions
  - E.g. NN outputs are functions of previous layers
- For example: Let $f(x) = g\big(h(x)\big)$
  - Let $y = h(x), z = g(y) = g\big(h(x)\big)$
- Then $f'(x) = \dfrac{d\,z}{d\,x} = \dfrac{d\,z}{d\,y}\dfrac{d\,y}{d\,x} = g'(y)h'(x)$
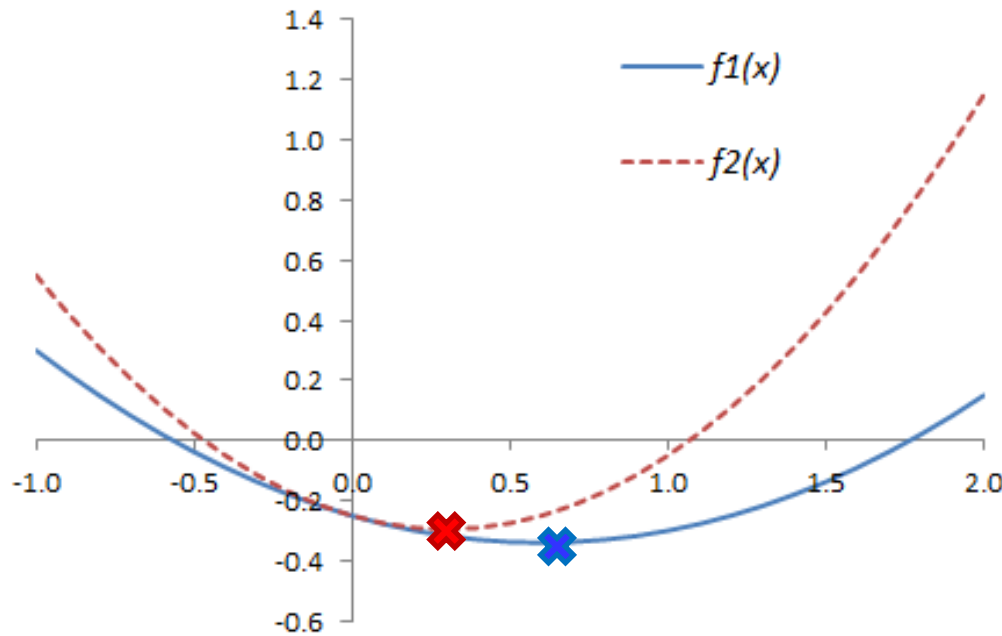- For example: $\dfrac{d\,\sin(x^2)}{d\,x} = 2x\cos(x^2)$

Double derivative
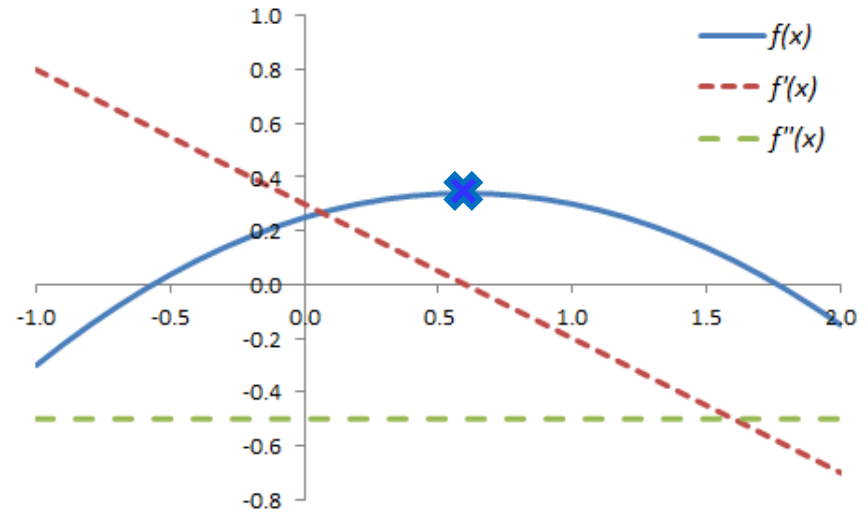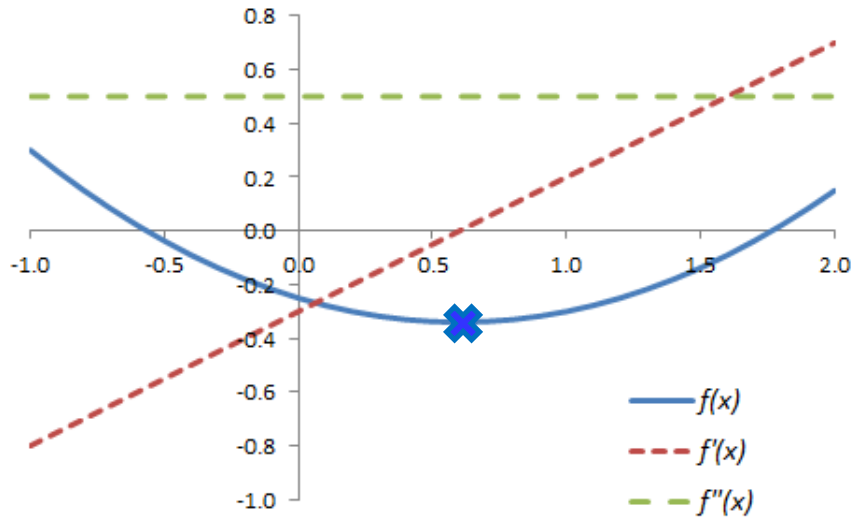
E.g. $f(x) = ax^2 + bx + c, \quad f'(x) = 2ax + b, \quad f''(x) = 2a$

- Double derivative $f''(x) = \dfrac{d^2 f(x)}{d x^2}$ is the derivative of derivative of $f(x)$

- Double derivative is positive for convex functions (have a single minima), and negative for concave functions (have a single maxima)
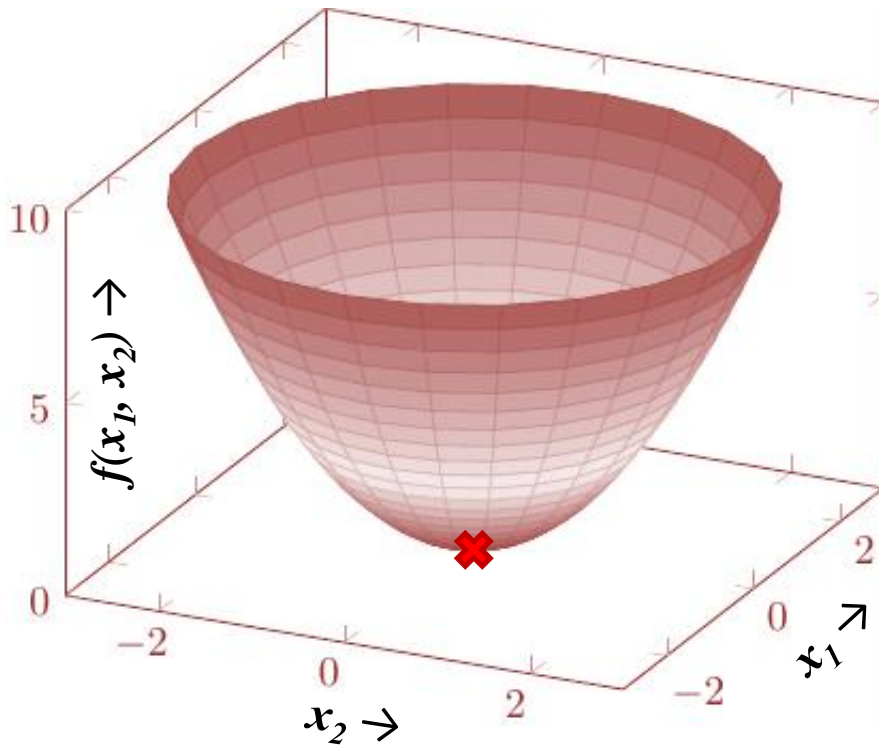
Double derivative

$$f(x) = ax^2 + bx + c,$$
$$f'(x) = 2ax + b,$$
$$f''(x) = 2a$$

- Double derivative tells how far the minima might be from a given point.

- From $x = 0$ the minima is closer for the red dashed curve than for the blue solid curve, because the former has a larger second derivative (its slope reverses faster)
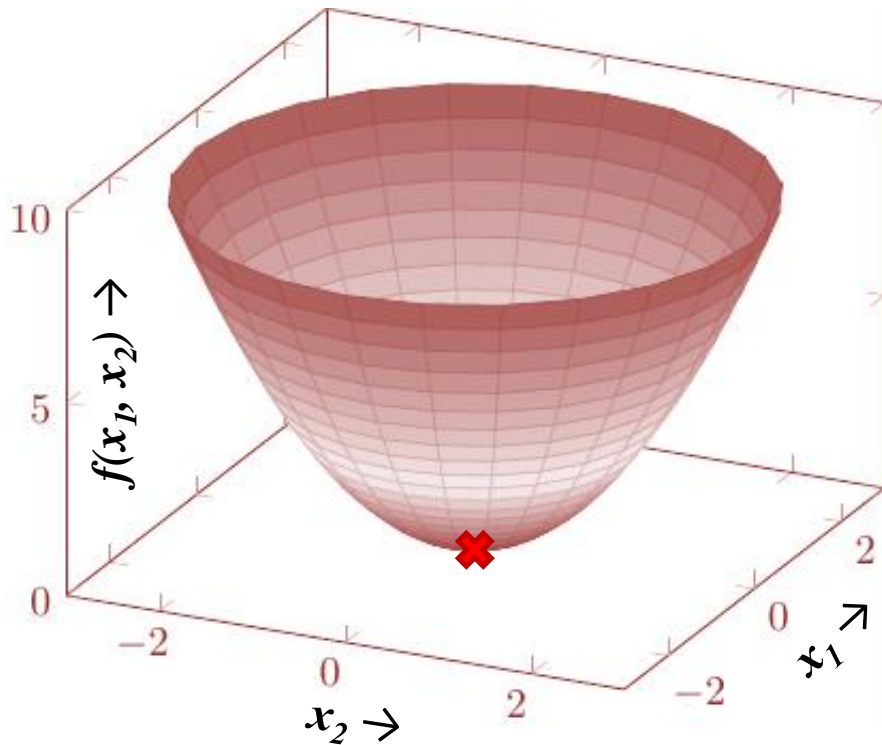
Gradient of a function of a vector

- Derivative with respect to each dimension, holding other dimensions constant
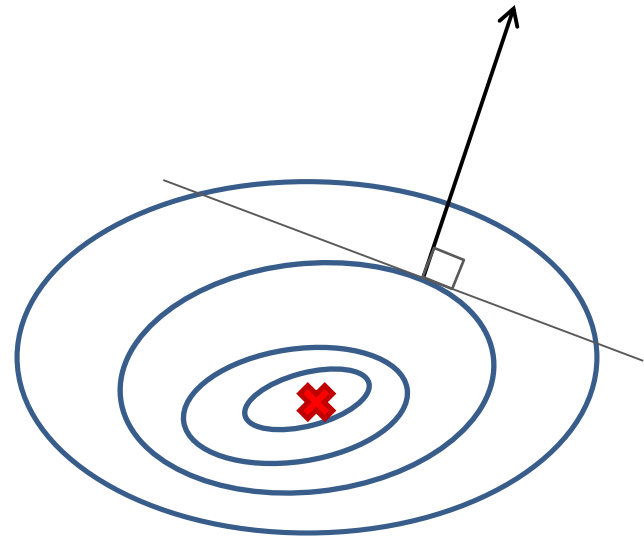
- $\nabla f(\boldsymbol{x}) = \nabla f(x_1, x_2) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \end{bmatrix}$

- At a minima or a maxima the gradient is a zero vector

   The function is flat in every direction

- At a minima or a maxima the gradient is a zero vector

Gradient of a function of a vector

- Gradient gives a direction for moving towards the minima
- Take a small step towards negative of the gradient

Example of gradient

- Let $f(\boldsymbol{x}) = f(x_1, x_2) = 5x_1{}^2 + 3x_2{}^2$

- Then $\nabla f(\boldsymbol{x}) = \nabla f(x_1, x_2) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\[2mm] \dfrac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 10x_1 \\ 6x_2 \end{bmatrix}$

- At a location $(2,1)$ a step in $\begin{bmatrix} 20 \\ 6 \end{bmatrix}$ or $\begin{bmatrix} 0.958 \\ 0.287 \end{bmatrix}$ direction will lead to maximal increase in the function

Hessian of a function of a vector
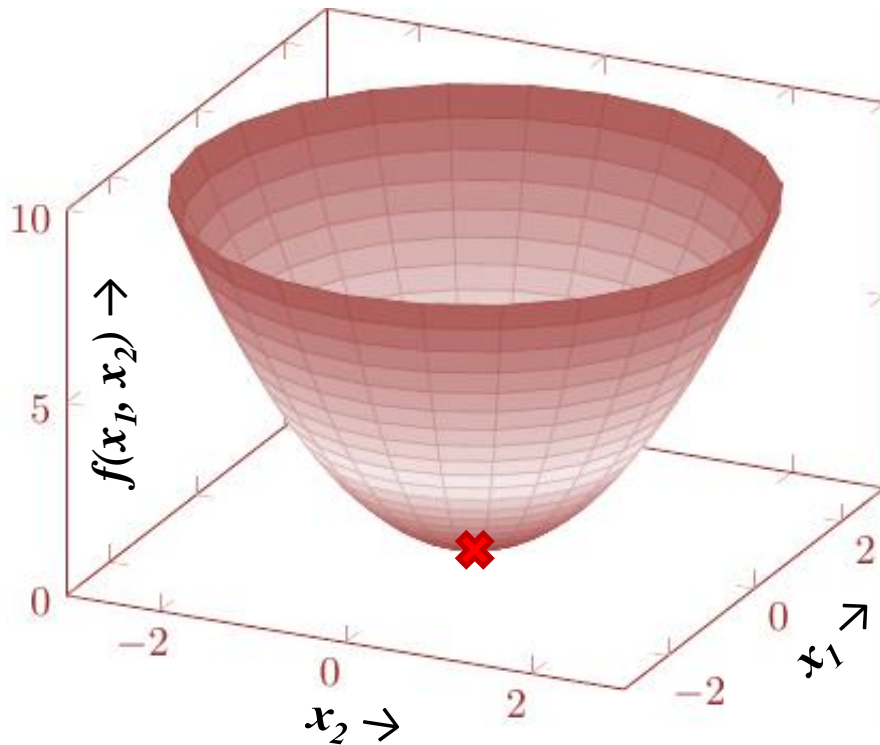
- Double derivative with respect to a pair of dimensions forms the Hessian matrix:

$$\mathbf{H} = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1\,\partial x_n} \\[2ex] \dfrac{\partial^2 f}{\partial x_2\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2\,\partial x_n} \\[2ex] \vdots & \vdots & \ddots & \vdots \\[2ex] \dfrac{\partial^2 f}{\partial x_n\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

- If all eigenvalues of a Hessian matrix are positive, then the function is convex

Example of Hessian

- Let $f(\boldsymbol{x}) = f(x_1, x_2) = 5{x_1}^2 + 3{x_2}^2 + 4x_1 x_2$

- Then $\nabla f(\boldsymbol{x}) = \nabla f(x_1, x_2) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \end{bmatrix} =$

$\begin{bmatrix} 10x_1 + 4x_2 \\ 6x_2 + 4x_1 \end{bmatrix}$

- And, $H(f(\boldsymbol{x})) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial {x_1}^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial {x_2}^2} \end{bmatrix} = \begin{bmatrix} 10 & 4 \\ 4 & 6 \end{bmatrix}$

Vector valued functions and Jacobians

- We often deal with functions that give multiple outputs

- Let $f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix} = \begin{bmatrix} f_1(x_1, x_2, x_3) \\ f_2(x_1, x_2, x_3) \end{bmatrix}$

- Thinking in terms of vector of functions can make the representation less cumbersome and computations more efficient

- Then Jacobian $J(f) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} & \dfrac{\partial f}{\partial x_2} & \dfrac{\partial f}{\partial x_3} \end{bmatrix} =$
$\begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} & \dfrac{\partial f_1}{\partial x_3} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} & \dfrac{\partial f_2}{\partial x_3} \end{bmatrix}$

# Relationship between Hessian, Jacobian, and gradient

- Interestingly, Hessian is the transpose of Jacobian of the gradient
  - Gradient can be thought of as a multi-output function
  - So, its Jacobian makes sense
  - And derivative (Jacobian) of a derivative (gradient) is the second derivative (Hessian transposed)

Outline

- Scalars, vectors, matrices, tensors

- Linear operations on vectors and matrices

- Vector and matrix properties

- Functions and derivatives
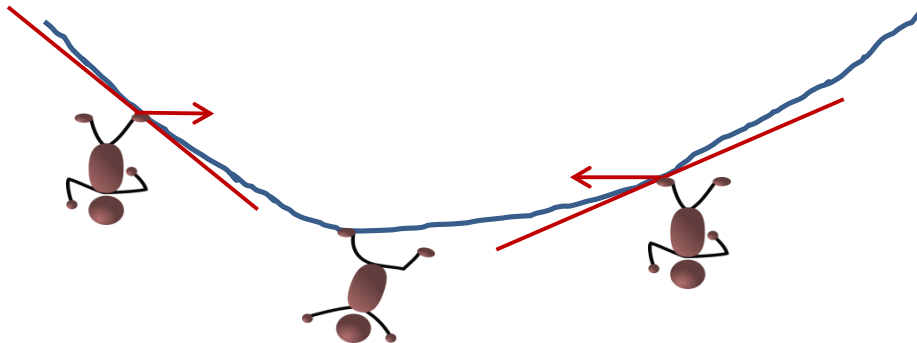
- Optimizing a continuous function

Gradient ascent

- If you didn't know the shape of a mountain
- But at every step you knew the slope
- Can you reach the top of the mountain?
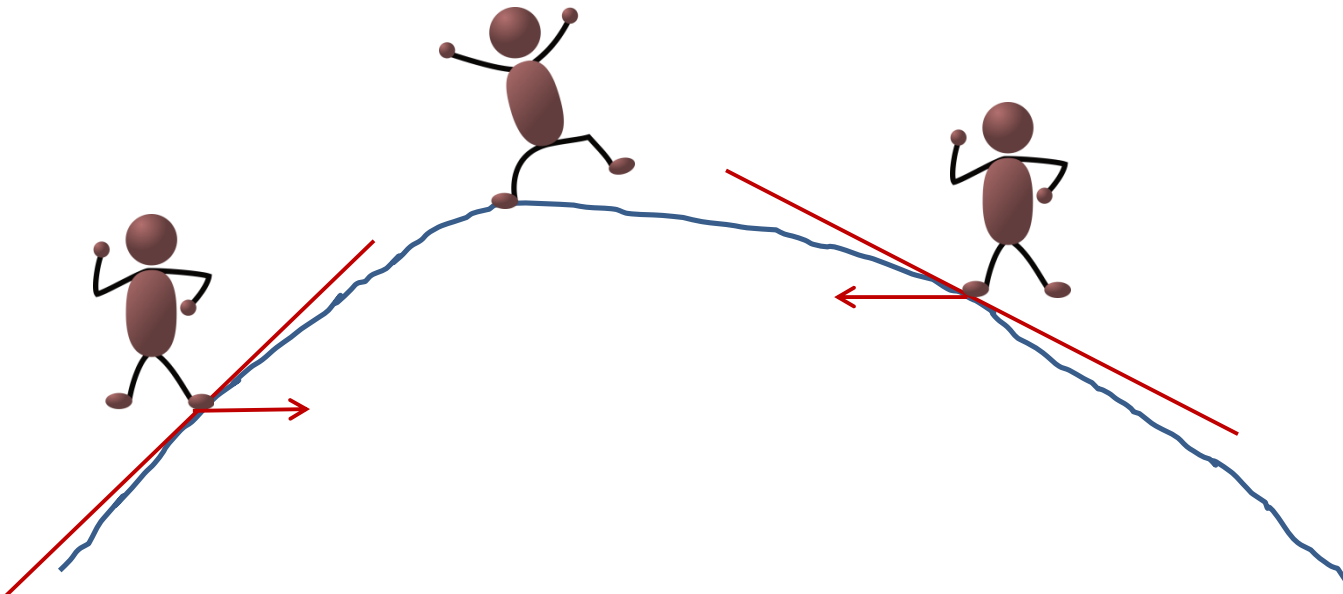
Gradient descent minimizes a function

- At every point, compute
  - $f(x)$
  - Gradient of loss with respect to weights (vector):
    $$\nabla_x f(x)$$
  - Take a step towards negative gradient:
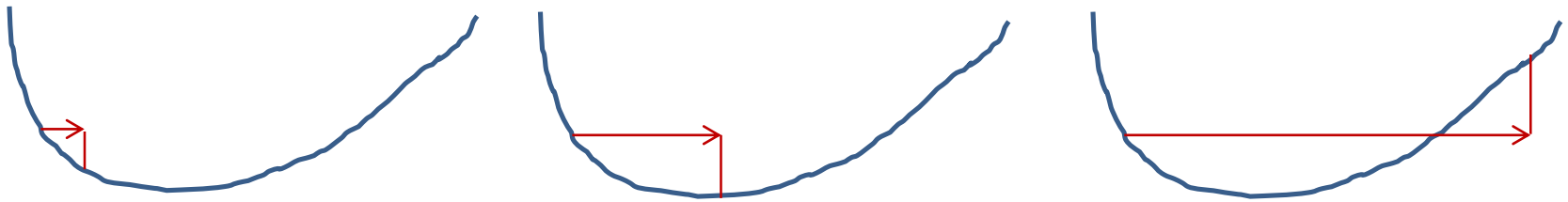 $$x \leftarrow x - \eta \nabla_x f(x)$$

Role of step size and learning rate

- Tale of two loss functions
  - Same value, and
  - Same gradient (first derivative), but
  - Different Hessian (second derivative)
  - Different step sizes needed
- Success not guaranteed for non-convex functions
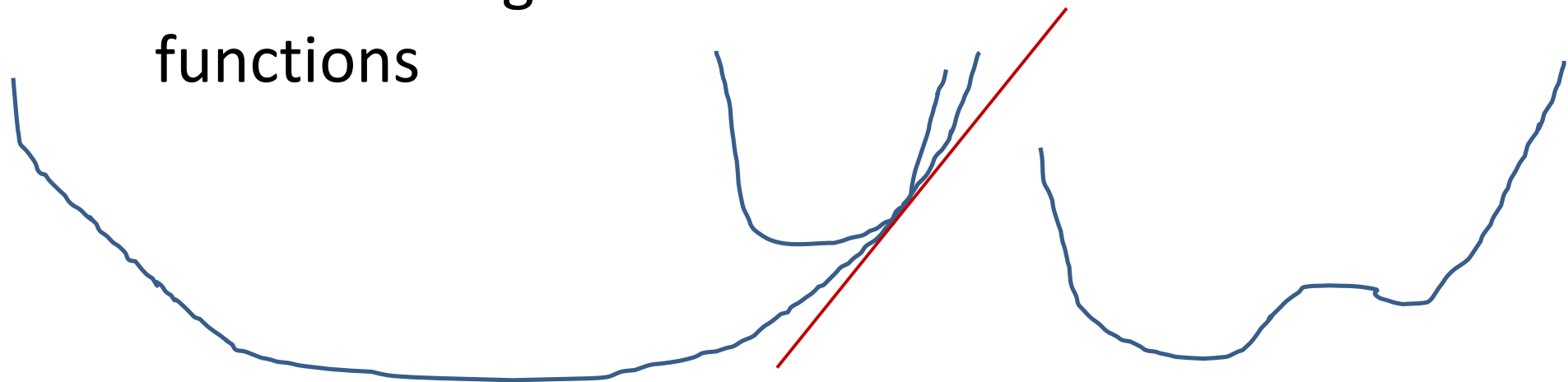
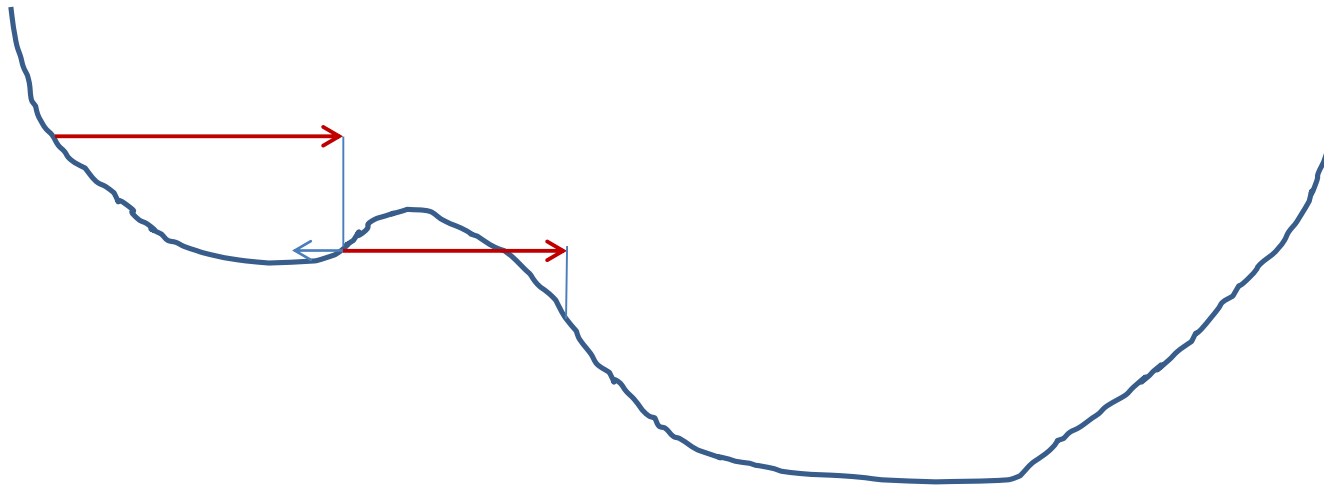# The perfect step size is impossible to guess

- Goldilocks finds the perfect balance only in a fairy tale

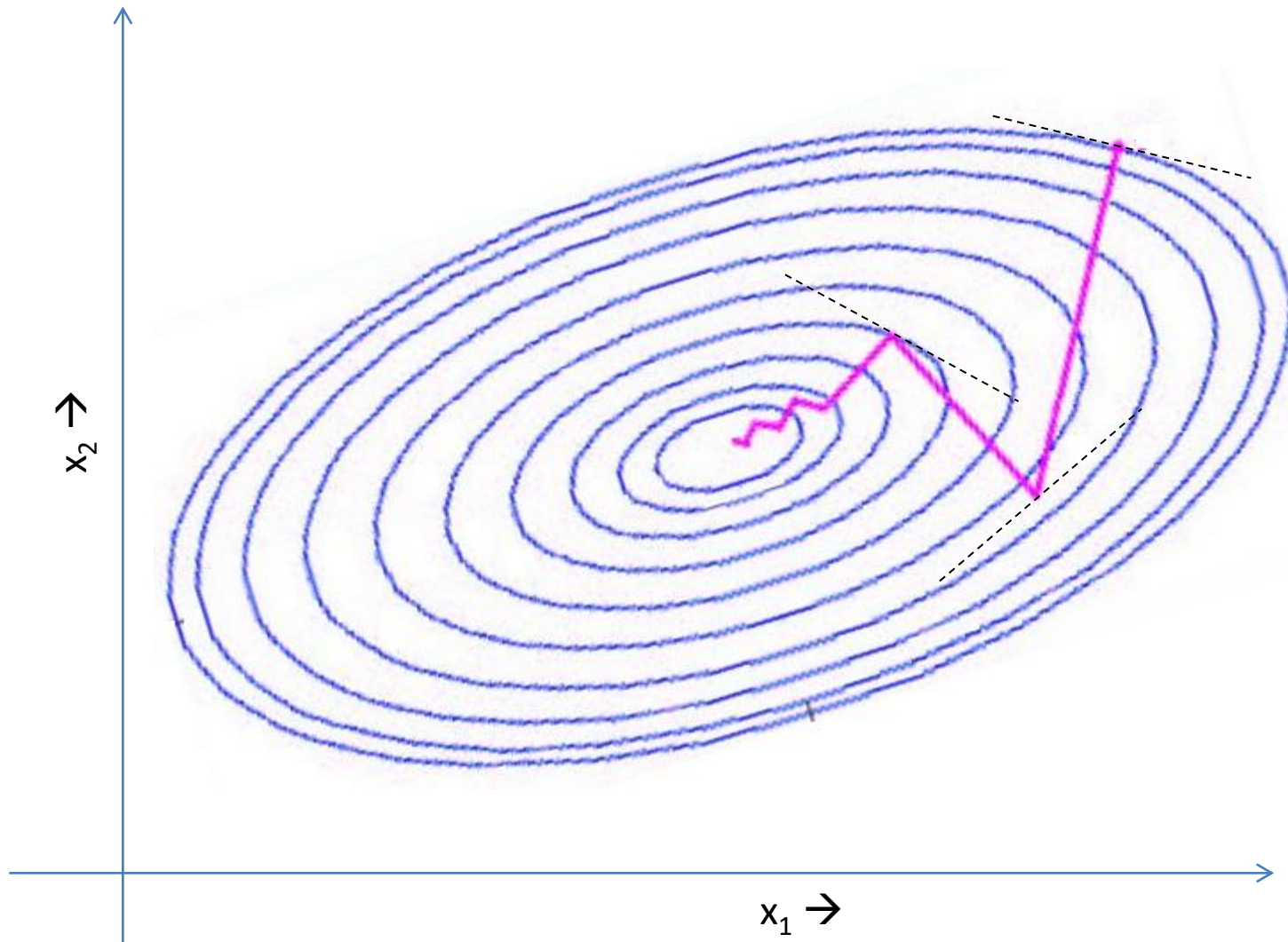- The step size is decided by learning rate η and the gradient

Momentum

- Momentum means using the memory of previous step to build up speed or to slow down with forgetting factor α; $0 \leq \alpha = 1$

$$\Delta \boldsymbol{x}^{(t)} = \alpha \, \Delta \boldsymbol{x}^{(t-1)} - \eta \, \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$$

This story is unfolding in multiple dimensions
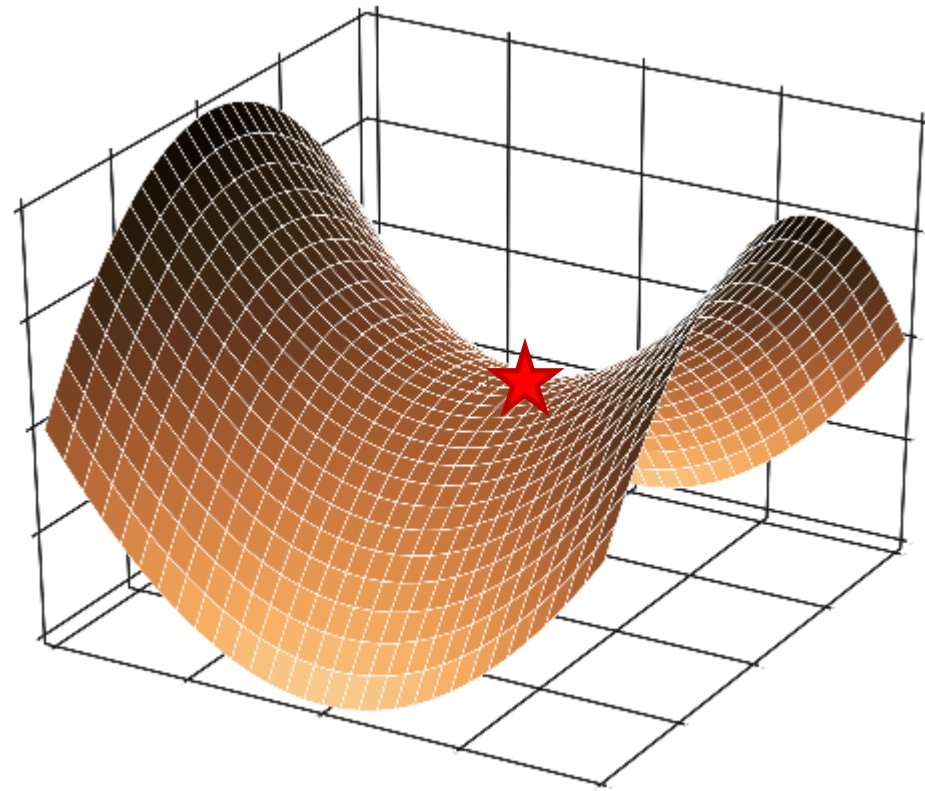
Perfect step size for a paraboloid

- Let $f(x) = ax^2 + bx + c$

- Assuming $a < 0$

- Minima is at: $x^* = -\dfrac{b}{2a}$

- For any x the perfect step would be:

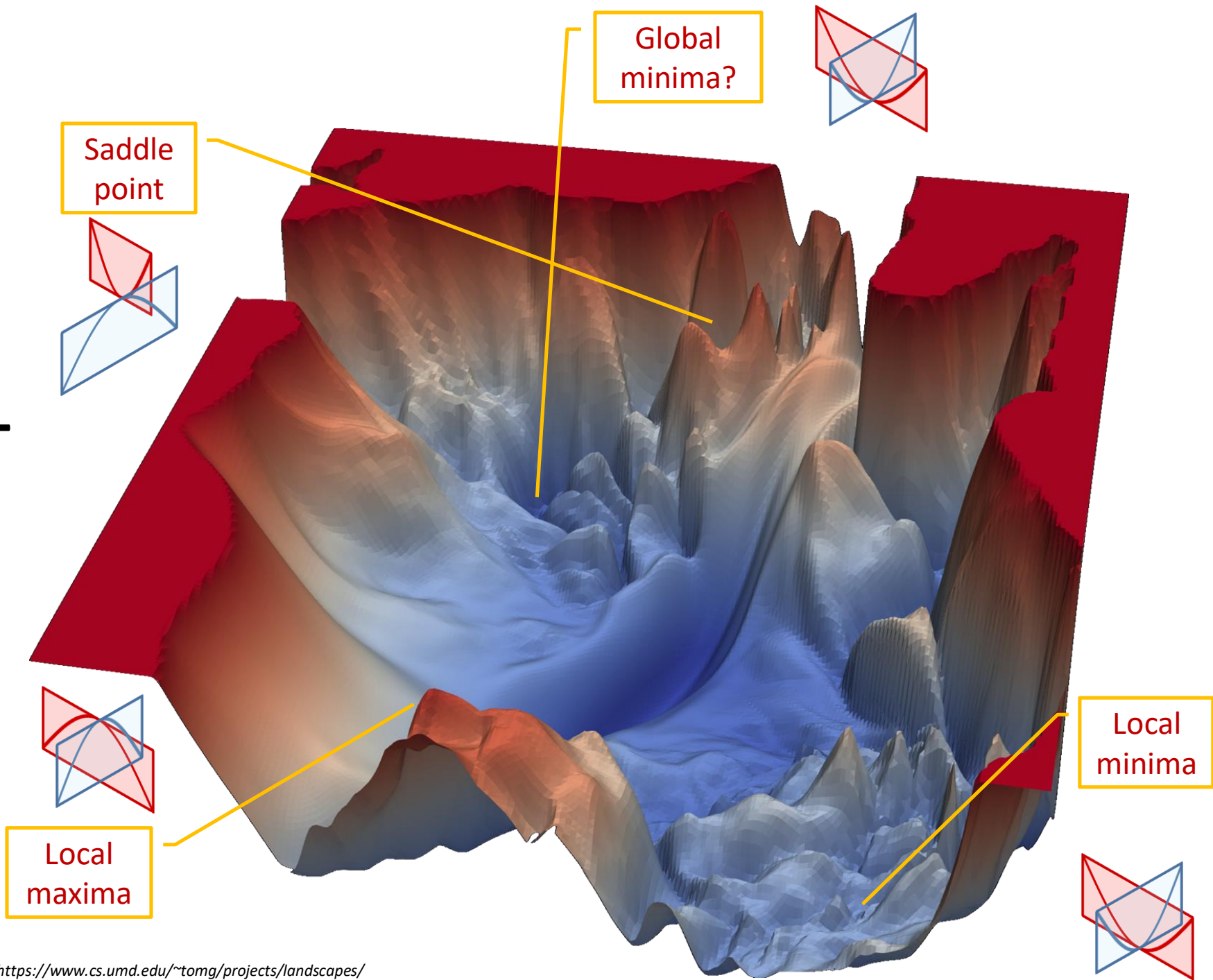$$-\frac{b}{2a} - x = -\frac{2ax+b}{2a} = -\frac{f'(x)}{f''(x)}$$

- So, the perfect learning rate is: $\eta^* = \dfrac{1}{f''(x)}$

- In multiple dimensions, $\boldsymbol{x} \leftarrow \boldsymbol{x} - H\big(f(\boldsymbol{x})\big)^{-1} \nabla\big(f(\boldsymbol{x})\big)$

- Practically, we do not want to compute the inverse of a Hessian matrix, so we approximate Hessian inverse

Saddle points, Hessian and long local furrows

- Some variables may have reached a local minima while others have not
- Some weights may have almost zero gradient
- At least some eigenvalues may not be negative

A realistic picture

Global minima?

Saddle point

Local maxima

Local minima

Image source: https://www.cs.umd.edu/~tomg/projects/landscapes/

Adam optimizer

- For update step t
  - $g_t \leftarrow \nabla_x f(x_{t-1})$
  - $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) g_t$
  - $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot (g_t \odot g_t)$
  - $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
  - $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$
  - $x_t \leftarrow x_{t-1} - \alpha \, \widehat{m}_t \,./\, (\sqrt{\widehat{v}_t} + \varepsilon)$
- Good default values:
  - $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$

- Explanation
  - Gradient
  - Momentum
  - Moment (grad. sq.)
  - Bias correction
  - Bias correction
  - Update
  - Why divide by moment?
    - It approximates Hessian inverse to give correct step size for each dimension