Elements of differential calculus and optimization.

Joan Alexis Glaunès

October 24, 2019

partial derivatives

Partial derivatives of a real-valued function defined on $\mathbf{R}^n$: $f : \mathbf{R}^n \to \mathbf{R}$.

- example : $f : \mathbf{R}^2 \to \mathbf{R}$,

$$f(x_1, x_2) = 2(x_1-1)^2+x_1x_2+x_2^2 \quad \Rightarrow \quad \begin{cases} \frac{\partial f}{\partial x_1}(x_1, x_2) = 4(x_1 - 1) + x_2 \\ \frac{\partial f}{\partial x_2}(x_1, x_2) = x_1 + 2x_2 \end{cases}$$

- example : $f : \mathbf{R}^n \to \mathbf{R}$,

$$f(x) = f(x_1, \ldots, x_n) = (x_2 - x_1)^2 + (x_3 - x_2)^2 + \cdots + (x_n - x_{n-1})^2$$

$$\Rightarrow \quad \begin{cases} \frac{\partial f}{\partial x_1}(x) = 2(x_1 - x_2) \\ \frac{\partial f}{\partial x_2}(x) = 2(x_2 - x_1) + 2(x_2 - x_3) \\ \frac{\partial f}{\partial x_3}(x) = 2(x_3 - x_2) + 2(x_3 - x_4) \\ \cdots \\ \frac{\partial f}{\partial x_{n-1}}(x) = 2(x_{n-1} - x_{n-2}) + 2(x_{n-1} - x_n) \\ \frac{\partial f}{\partial x_n}(x) = 2(x_n - x_{n-1}) \end{cases}$$

Directional derivatives

- Let $x, h \in \mathbf{R}^n$. We can look at the derivative of f at x in the direction h. It is defined as

$$f'_h(x) := \lim_{\varepsilon \to 0} \frac{f(x + \varepsilon h) - f(x)}{\varepsilon},$$

  i.e. $f_h(x) = g'(0)$ where $g(\varepsilon) = f(x + \varepsilon h)$ (the restriction of f along the line passing through x with direction h.

- The partial derivatives are in fact the directional derivatives in the directions of the canonical basis $e_i = (0, \ldots, 1, 0, \ldots, 0)$:

$$\frac{\partial f}{\partial x_i} = f'_{e_i}(x).$$

Differential form and Jacobian matrix

- The application that maps any direction h to $f'_h(x)$ is a linear map from $\mathbf{R}^n$ to $\mathbf{R}$. It is called the differential form of f at x, and denoted $f'(x)$ or $Df(x)$. Its matrix in the canonical basis is called the Jacobian matrix at x. It is a $1 \times n$ matrix whose coefficients are simply the partial derivatives :

$$Jf(x) = \left(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x) \right).$$

- Hence one gets the expression of the directional derivative in any direction $h = (h_1, \ldots, h_n)$ by multiplying this Jacobian matrix with the column vector of the h_i :

$$f'_h(x) = f'(x).h = Jf(x) \times h = \frac{\partial f}{\partial x_1}(x)h_1 + \cdots + \frac{\partial f}{\partial x_n}(x)h_n \quad (1)$$

$$= \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(x)h_i. \quad (2)$$

Differential form and Jacobian matrix

- More generally, if $f : \mathbf{R}^n \to \mathbf{R}^p$, $f = (f_1, \ldots, f_p)$ one defines the **differential** of f, $f'(x)$ or $Df(x)$ as the linear map from $\mathbf{R}^n$ to $\mathbf{R}^p$ whose matrix in the canonical basis is

$$Jf(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x) & \cdots & \frac{\partial f_1}{\partial x_n}(x) \\ \cdots & \cdots & \cdots \\ \frac{\partial f_p}{\partial x_1}(x) & \cdots & \frac{\partial f_p}{\partial x_n}(x) \end{pmatrix}$$

Differential form and Jacobian matrix

Some rule of differentiation

- linearity: if $f(x) = au(x) + bv(x)$, with u and v two functions and a, b two real numbers, then $f'(x).h = au'(x).h + bv'(x).h$.
- The chain rule: if $f : \mathbf{R}^n \to \mathbf{R}$ is a composition of two functions $v : \mathbf{R}^n \to \mathbf{R}^p$ and $u : \mathbf{R}^p \to \mathbf{R}$: $f(x) = u(v(x))$, then one has

$$f'(x).h = (u \circ v)'(x).h = u'(v(x)).v'(x).h$$

Gradient

- If $f : \mathbf{R}^n \to \mathbf{R}$, the matrix multiplication $Jf(x) \times h$ can be viewed also as a scalar product between the vector h and the vector of partial derivatives. We call this vector of partial derivatives the gradient of f at x, denoted $\nabla f(x)$.

$$f'(x).h = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(x) h_i = \langle \nabla f(x) , h \rangle .$$

- Hence we get three different equivalent ways for computing a derivative of a function : either as a directional derivative, or using the differential form notation, or using the partial derivatives.

Example

Example with $f(x) = \sum_{i=1}^{n-1}(x_{i+1} - x_i)^2$:

- Using directional derivatives : we write

$$g(\varepsilon) = f(x + \varepsilon h) = \sum_{i=1}^{n-1} (x_{i+1} - x_i + \varepsilon(h_{i+1} - h_i))^2$$

$$g'(\varepsilon) = 2\sum_{i=1}^{n-1} (x_{i+1} - x_i + \varepsilon(h_{i+1} - h_i))\,(h_{i+1} - h_i)$$

$$f'(x).h = g'(0) = 2\sum_{i=1}^{n-1} (x_{i+1} - x_i)\,(h_{i+1} - h_i)$$

Example

- Using differential forms : we write

$$f(x) = \sum_{i=1}^{n-1} (x_{i+1} - x_i)^2$$

$$f'(x) = 2 \sum_{i=1}^{n-1} (x_{i+1} - x_i)(dx_{i+1} - dx_i)$$

where dx_i denotes the differential form of the coordinate function $x \mapsto x_i$ which is simply $dx_i.h = h_i$.

- Applying this differential form to a vector h we retrieve

$$f'(x).h = 2 \sum_{i=1}^{n-1} (x_{i+1} - x_i)(h_{i+1} - h_i)$$

Example

- Using partial derivatives : we write

$$\begin{aligned} f'(x).h = f'_h(x) &= \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(x) h_i \\ &= 2(x_1 - x_2)h_1 \\ &\quad + (2(x_2 - x_1) + 2(x_2 - x_3))\, h_2 \\ &\quad + \ldots + 2(x_n - x_{n-1})h_n \end{aligned}$$

Arranging terms differently we get finally the same formula:

$$f'(x).h = 2 \sum_{i=1}^{n-1} (x_{i+1} - x_i)\,(h_{i+1} - h_i)$$

- This calculus is less straightforward because we first identified terms corresponding to each h_i to compute the partial derivatives, and then grouped terms back to the original summation.

Example

Corresponding Matlab codes : these two codes compute the gradient of f (they give exactly the same result) :

- Code that follows the partial derivatives calculus : we compute the partial derivative $\frac{\partial f}{\partial x_i}(x)$ for each i and put it in the coefficient i of the gradient.

```
function G = gradient_f(x)
  n = length(x);
  G = zeros(n,1);
  G(1) = 2*(x(1)-x(2));
  for i=2:n-1
    G(i) = 2*(x(i)-x(i-1)) + 2*(x(i)-x(i+1));
  end
  G(n) = 2*(x(n)-x(n-1));
end
```

Example

- Code that follows the differential form calculus : we compute coefficients appearing in the summation and incrementally fill the corresponding coefficients of the gradient

```
function G = gradient_f(x)
  n = length(x);
  G = zeros(n,1);
  for i=1:n-1
    c = 2*(x(i+1)-x(i));
    G(i+1) = G(i+1) + c;
    G(i) = G(i) - c;
  end
end
```

- This second code is better because it only requires the differential form, and also because it is faster : at each step in the loop, only one coefficient $2(x_{i+1} - x_i)$ is computed instead of two.

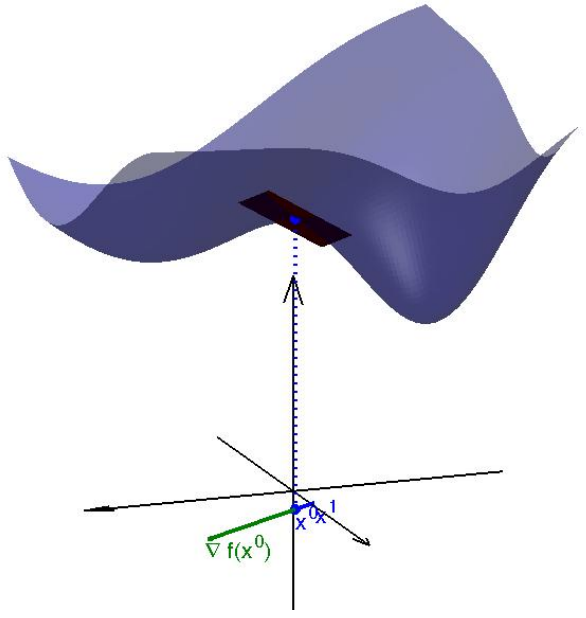
Gradient descent algorithm

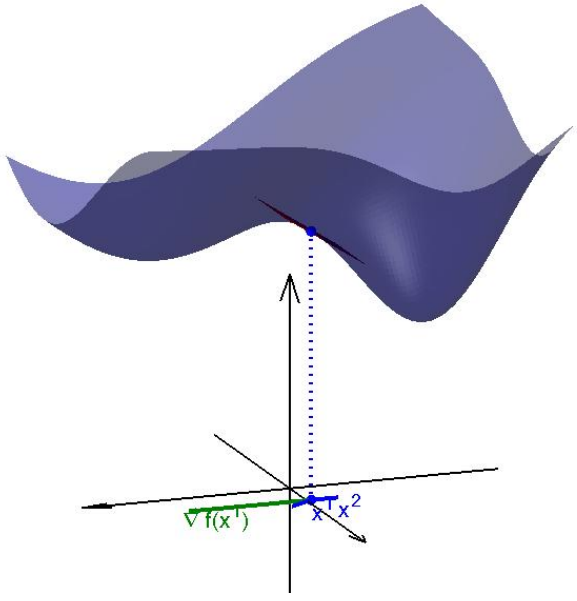
- Let $f : \mathbf{R}^n \to \mathbf{R}$ be a function. The gradient of f gives the direction in which the function increases the most. Conversely the opposite of the gradient gives the direction in which the function decreases the most.
- Hence the idea of gradient descent is to start from a given vector $x^0 = (x_1^0, x_2^0, \ldots, x_n^0)$, move from x^0 with a small step in the direction $-\nabla f(x_0)$, recompute the gradient at the new position x^1 and move again in the $-\nabla f(x^1)$ direction, and repeat this process a large number of times to finally get to the position for which f has a minimal value.
- Gradient descent algorithm : choose initial position $x^0 \in \mathbf{R}^n$ and stepsize $\eta > 0$, and compute iteratively the sequence

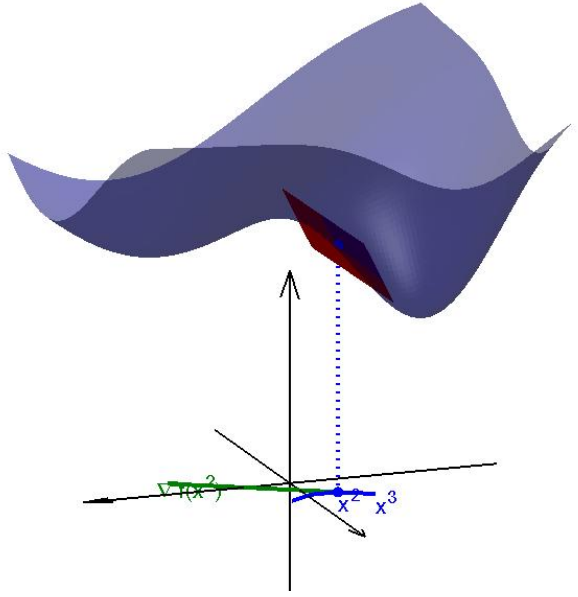
$$x^{k+1} = x^k - \eta \nabla f(x^k).$$

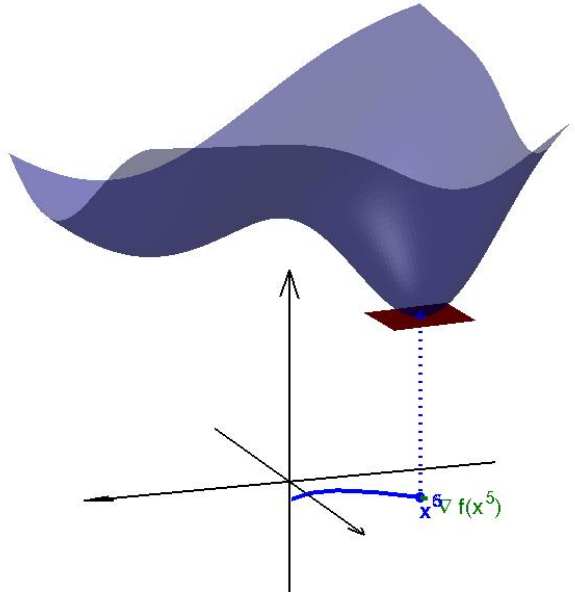
- The convergence of the sequence to a minimizer of the function depends on properties of the function and the choice of η (see later).

Gradient descent algorithm









First order Taylor expansion of a function

- Let $f : \mathbf{R}^n \to \mathbf{R}$. The first-order Taylor expansion at point $x \in \mathbf{R}^d$ writes

$$f(x+h) = f(x) + \langle h \, , \nabla f(x) \rangle + o(\|h\|),$$

or equivalently

$$f(x+h) = f(x) + \sum_{i=1}^{n} h_i \frac{\partial f}{\partial x_i}(x) + o(\|h\|).$$

- This means f is approximated by a linear map locally around point x.

Hessian and second-order Taylor expansion

- The Hessian matrix of a function f is the matrix of second-order partial derivatives :

$$Hf(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) & \cdots & \frac{\partial^2 f}{\partial x_n^2}(x) \end{pmatrix}$$

- The second-order Taylor expansion writes

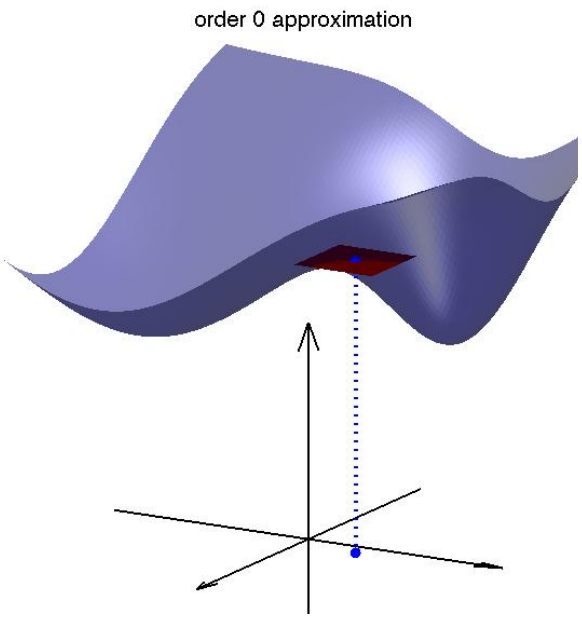
$$f(x+h) = f(x) + \langle h\,, \nabla f(x) \rangle + \frac{1}{2} h^T Hf(x) h + o(\|h\|^2),$$

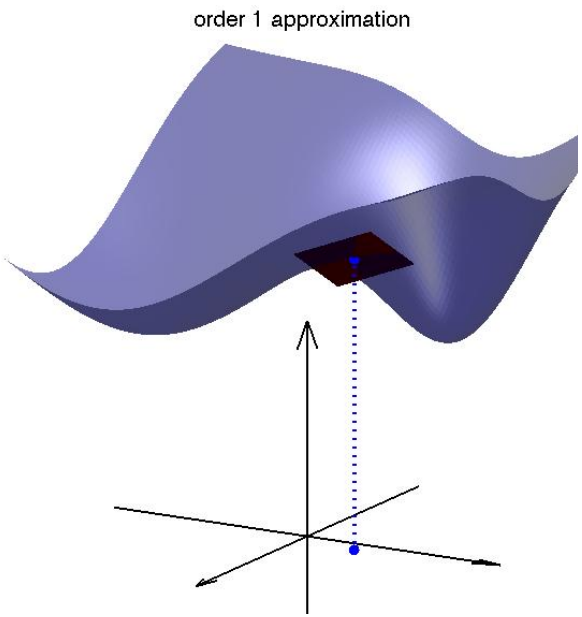
where h is taken as a column vector and h^T is its transpose (row vector).

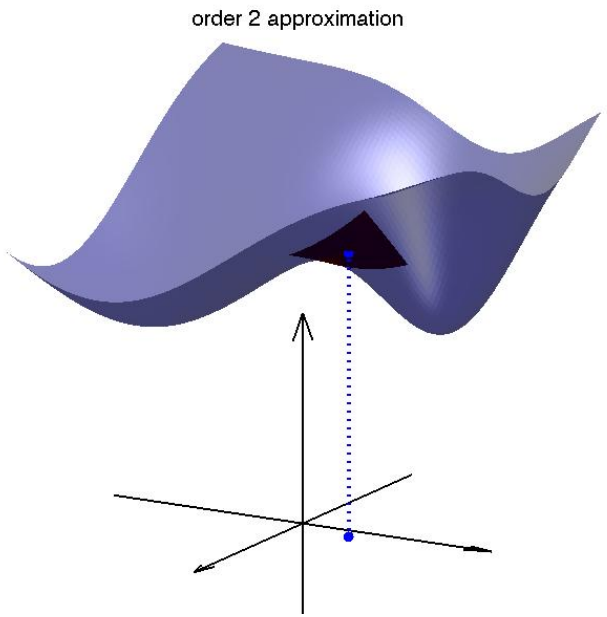
- Developing this formula gives

$$f(x+h) = f(x) + \sum_{i=1}^{n} h_i \frac{\partial f}{\partial x_i}(x) + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} h_i h_j \frac{\partial^2 f}{\partial x_i \partial x_j}(x) + o(\|h\|^2).$$

Taylor expansion







1st order optimality condition

- If x is a local minimizer of f, i.e. $f(x) \leq f(y)$ for any y in a small neighbourhood of x, then

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

- A point x that satisfies $\nabla f(x) = 0$ is called a critical point. So every local minimizer is a critical point, but the converse is false.
- In fact we distinguish three types of critical points: local minimizers, local maximizers, and saddle points (saddle points are just critical points that are neither local minimizers or maximizers).
- Generally the analysis of the hessian matrix allows to distinguish between these three types (see next slide)

2nd order optimality condition

- The Hessian matrix $Hf(x)$ is symmetric ; hence it has n real eigenvalues.
- A symmetric matrix M whose eigenvalues are all positive is called positive definite matrix. It is characterized by the fact that $v^T Mv > 0$ for every vector $v \neq 0$.
- If x is a critical point (i.e. $\nabla f(x) = 0$), then the Taylor expansion writes

$$f(x+h) = f(x) + \frac{1}{2} h^T Hf(x) h + o(\|h\|^2).$$

- So if all eigenvalues of $Hf(x)$ are positive then $f(x+h) > f(x)$ for h small enough. This means x is a local minimizer.
- Conversely if all eigenvalues of $Hf(x)$ are negative then x is a local maximizer.
- If at least one eigenvalue is positive and another is negative, then x is a saddle point.
- In other cases we cannot determine the type of critical point by the analysis of the hessian matrix.

Convex sets and convex functions

- A set $C \subset \mathbf{R}^n$ is convex if for any two points $x, y \in C$, the segment joining x and y is included in C. Equivalently this writes

$$\forall x, y \in C,\ \forall \lambda \in [0,1],\ \lambda x + (1-\lambda)y \in C.$$

- If $C \subset \mathbf{R}^n$ is convex, we say that a function $f : C \to \mathbf{R}$ is convex if

$$\forall x, y \in C,\ \forall \lambda \in [0,1],\ f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y).$$

- A function $f : C \to \mathbf{R}$ is strictly convex if

$$\forall x, y \in C, x \neq y\ \forall \lambda \in (0,1),\ f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda) f(y).$$

Convex sets and convex functions

- Characterizations of convex and strictly convex functions :

$$\forall x, y,\ \langle \nabla f(x) - \nabla f(y)\,, x - y \rangle \geq 0 \quad \Leftrightarrow \quad f \text{ is convex,}$$

$$\forall x \neq y,\ \langle \nabla f(x) - \nabla f(y)\,, x - y \rangle > 0 \quad \Leftrightarrow \quad f \text{ is strictly convex.}$$

Also :

$$\forall x,\ Hf(x) \text{ has nonegative eigenvalues} \quad \Leftrightarrow \quad f \text{ is convex,}$$

$$\forall x,\ Hf(x) \text{ has positive eigenvalues} \quad \Rightarrow \quad f \text{ is strictly convex.}$$

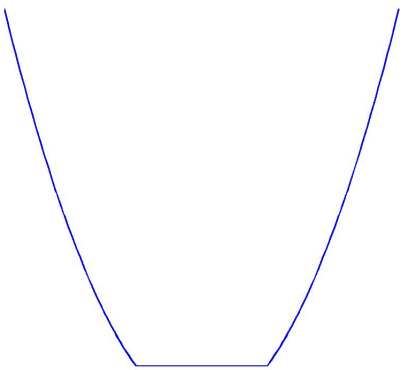
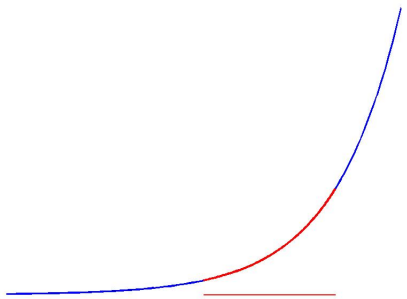
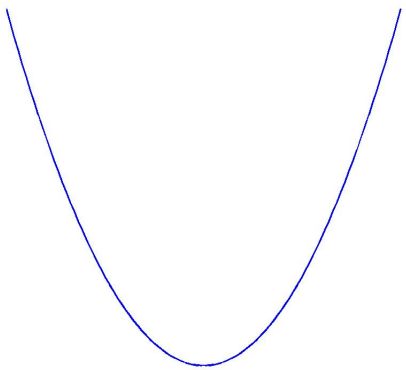
- **Elliptic functions** : f is elliptic if there exists $\alpha > 0$ such that all eigenvalues of $Hf(x)$ are greater than or equal to α for all x. This means $Hf(x)$ has positive eigenvalues everywhere (so it is strictly convex) and that these eigenvalues cannot get arbitrary small values when varying x.

Existence and uniqueness results for minimizers

- If $f : C \to \mathbf{R}$ is convex, then every critical point is a minimizer of f :

$$\nabla f(x) = 0 \quad \Rightarrow \quad \forall y \in \mathbf{R}^d, f(x) \leq f(y).$$

- If f is strictly convex, then f have at most one minimizer.
- If f is strictly convex and C is closed, non empty, convex and bounded, then f has a unique minimizer.
- If f is elliptic with C a closed non empty convex set, then f has a unique minimizer.

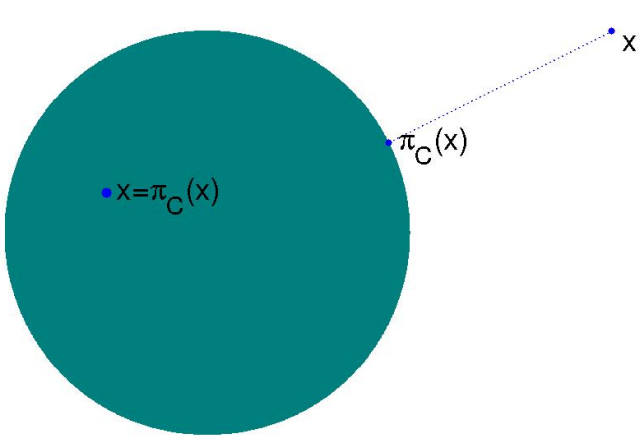
Projection on a convex set

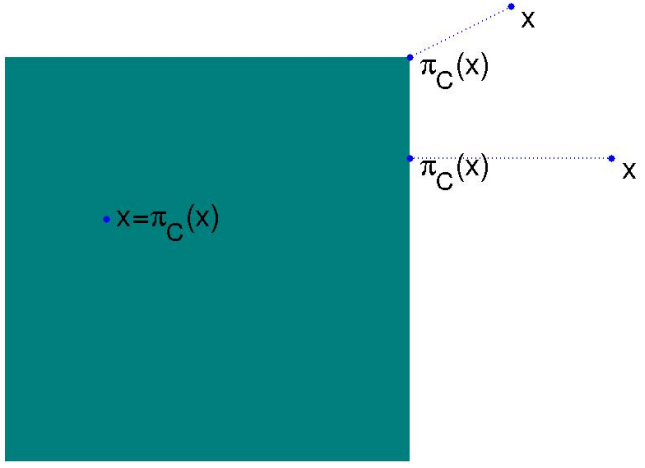
- If $C \subset \mathbf{R}^n$ is a closed, convex and non-empty set, then ones can define the projection of any $x \in \mathbf{R}^n$ onto the set C : it is the unique point $\bar{x} \in C$ which is the closest to x among all points in C :

$$\forall y \in C, \|x - \bar{x}\| \leq \|x - y\|$$

- It is also characterized as the unique point $\bar{x} \in C$ such that

$$\forall y \in C, \langle x - \bar{x}\, , y - \bar{x} \rangle \leq 0.$$





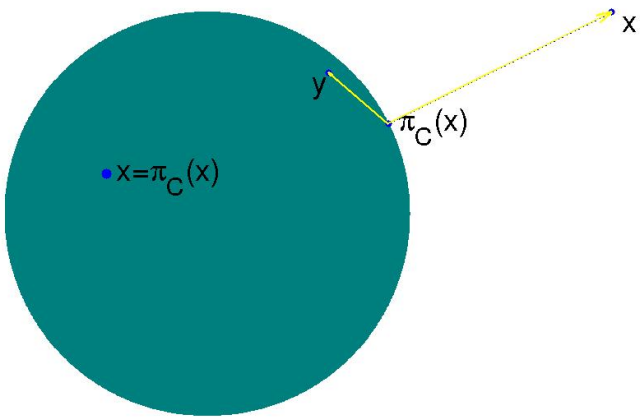
Projection on a convex set

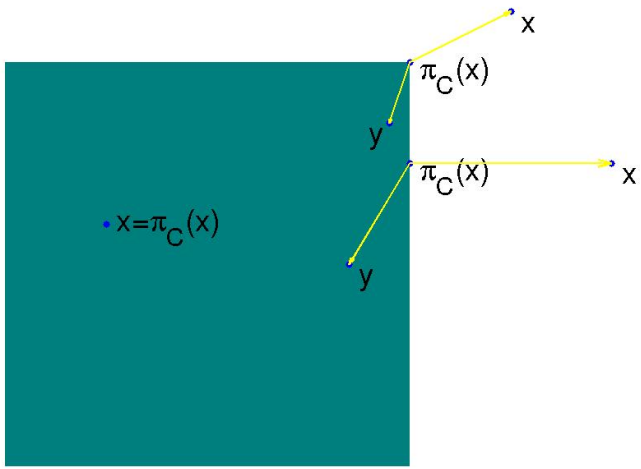
- If $C \subset \mathbf{R}^n$ is a closed, convex and non-empty set, then ones can define the projection of any $x \in \mathbf{R}^n$ onto the set C : it is the unique point $\bar{x} \in C$ which is the closest to x among all points in C :

$$\forall y \in C, \|x - \bar{x}\| \leq \|x - y\|$$

- It is also characterized as the unique point $\bar{x} \in C$ such that

$$\forall y \in C, \langle x - \bar{x}\, , y - \bar{x} \rangle \leq 0.$$





Projected gradient descent

- Projected gradient descent can be used to solve constrained optimization problems:

$$\begin{cases} \text{Find the minimizer of } J(x), \\ \text{with } x \in C \end{cases}$$

where $J : \mathbf{R}^n \to \mathbf{R}$ and $C \subset \mathbf{R}^n$ is a closed convex non-empty set. The algorithm is the following :

  - Choose initial $x^0 \in \mathbf{R}^n$, stepsize λ and number of iterations N.
  - For $k = 1$ to N compute

$$x^k = \pi_C(x^{k-1} - \lambda \nabla J(x^{k-1})).$$

- This is specially useful when the projection π_C can be computed easily (via a simple formula).
- The convergence of the projected gradient descent is ensured for a small stepsize η when J has some nice properties. In particular it is true when $Hf(x)$ has bounded positive eigenvalues.

Projected gradient descent

- Example :

$$\begin{cases} \text{Minimize } J(x_1, x_2) = 3x_1^2 + x_2^2 - x_1x_2 + x_2 \\ \text{with the constraint } x_1^2 + x_2^2 \leq 1. \end{cases}$$

- Here the set C of constraints is the unit disc. The projection on the disc is straightforward :

$$\pi_C(x) = \begin{cases} x \text{ if } x \in C \\ \frac{x}{\|x\|} \text{ otherwise} \end{cases} = \frac{x}{\max(1, \|x\|)}.$$

- The gradient of J writes

$$\nabla J(x) = \begin{pmatrix} 6x_1 - x_2 \\ 2x_2 - x_1 + 1 \end{pmatrix}$$

Projected gradient descent

- Corresponding Matlab code of the projected gradient descent :

```
function x = ProjectedGradient(x,eta,N)
  for k=1:N
    G = [6*x(1)-x(2);2*x(2)-x(1)+1];
    x = x - eta * G;
    x = x / max(1,norm(x));
  end
end
```

Projected gradient descent

- Another example : let A be an $n \times n$ matrix and b a vector of length n.

$$\begin{cases} \text{Minimize } J(x) = \|Ax - b\|^2 \\ \text{with the constraints } x_i \geq 0 \quad \forall i. \end{cases}$$

- The set C here is the set of vectors x with non-negative coefficients. It is easy to show that it is convex, closed and non-empty. The projection on this set is

$$\pi_C(x) = \bar{x} \qquad \text{with } \bar{x}_i = \max(0, x_i)$$

- The gradient of J is

$$DJ(x).h = 2\langle Ax - b\,,\, Ah\rangle = 2\left\langle A^T(Ax - b)\,,\, h\right\rangle,$$

and thus $\nabla J(x) = 2A^T(Ax - b)$.

Projected gradient descent

- Corresponding Matlab code for the projected gradient descent :

```
function x = ProjectedGradient(x,A,b,eta,N)
  for k=1:N
    G = 2*A'*(A*x-b);
    x = x - eta * G;
    x = max(0,x);
  end
end
```