CE 395 Special Topics in Machine Learning

Assoc. Prof. Dr. Yuriy Mishchenko

Fall 2017

MATRICES AND VECTORS

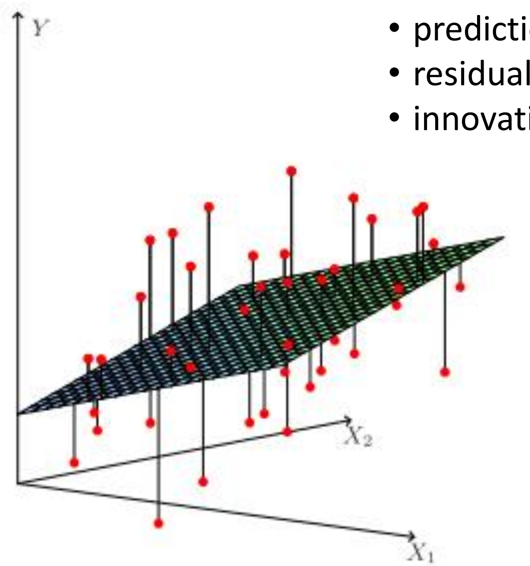
Linear regression in matrix notation

Linear regression problem: find the best linear model (geometrically – a plane) passing through data

$$y = x^T\theta + \varepsilon \Rightarrow \min \sum_{i=1}^{n} (y_i - x_i^T\theta)^2$$

- error
- prediction error
- residual
- innovation

Least Squares (LS) – minimize Residual-Sum-of-Squares

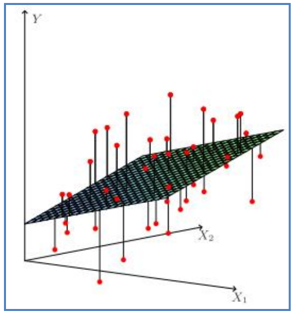


y←x relationship is linear

$y=\theta_0+\theta_1 x_1+\theta_2 x_2+\ldots+\theta_p x_p$

Note that offset θ_0 can be absorbed into the regression by introducing a new constant, "dummy" variable x_0=1

Linear regression in matrix notation



Matrix form of linear regression problem:

$$\min \sum_{i=1}^{n} (y_i - x_i{}^T \theta)^2 \Rightarrow \min |y - X\theta|^2$$

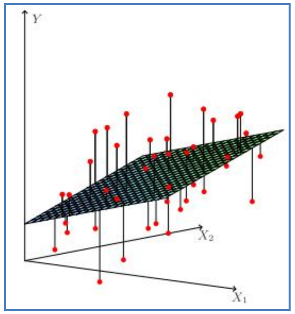
n – number of data points aka data examples or samples
p – number of variables aka features or predictors

Here X is the **feature matrix**:

$$X = [x_{np}] = \begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \dots \\ \vec{x}_n \end{bmatrix} = \begin{bmatrix} \vec{f}_1 & \vec{f}_2 & \dots & \vec{f}_p \end{bmatrix} =$$

$$= \begin{bmatrix} x_{1;1} & x_{1;2} & \dots & x_{1;p} \\ x_{2;1} & x_{2;2} & \dots & x_{2;p} \\ \dots & & & \\ x_{n;1} & x_{n;2} & \dots & x_{n;p} \end{bmatrix}$$

Linear regression in matrix notation



Matrix form of linear regression problem:

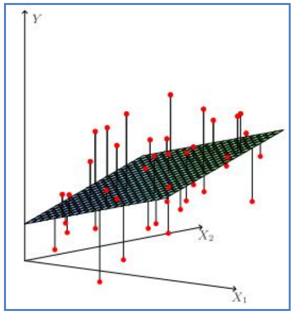
$$\min \sum_{i=1}^{n} (y_i - x_i^T \theta)^2 \Rightarrow \min |y - X\theta|^2$$

Here X is the **feature matrix**:

$$X = [x_{np}] = \begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \dots \\ \vec{x}_n \end{bmatrix} = \begin{bmatrix} \vec{f}_1 & \vec{f}_2 & \dots & \vec{f}_p \end{bmatrix} = \begin{bmatrix} x_{1;1} & x_{1;2} & \dots & x_{1;p} \\ x_{2;1} & x_{2;2} & \dots & x_{2;p} \\ \dots & & & \\ x_{n;1} & x_{n;2} & \dots & x_{n;p} \end{bmatrix}$$

Feature matrix can be seeing as n predictor vectors x_i for each of the n data examples stacked as rows x_i^T (the 1st form)

Linear regression in matrix notation



Matrix form of linear regression problem:

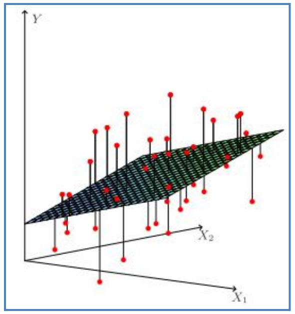
$$\min \sum_{i=1}^{n} (y_i - x_i^T \theta)^2 \Rightarrow \min |y - X\theta|^2$$

Here X is the **feature matrix**:

$$X = [x_{np}] = \begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \dots \\ \vec{x}_n \end{bmatrix} = \begin{bmatrix} \vec{f}_1 & \vec{f}_2 & \dots & \vec{f}_p \end{bmatrix} = \begin{bmatrix} x_{1;1} & x_{1;2} & \dots & x_{1;p} \\ x_{2;1} & x_{2;2} & \dots & x_{2;p} \\ \dots & & & \\ x_{n;1} & x_{n;2} & \dots & x_{n;p} \end{bmatrix}$$

Equally, feature matrix can be seeing as p column feature vectors, f_k , each containing the value of k^{th} feature for all n data examples, stacked as columns (the 2^{nd} form)

Linear regression in matrix notation



Matrix form of linear regression problem:

$$\min \sum_{i=1}^{n} (y_i - x_i^{\;T}\theta)^2 \Rightarrow \min \left| y - X\theta \right|^2$$

Here X is the **feature matrix**:

$$X = [x_{np}] = \begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \dots \\ \vec{x}_n \end{bmatrix} = \begin{bmatrix} x_{1;1} & x_{1;2} & \dots & x_{1;p} \\ x_{2;1} & x_{2;2} & \dots & x_{2;p} \\ \dots & & & \\ x_{n;1} & x_{n;2} & \dots & x_{n;p} \end{bmatrix}$$

The first view allows one to see that $X\theta$ gives the column vector of n predicted responses $x_i^T\theta$:

$$X\theta = \begin{bmatrix} x_{1;1} & x_{1;2} & \dots & x_{1;p} \\ x_{2;1} & x_{2;2} & \dots & x_{2;p} \\ \dots & & & \\ x_{n;1} & x_{n;2} & \dots & x_{n;p} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dots \\ \theta_p \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} = \hat{\vec{y}}$$

Linear regression in matrix notation

Matrix solution:

$$\min_{\theta} |y - X\theta|^2 = \min_{\theta}(y - X\theta)^T (y - X\theta) \Rightarrow$$

$$\partial_{\theta}(y - X\theta)^T (y - X\theta) = X^T (y - X\theta) \Rightarrow$$

$$X^T (y - X\theta) = 0 \Rightarrow (X^T X)\theta = X^T y$$

$$\Rightarrow \theta = (X^T X)^{-1} X^T y$$

$(X^T X)^{-1} X^T$ is called pseudo-inverse of matrix X;
for square matrix X pseudo-inverse is clearly just X^{-1}

Linear regression in matrix notation

Linear regression filter or hat matrix *H* –
predicted (expected) values of y^ can be linearly expressed from input data y. The matrix connecting the two is called **the hat matrix** and defines a filter on y.

$$\hat{y} = X\theta = \underbrace{X(X^T X)^{-1} X^T}_{H} y = Hy$$

H – the hat matrix

$$\theta = (X^T X)^{-1} X^T y$$

Linear regression in matrix notation

Another way to view linear regression is as a system of linear equations $y_i = x_i^T \theta$. Depending on the relationship between n and p, this may be either over-determined (n>p) or under-determined (n<p) (and in very rare cases n=p and one has a well formed system with single solution):

$$\begin{bmatrix} \quad X \quad \end{bmatrix}_{n \times p} \cdot [\theta] = [y]$$

if **p>n**, there is not enough data to uniquely identify all θ_p. Ridge regression typically is used in this case

$$\begin{bmatrix} \\ X \\ \\ \end{bmatrix}_{n \times p} \cdot [\theta] = \begin{bmatrix} \\ y \\ \\ \end{bmatrix}$$

if **n>p**, there is no solution that fits all data points perfectly. θ_p is determined in the “best-average” sense, as we discussed

Linear regression in matrix notation

Either for $n>p$ or $n<p$, the solution θ is always a linear combination of the data vectors x_i

$$\theta = \sum_i \alpha_i x_i$$

$$x_i = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_p \end{bmatrix}$$

This implies that θ can be obtained by knowing ***only*** the matrix of the dot products $G_{ij}=x_i^{\mathrm{T}}x_j$

This is called the ***kernel property*** and is important in ML with respect to very high-dimensional feature spaces

Vectors and vector spaces

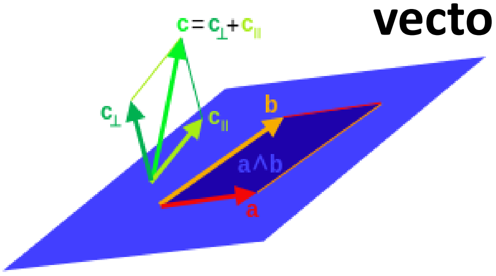
Our vectors were numerical arrays so far, but it is useful to also keep in mind their *linear algebra* side …

Vectors and vector spaces

Useful mental map:

$$[x_i]_{n\times 1} \longrightarrow \textbf{vector space} \longrightarrow [x_i]_{n\times 1}$$

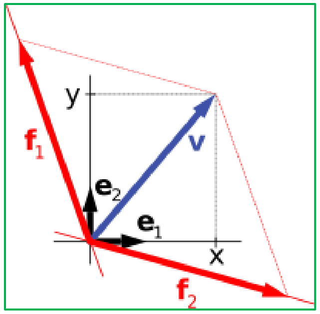
vector space



Vector space is the set of all v such that $v = a_1 v_1 + a_2 v_2 + \ldots + a_n v_n \quad \rightarrow$

Fact: vector space basis can be chosen arbitrarily

$$[x_i] \Rightarrow \sum_{i=1}^{n} \vec{e}_i x_i = \vec{x} = \sum_{i=1}^{n} \vec{f}_i x_i^{'} \Rightarrow [x_i^{'}]$$



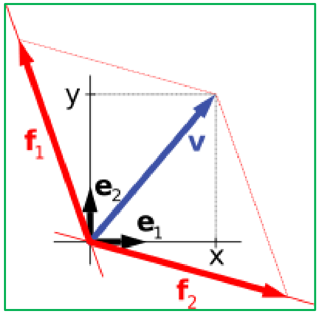
Vectors and vector spaces

Useful mental map:

$$[x_i]_{n\times 1} \longrightarrow \textbf{vector space} \longrightarrow [x_i]_{n\times 1}$$

Mutable Features: Numerical vectors (think – features in ML algorithms) are not immutable – they can be changed into different but equivalent forms (corresponding to the change of basis in corresponding vector space)

$$\vec{x} = \sum_{i=1}^{n} \vec{e}_i x_i = \sum_{i=1}^{n} \vec{f}_i x_i^{'}$$



Vectors and vector spaces

Example:

$[x_i]_{n\times 1}$

f:name	f:val (norm)
temperat.	0.78
humidity	0.75
pressure	1.05
solar act.	0.95
wind	2.0

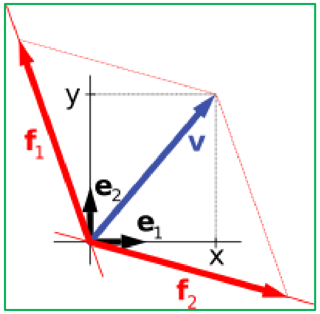
$\longrightarrow$

$[x_i^{'}]_{n\times 1}$

f:name	f:val
temp.+hum.	1.53
press.-wind	-0.95
temp.+solar.	1.73
hum.-press.	-0.30
wind	2.0

Vectors and vector spaces

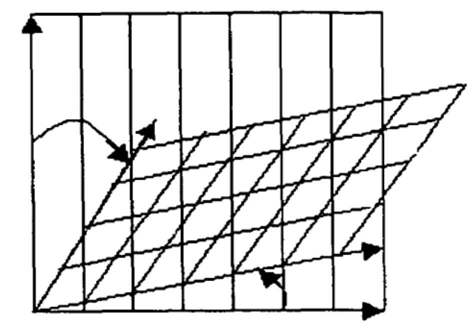
$$\vec{x} = \sum_{i=1}^{n} \vec{e}_i x_i = \sum_{i=1}^{n} \vec{f}_i x_i' = \vec{x}, \;\; x_i' = \sum_{j=1}^{n} A_{ij} x_j = Ax$$



Matrix Multiplication as Basis Change: Matrix multiplication can be viewed as a result of the change of basis in a vector space, which affects the **vectors' representation** in terms of its components x_i but leaves the **underlying vector object** without change

Vectors and vector spaces

$$\vec{x} \rightarrow \vec{x}' = \sum_{i=1}^{n} x_i' \vec{e}_i = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} A_{ij} x_j \right) \vec{e}_i$$



Matrix Multiplication as Transformation: Matrix multiplication also can be viewed as a linear transformation of the vectors themselves, which carries the vector object into a different orientation, without the change of basis

Vectors and vector spaces

$$y = \theta^T x \rightarrow\rightarrow y = \theta_0 x_0$$

For linear regression, there is always a representation of the original features in which the regression is simply a multiplication with single scalar θ_0.

HOW?

Vectors and vector spaces

$$y = \theta^T x \to\to y = \theta_0 x_0$$

Choose the first vector of a new basis to point in the direction of θ. In this basis, $\theta=[\theta_0,0,0,\ldots,0]$ and the dot product is trivial.

NUMERICAL OPTIMIZATION

Numerical optimization

Optimization is the problem of finding the best option among a set of possible solutions.

Numerical optimization

The essence of optimization problem is represented by the **objective function**, which evaluates the goodness of a possible solution

$$f$$

Numerical optimization

The solutions are represented by a sequence of numerical values called **parameters**

$$f(x_1, x_2, \ldots, x_p)$$

Numerical optimization

Optimization problem then is formulated mathematically as finding the choice of parameters within a certain allowed set that maximizes or minimizes the value of the objective function

$$\min f(x_1, x_2, ..., x_p)$$
$$(x_1, x_2, ..., x_p) \in \Omega$$

Note the relationship between min and max problems:

$$\min(f) = \max(-f)$$

Numerical optimization

Examples:

- Buy a house:
  - Parameters ?
  - Objective function ?
  - Solution domain ?
  - Solution ?
- Choose undergraduate degree:
  - Parameters ?
  - Objective function ?
  - Solution domain ?
  - Solution ?

Numerical optimization

Examples:

- Find path from server to client:
  - Parameters ?
  - Objective function ?
  - Solution domain ?
  - Solution ?
- Schedule software threads onto a CPU:
  - Parameters ?
  - Objective function ?
  - Solution domain ?
  - Solution ?

Numerical optimization

- Very typically we collect all the parameters into single numerical vector *x*:

$$(x_1, x_2, ..., x_p) \rightarrow x = [x_1, x_2, ..., x_p]^T$$
$$\min f(x_1, x_2, ..., x_p) \rightarrow \min_x f(x)$$

- Normally, *x* will be viewed as a <u>**column vector**</u>
- Most optimization problems require numerical solutions

Numerical optimization

The set of parameter vectors comprising valid possible solutions is called **feasibility domain** of optimization problem, and such solutions are called **feasible**

Feasibility domain is typically expressed via a series of equalities and inequalities, but we'll get to that later. For now we just treat it as some set Ω *in the parameter space* Φ*.*

We write

$$\min_{x} f(x) \quad \text{s.t.} \quad x \in \Omega$$

or

$$\min_{x \in \Omega} f(x)$$

("s.t." reads – subject to)

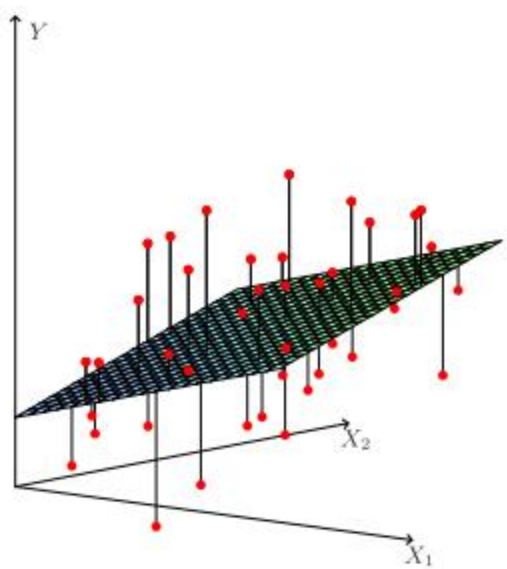
Why numerical optimization?

- All machine learning problems are (large scale) optimization problems:

$$\min_{\theta} L(\theta; \{x_i, y_i\})$$

- For example – linear regression:

$$\min \left\{ L(\theta) = \sum (y_i - x_i^{\,T} \theta)^2 \right\}$$



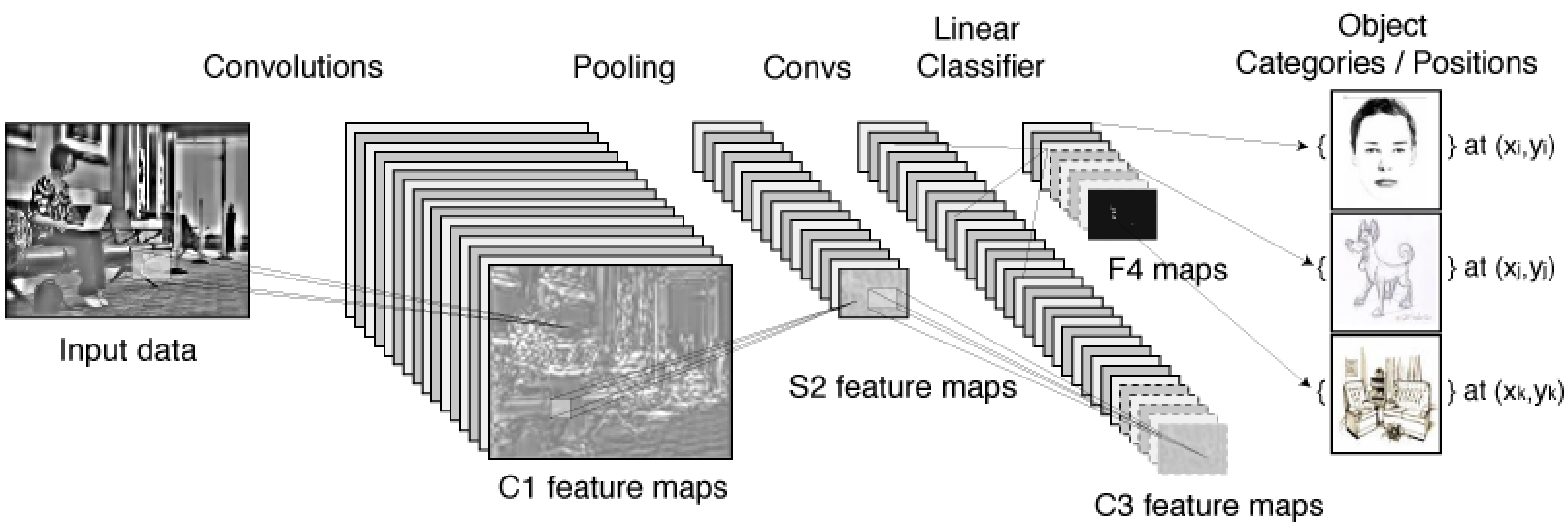
Why numerical optimization?

- Modern ML framework:
  - **Data** to be described $\{(x_i,y_i)\}$
  - A **model** for describing or fitting the data $\{y=f(x)\}$
  - A **loss function** for describing the quality of fit $L(\{(x_i,y_i)\}, f\,)$
- **ML problem:** find the model that minimizes the loss function:

$$\min_f L(\{x_i, y_i\}, f\,)$$

Why numerical optimization?

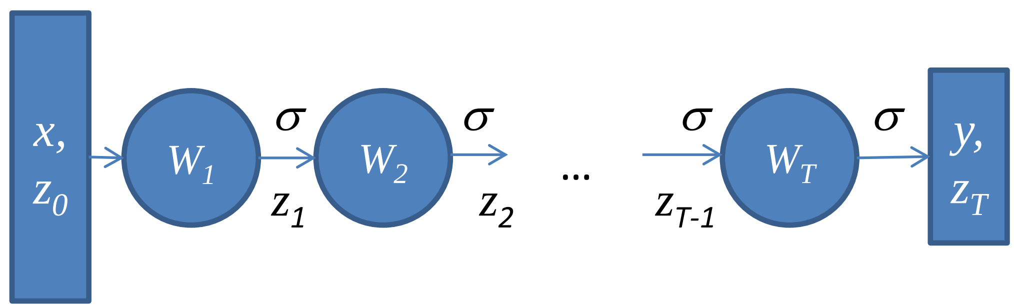
- Example – Deep Convolutional Neural Network:

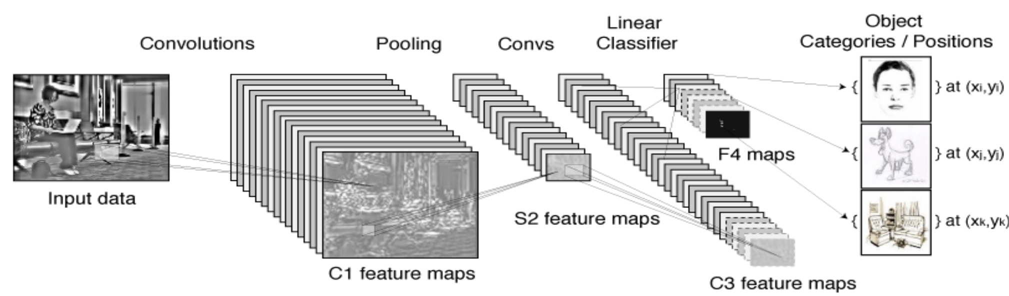


Why numerical optimization?

Artificial neural network (ANN) is a sequence of digital signal processing steps organized as follows

$$t = 1, \ldots, T$$
$$z_t = \sigma_t(W_t z_{t-1})$$
$$z_0 = x$$
$$z_T = y$$



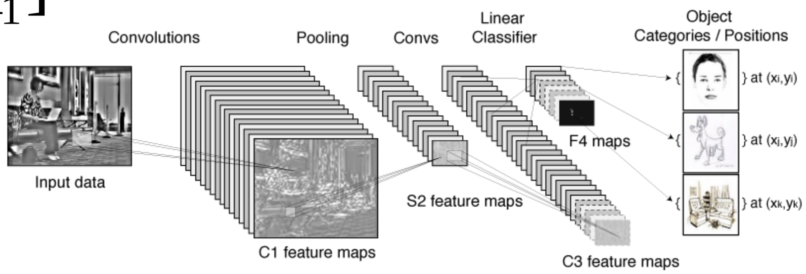


Why numerical optimization?

Convolutional neural network (CNN) is a neural network in which Wz are linear (image) filters, described by response functions h^k, applied to z and followed by one of {*sigmoid*, *ReLu*, *max* or *softmax*} transformations (σ)

$$W_t z_{t-1} = [h_t^1 * z_{t-1}, h_t^2 * z_{t-1}, \ldots, h_t^m * z_{t-1}]$$

$$\sigma_t \in \left\{ \frac{1}{1+e^{-z}}, \max(0, z), \ldots \right\}$$

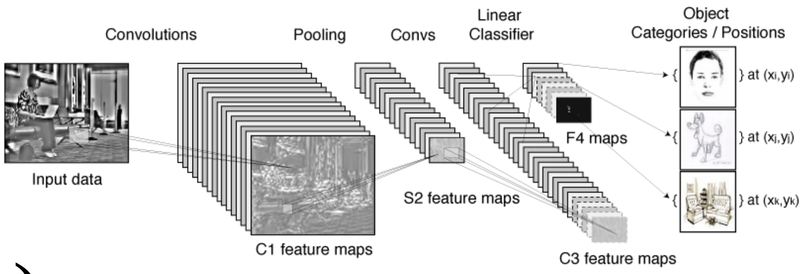


Why numerical optimization?

Deep CNN:

- Data: x is image represented as 2D numerical matrix, y is a list of objects represented as a vector of binary responses y_i={*object*$_i$=yes|no}
- Parameters: the sequence of $h^k{}_t$ (σ_t are typically fixed at design as the **DCNN architecture**)
- Loss function: typically **MSE – Mean Square Error**

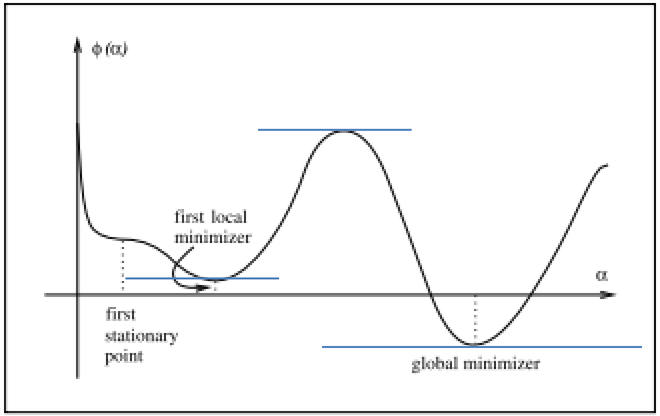
$$L(\{x_i, y_i\}, \{h_t^k\}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - dcnn(x_i; \{h_t^k\}))^2$$



- The problem:

$$\min L(\{x_i, y_i\}, \{h_t^k\})$$

Optimality conditions

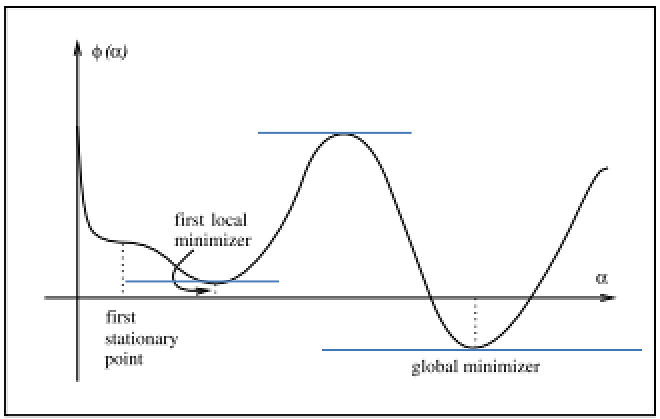


One dimensional case (the necessary condition for min or max):

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

Proof: [whiteboard]

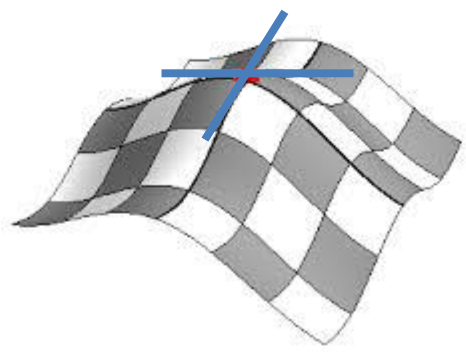
Optimality conditions



Example find minima of

$$f(x) = x^3 + 1.5x^2 - 6x + 8$$

Optimality conditions

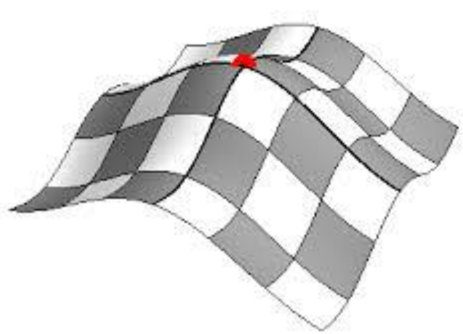


Many dimensional case (the necessary condition for min or max):

$$\frac{\partial f(x_1, \ldots, x_p)}{\partial x_i} = 0 \quad \text{for every } i$$

Proof: [whiteboard]

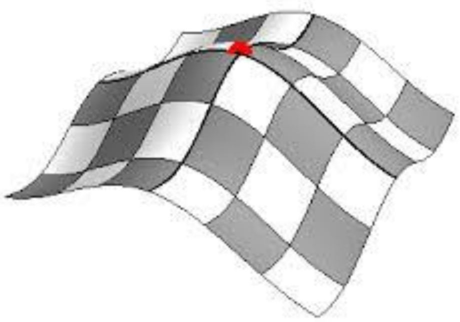
Optimality conditions



Find minimum of

$$f(x,y)=x^2+3xy+y^2-2x+y+1$$

Optimality conditions



General **quadratic form**:

$$\min\left\{ f(x) = \frac{1}{2} x^T Q x + L^T x + c \right\}$$

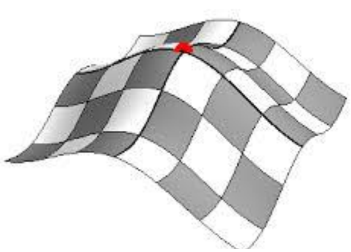
Solution is:

$$\boxed{x^* = -Q^{-1} L}$$

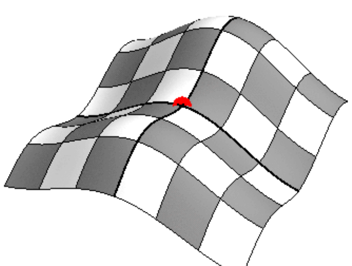
Optimality conditions

Types of **extreme points** and visual representation of many-D functions

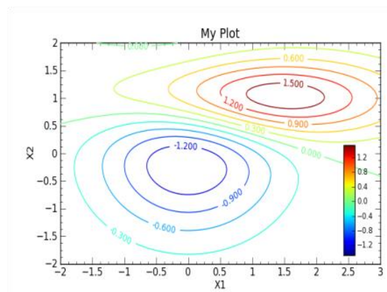
minimum

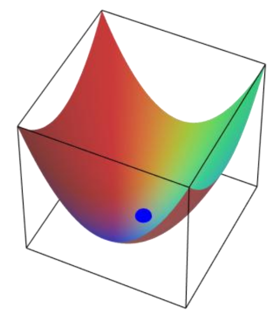


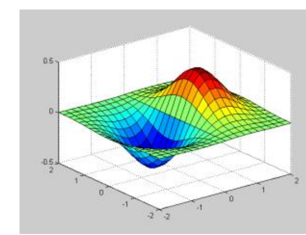
maximum

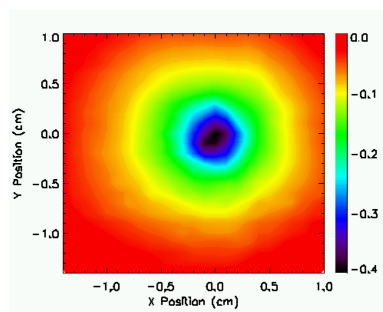


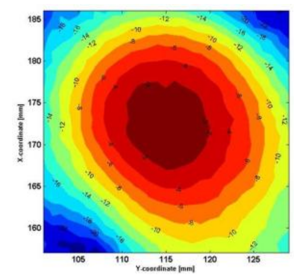
saddle point

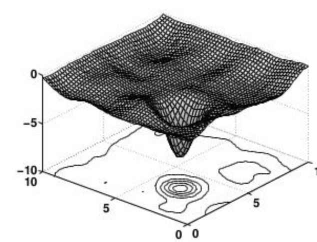


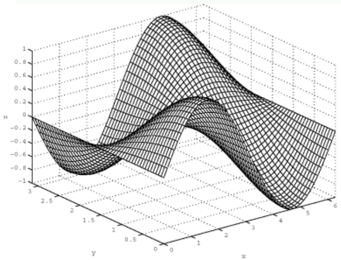


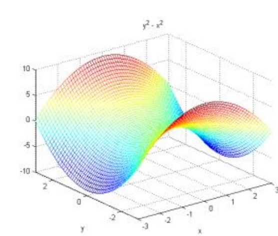


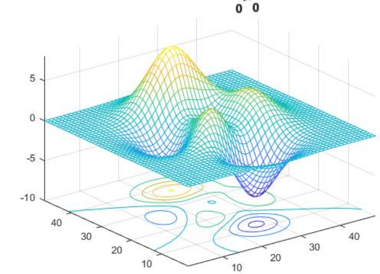






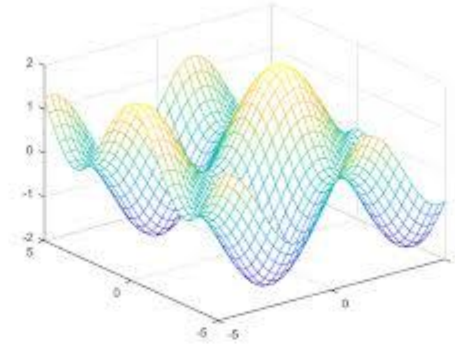


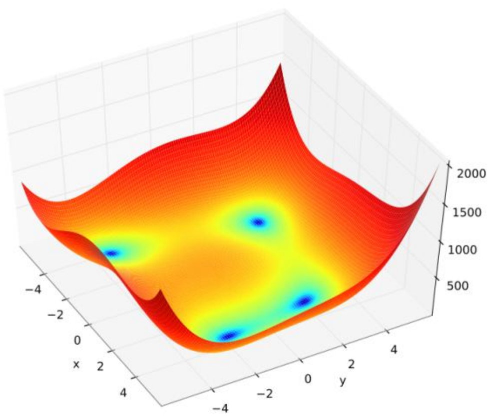


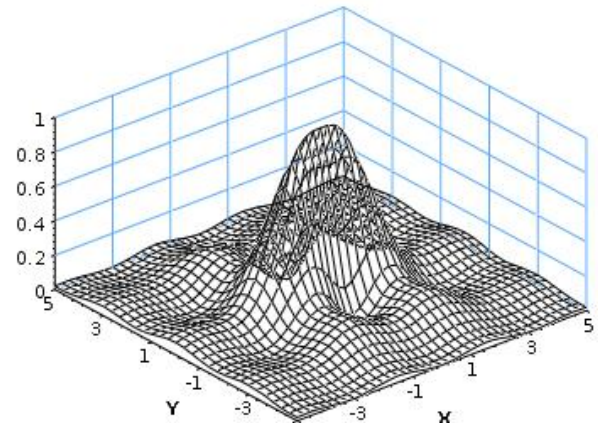


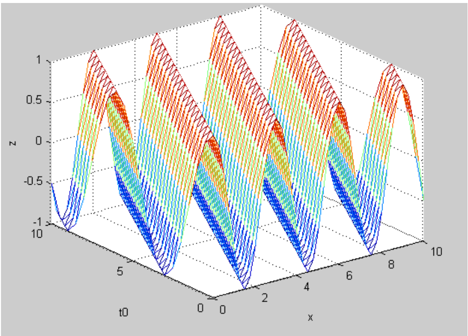
Optimality conditions

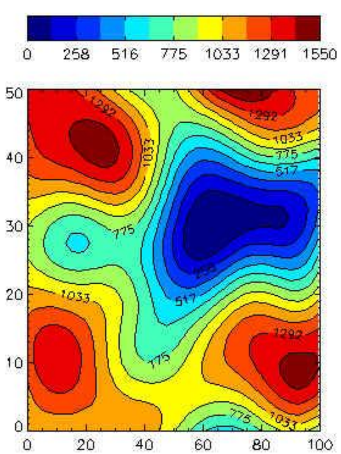
Challenge: describe these functions

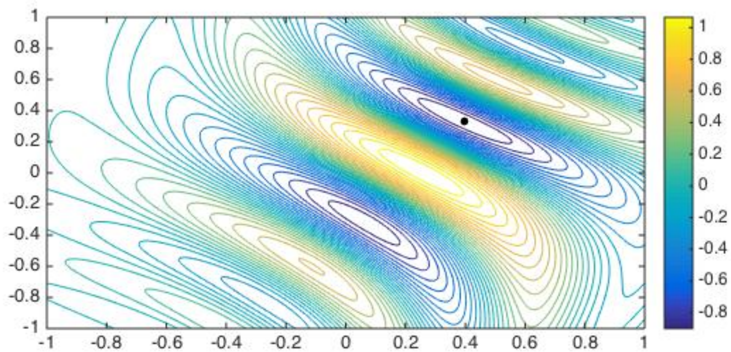


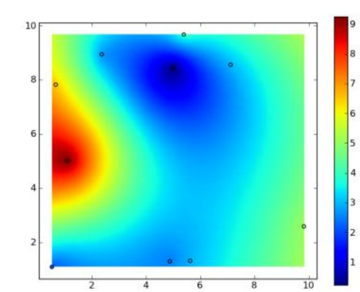






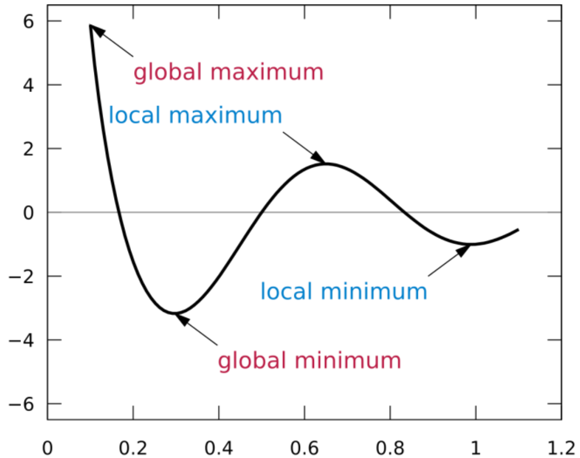




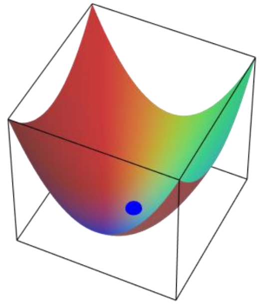


Optimality conditions

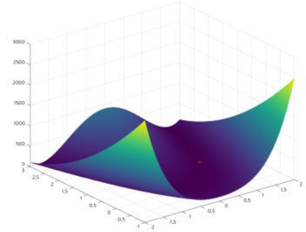
Local vs. **global optima**: difficulty of finding **global optimum**



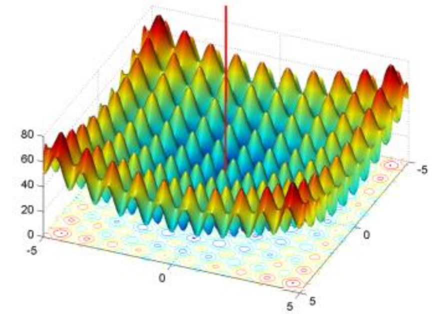
Easy – follow the increase locally



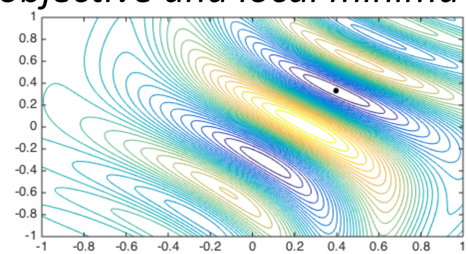
Hard – slow convergence because shallow

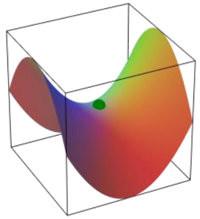


Hard – many local minima



Hard – highly non-uniform objective and local minima





NOT A MINIMUM

Gradient

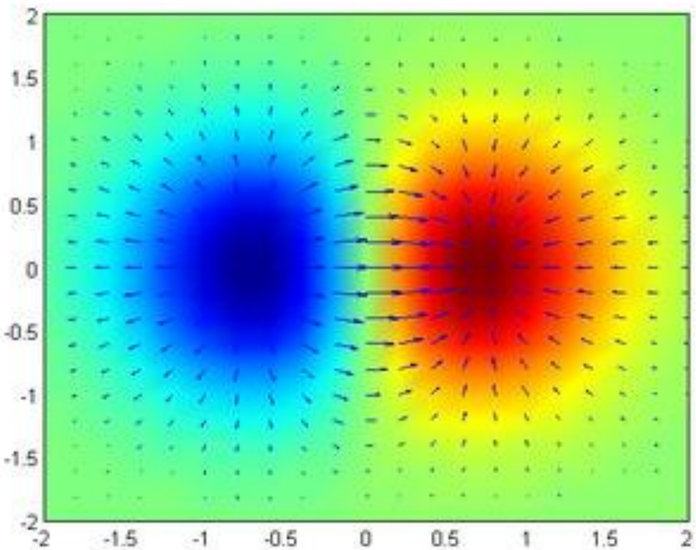
- Gradient descent is one of the simplest yet most powerful numerical optimization algorithms
- **Gradient** of a many-dimensional function *f*(*x*) is defined as the vector of its partial derivatives **(Definition)**

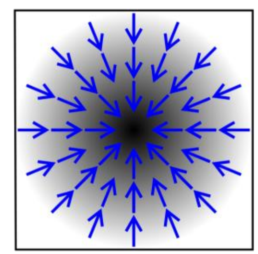
$$\nabla f = \left[\frac{\partial f(x)}{\partial x_i} \right] = \left[\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \ldots, \frac{\partial f(x)}{\partial x_p} \right]$$

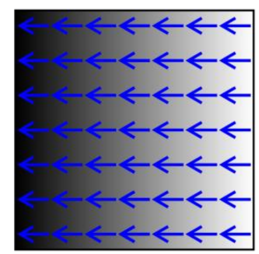
Gradient

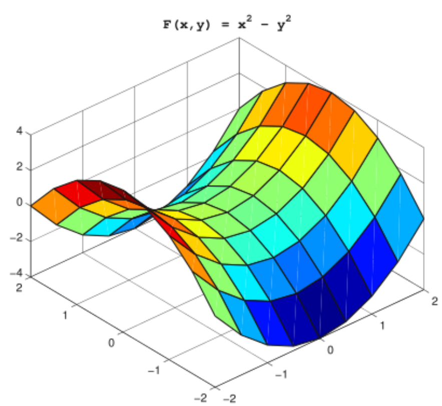
Direction of the gradient is towards the function's fastest increase

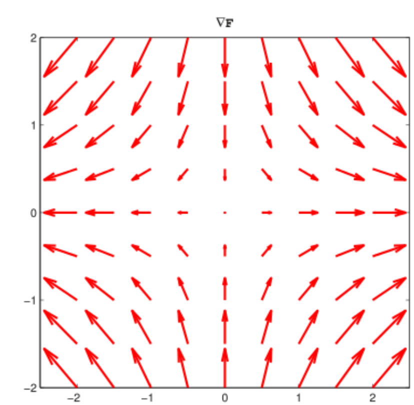
$$\nabla f = \left[\frac{\partial f(x)}{\partial x_i} \right]$$







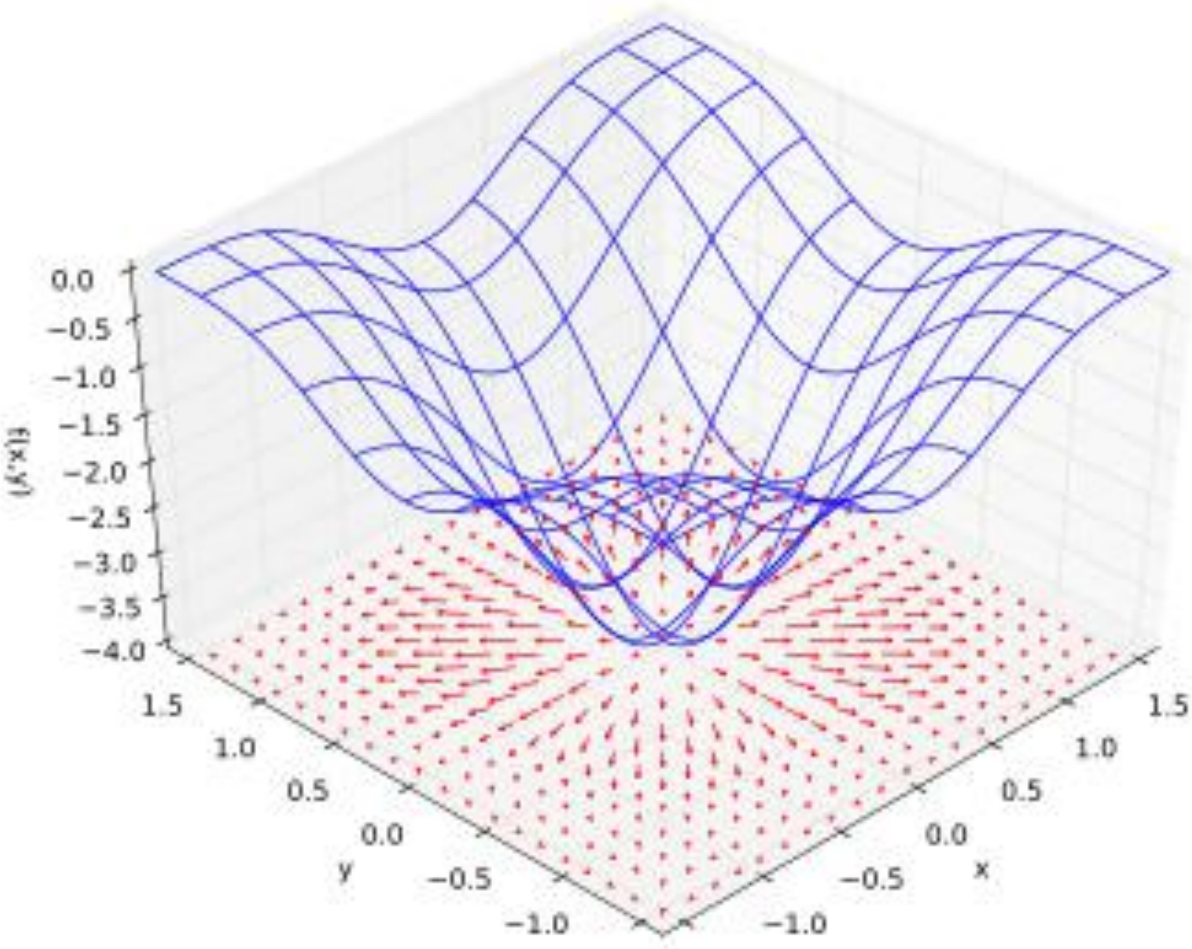




Gradient

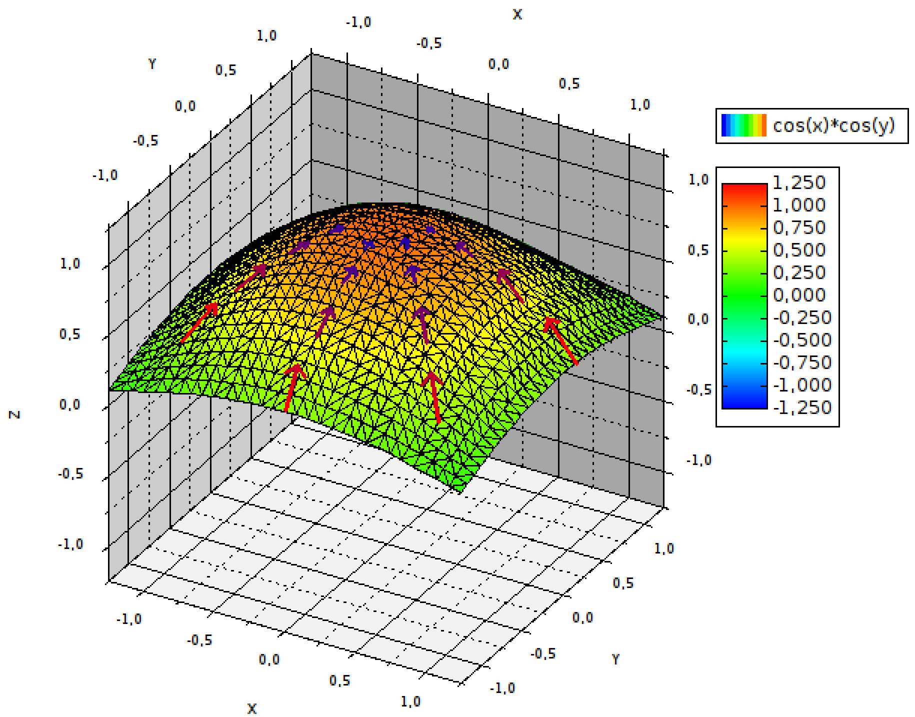
Challenge: read this plot

$$\nabla f = \left[\frac{\partial f(x)}{\partial x_i} \right]$$



Gradient

In order to increase the value of a function, simply need to follow its gradient up !



QUESTIONS FOR SELF-CONTROL

- Repeat and prove the correctness of all the steps of the derivation of the linear regression formula $\theta=(X^TX)^{-1}X^Ty$.
- Prove the correctness of the necessary optimality conditions on slide 34.
- Carry out the derivation of the minimum of a general quadratic form on slide 35.
- Carry out the challenge on slide 38.
- Explain why each of the examples of finding global minimum on slide 39 is easy or hard.
- Define partial derivative.
- Define gradient.
- Describe functions using their gradients on slide 41.

- Go back to slide 42 and make sure you understand the relationship between the function's surface and the gradients, plotted in xy plane, shown in the figure on that slide.
- Find the minimum of $f(x,y,z)=2x^2+y^2+4z^2-4xy+2yz+5x-3y+7$.
- Find the minimum of $f(x)=x^4-2x^2+7$.
- Write a small program that finds the minimum of arbitrary 1D function f(x) by finding the solution of $f'(x)=0$ by using any method.

- Calculate the gradient of $f(x,y,z)=2x^2+y^2+4z^2-4xy+2yz+5x-3y+7$ at every point.
- Calculate the gradient of $f(x,y)=x^4y^2-2x^2y^3+7y-3$ at every point.
- Calculate the gradient of $1-\exp(-x^2-2y^2)$ at every point. Where is the minimum of this function?
- Calculate the gradient of $\sin(x+2y^2)$ at every point. Where are the minima of this function?
- Calculate and express in matrix notation the gradient of general quadratic form x^TQx+g^Tx where x and g are px1 column vectors and Q is pxp symmetric square matrix.
- Calculate and express in matrix notation the gradient of the general Gaussian $\exp(-(x-g)^TQ(x-g))$, where x, g and Q are as in the previous question.

ADVANCED

The kernel property of linear regression

Take the solution that we obtained in the class:

$$\theta = (X^T X)^{-1} X^T y$$

Let's change the basis of x_i so that the new bases are vectors x_i themselves (and perhaps other stuff, if n<p); in this basis X is:

$$X = \begin{bmatrix} 1 & & & 0 & 0 \\ & 1 & & 0 & 0 \\ & & 1 & 0 & 0 \end{bmatrix}_{n \times p}$$

The kernel property of linear regression

Given that new basis view, we can find the solution for θ trivially now:

$$X = \begin{bmatrix} 1 & & & 0 & 0 \\ & 1 & & 0 & 0 \\ & & 1 & 0 & 0 \end{bmatrix}_{n \times p} = [I \quad 0] \; (block \; view)$$

$$X^T X = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} (block \; view)$$

Note that the inverse $(X^TX)^{-1}$ here strictly speaking doesn't exist. Think about the below formula and try to convince yourself that it makes sense in the sense of θ being a solution to the original residual minimization problem.

$$\hat{\theta} = (X^T X)^{-1} X^T y = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}^{-1} \cdot \begin{bmatrix} I \\ 0 \end{bmatrix}_{p \times n} \cdot [y]_{n \times 1} = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}_{px1}$$

The kernel property of linear regression

Note that the solution here θ^ is in the new basis X. To get θ^ in terms of the original features, we need to transform θ^ back, A^{-1}θ^, with whatever matrix A that was necessary to change the basis to the basis X, and the matrix X to the form [I 0].

$$\hat{\theta} = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}^{-1} \cdot \begin{bmatrix} I \\ 0 \end{bmatrix}_{p\times n} \cdot [y]_{n\times 1} = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}_{px1}$$