Lecture 5-6

CM50264: Machine Learning 1

Optimization Basics

Kwang In Kim

Previously in decision trees ...

We are given a dataset

$$D = \{(\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^N, y^N)\} \subset \mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^n \times \mathbb{R}.$$

Training a decision tree:

- Grow a tree from a rootnote.
- At each node, *split* the data into *children* nodes.
- Splits are chosen using a splitting criterion.

At each splitting stage, a single feature x_* is selected from n-possible features $\{x_1, \ldots, x_n\}$ ($\mathbf{x} = [x_1, \ldots, x_n]^\top$) by

- evaluating the information gain $IG(x_i)$ for each feature x_i.
- selecting the index i that achieves the highest information gain IG.

Previously in decision trees ...

A single feature x_* is selected from n-possible features $\{x_i\}_{i=1}^n$ ($\mathbf{x} = [x_1, \ldots, x_n]^\top$) by

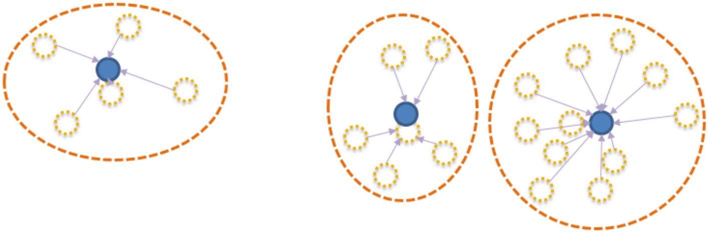
- evaluating the information gain $IG(x_i)$ for each feature x_i.
- selecting the index i that achieves the highest information gain IG.

A bit more formally ...

- We are given a candidate set $\mathcal{X} = \{x_1, \ldots, x_n\}$ and a function $IG : \mathcal{X} \to \mathbb{R}$.
- The optimal feature x_* is found by maximizing IG:

$$x_* = \arg\max_{x \in \mathcal{X}} IG(x).$$

Previously in clustering...



Given a dataset $D = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\} \subset \mathbb{R}^n$, our goal is to find cluster centers $\{\mathbf{c}^1, \mathbf{c}^2, \mathbf{c}^3\} \subset \mathbb{R}^N$ that minimizes the average quantization (or reconstruction) error:

$$\mathcal{O}(\{\mathbf{c}^1, \mathbf{c}^2, \mathbf{c}^3\}) = \sum_{i=1}^{N} \|\mathbf{x}^i - \mathbf{c}(\mathbf{x}^i)\|^2.$$

Regression problem ...

We are given a training dataset

$$D = \{(\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^N, y^N)\} \subset \mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^n \times \mathbb{R}.$$

Our goal is to find a function

$$f : \mathbb{R}^n \to \mathbb{R}$$

such that its output $f(\mathbf{x}^*)$ for an unseen input $\mathbf{x}^* \notin D$ is close to the underlying ground-truth output y^*:

We want to find a function f^* that minimizes

$$\int l(f(\mathbf{x}), y) dP(\mathbf{x}, y),$$

for a loss function $l(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, e.g.

$$l(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2.$$

Regression problem ...

$$D = \{(\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^N, y^N)\} \subset \mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^n \times \mathbb{R}.$$

We wish to minimize

$$\mathcal{O}(f) = \int (f(\mathbf{x}) - y)^2 dP(\mathbf{x}, y).$$

However, in practice we do not have access to the underlying data generation process $P(\mathbf{x}, y)$.

Instead, we minimize the empirical mean squared error:

$$\mathcal{O}'(f) = \frac{1}{N} \sum_{i=1}^{N} (f(\mathbf{x}^i) - y^i)^2.$$

Regression problem ...

Our objective function

$$\mathcal{O}'(f) = \frac{1}{N}\sum_{i=1}^{N}(f(\mathbf{x}^i) - y^i)^2.$$

Our goal is to find a function $f^* \mathbb{R}^n \to \mathbb{R}$ that minimizes $\mathcal{O}'$:

$$f^* = \operatorname*{arg\,min}_{f \in \text{All possible functions from } \mathbb{R}^n \text{ to } \mathbb{R}} \mathcal{O}'(f).$$

A related problem:

$$f^* = \operatorname*{arg\,min}_{f \in \text{All continuous functions from } \mathbb{R}^n \text{ to } \mathbb{R}} \mathcal{O}'(f).$$

Another related problem:

$$f^* = \operatorname*{arg\,min}_{f \in \text{All linear functions from } \mathbb{R}^n \text{ to } \mathbb{R}} \mathcal{O}'(f).$$

Regression problem ...

$$f^* = \operatorname*{arg\,min}_{f \in \text{All linear functions from } \mathbb{R}^n \text{ to } \mathbb{R}} \mathcal{O}'(f). \tag{1}$$

A linear function $f : \mathbb{R}^n \to \mathbb{R}$ is represented as

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}.$$

A linear function f is uniquely determined by a vector $\mathbf{w}$:
f is *parametrized* by $\mathbf{w} \in \mathbb{R}^n$.
Problem (1) is equivalent to

$$\begin{aligned}
\mathbf{w}^* &= \operatorname*{arg\,min}_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{N} \sum_{i=1}^{N} (f(\mathbf{x}^i) - y^i)^2 \\
&= \operatorname*{arg\,min}_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{N} \sum_{i=1}^{N} (\mathbf{w}^\top \mathbf{x}^i - y^i)^2.
\end{aligned}$$

Why optimization?

- Many (most) machine learning problems are eventually formulated as optimization, e.g.
  - Training decision trees.
  - Training linear (or nonlinear) regression.
  - Discovering cluster structure.
  - Training neural networks.
  - Gunnar Rätsch, "*Machine learning is statistics combined with optimization.*"
- A typical procedure for solving a machine learning problem is
  - (I) Reformulate the problem into an optimization: a function f and parameters $\mathbf{x}$.
  - (II) Simplify the optimization problem, if necessary.
  - (III) Choose a proper optimization algorithm.

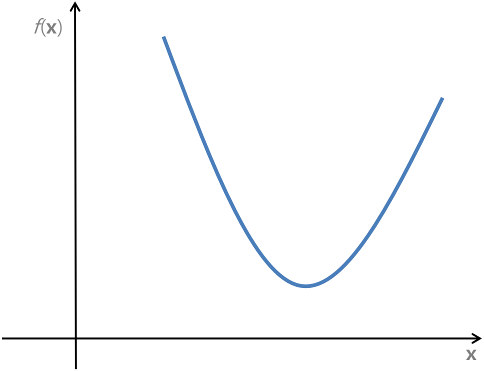
Previously in decision trees ...

- A single feature x_* is selected from n-possible features $\{x_i\}_{i=1}^n$ by
  - evaluating the information gain $IG(x_i)$ for each feature x_i.
  - selecting the index i that achieves the highest information gain IG.
- Equivalently,
  - We are given a candidate set $\mathcal{X} = \{x_1, \ldots, x_n\}$ and a function $IG : \mathcal{X} \to \mathbb{R}$.
  - The best solution x_* is obtained as the maximizer of IG:

$$x_* = \arg\max_{x \in \mathcal{X}} IG(x).$$

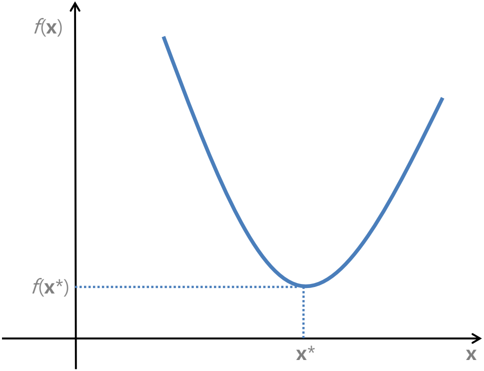
- We solved this optimization problem by evaluating IG of all candidates in $\mathcal{X}$:
  Impossible when $|\mathcal{X}| = \infty$.

A simple 1D optimization (minimization) problem



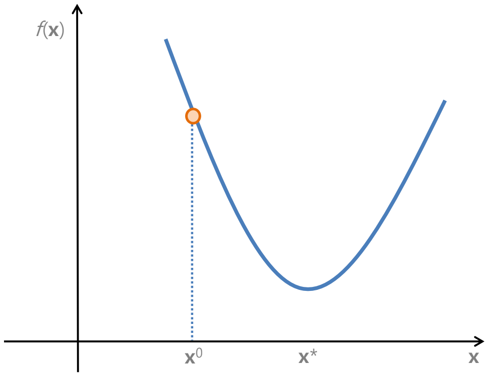
Our objective function f is one-dimensional: $f : \mathbb{R} \to \mathbb{R}$.

A simple 1D optimization (minimization) problem



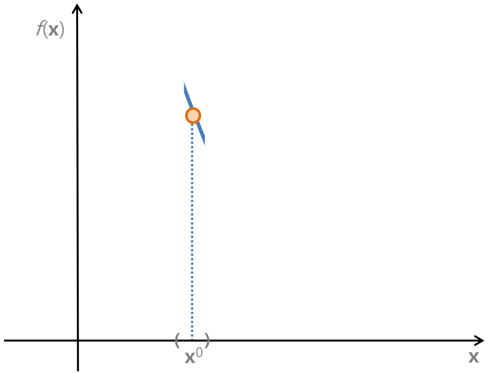
Inspecting the graph of f shows that $\mathbf{x}^*$ is the optimum point. For general high-dimensional problems ($f : \mathbb{R}^n \to \mathbb{R}$), this approach is not applicable.

A simple 1D optimization (minimization) problem



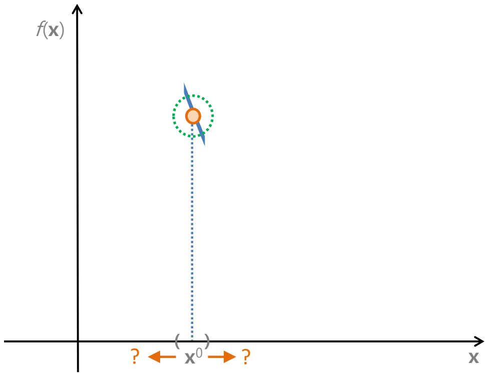
An iterative optimization method starts with an initial guess (or the initial solution) $\mathbf{x}^0$.

A simple 1D optimization (minimization) problem



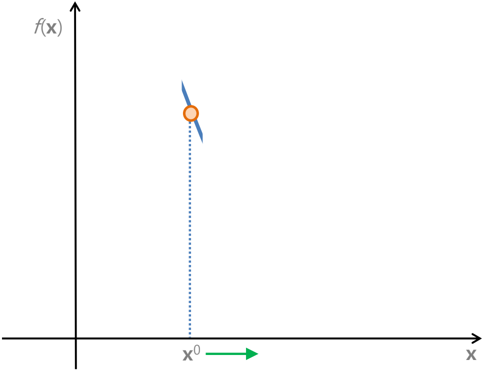
We cannot see the entire graph of f.
Instead, we can observe f in a (very small) local neighborhood of the zero-th solution $\mathbf{x}^0$.

A simple 1D optimization (minimization) problem



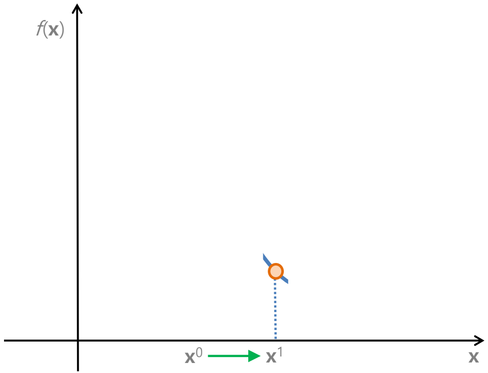
By inspecting f values around $\mathbf{x}^0$, we can decide which direction to explore.

A simple 1D optimization (minimization) problem



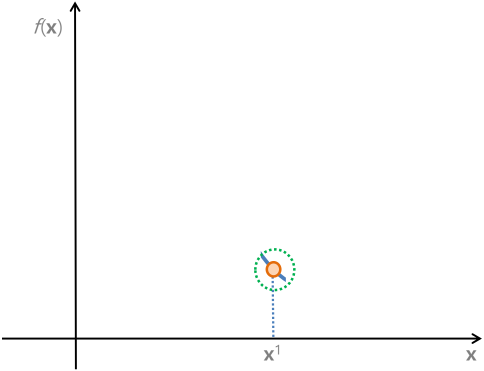
By inspecting f values around $\mathbf{x}^0$, we can decide which direction to explore.

A simple 1D optimization (minimization) problem



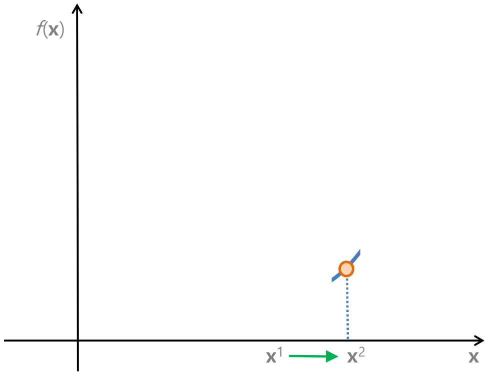
Now we are at the first solution $\mathbf{x}^1$.

A simple 1D optimization (minimization) problem



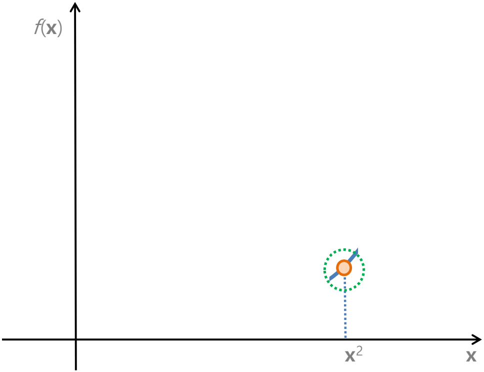
Again, observing f in a neighborhood, we decide the next direction to move.

A simple 1D optimization (minimization) problem



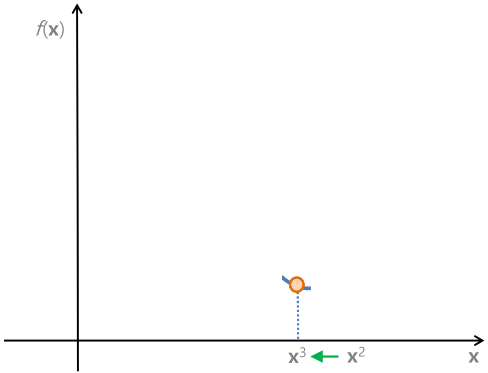
Now we are at the second solution $\mathbf{x}^2$.

A simple 1D optimization (minimization) problem



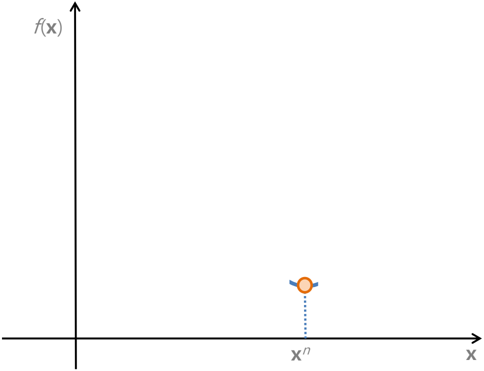
Decide the next direction to move.

A simple 1D optimization (minimization) problem



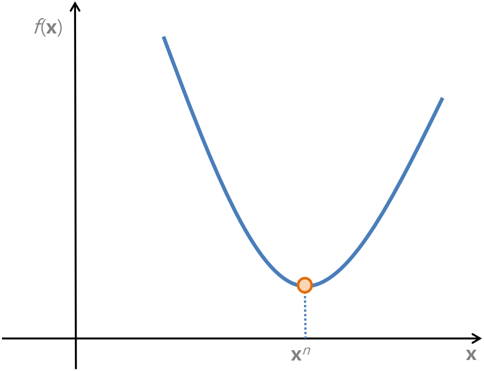
Decide the next direction to move.

A simple 1D optimization (minimization) problem



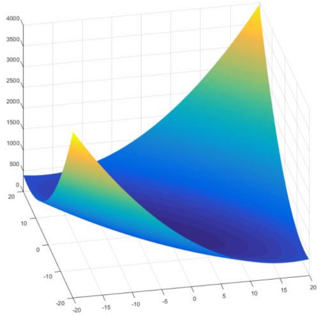
After a few iterations, we arrive at the optimum $\mathbf{x}^n = \mathbf{x}^*$.

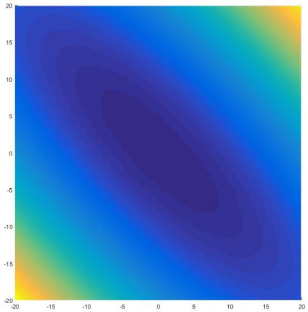
A simple 1D optimization (minimization) problem



After a few iterations, we arrive at the optimum $\mathbf{x}^n = \mathbf{x}^*$.

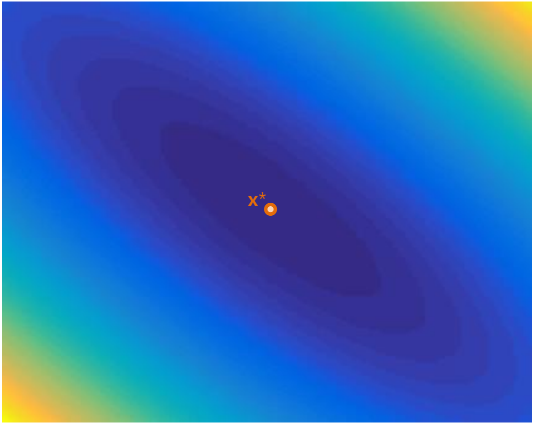
A simple 2D optimization problem





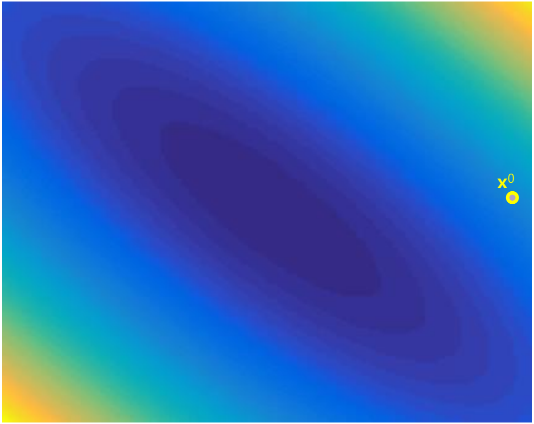
Our objective function f is two-dimensional: $f : \mathbb{R}^2 \to \mathbb{R}$.

A simple 2D optimization problem



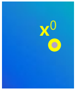
Our objective function f is two-dimensional: $f : \mathbb{R}^2 \to \mathbb{R}$.

A simple 2D optimization problem



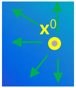
We start the optimization with an initial (zero-th) solution $\mathbf{x}^0$.

A simple 2D optimization problem



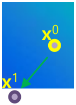
Again, we can observe f only in a small neighborhood of $\mathbf{x}^0$.

A simple 2D optimization problem



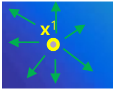
By inspecting f around $\mathbf{x}^0$ ...

A simple 2D optimization problem



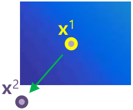
We decide a direction to move (to decrease the f value).

A simple 2D optimization problem



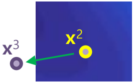
Now we are at the first solution $\mathbf{x}^1$, and observing f around ...

A simple 2D optimization problem



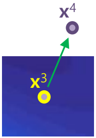
and decide a new direction ...

A simple 2D optimization problem



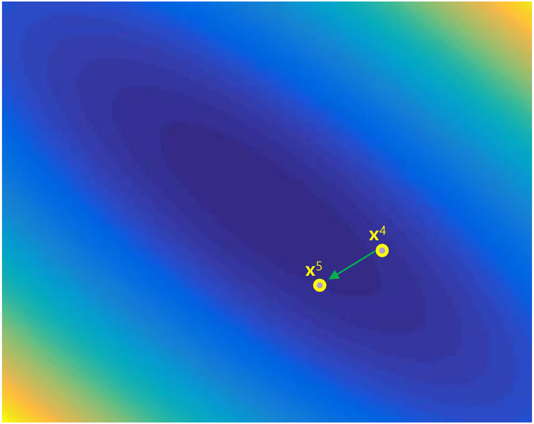
From the second solution to the third ...

A simple 2D optimization problem



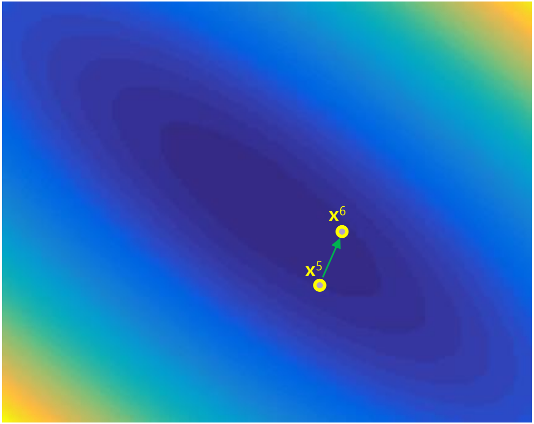
from the third solution to the fourth ...

A simple 2D optimization problem



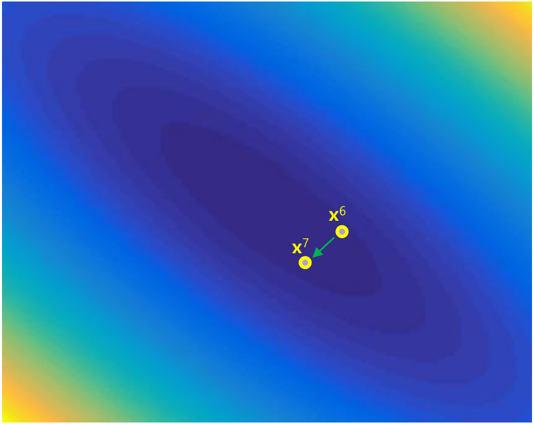
from $\mathbf{x}^4$ to $\mathbf{x}^5$ ...

A simple 2D optimization problem



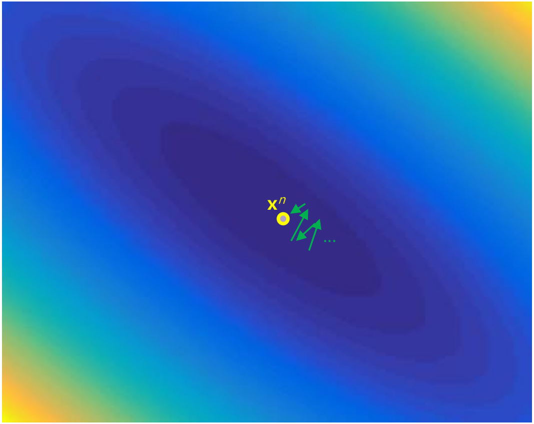
from $\mathbf{x}^5$ to $\mathbf{x}^6$ ...

A simple 2D optimization problem



from $\mathbf{x}^6$ to $\mathbf{x}^7$ ...

A simple 2D optimization problem



After a few iterations, we arrive at the optimum $\mathbf{x}^n = \mathbf{x}^*$.

Iterative descent algorithm

Summarizing our optimization approach.

(I) $t = 0$; Make an initial guess $\mathbf{x}^t$;

(II) Iterate until the termination condition is met.

   (i) Find a direction $\mathbf{p}^t$ to move;

   (ii) Decide how much (α^t) to move along $\mathbf{p}^t$ direction;

   (iii) $\mathbf{x}^{t+1} = \mathbf{x}^t + \alpha^t \mathbf{p}^t$;

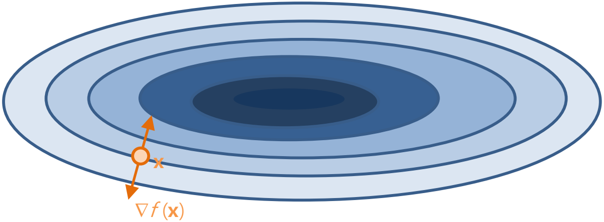
   (iv) $t = t + 1$;

How do we decide

- direction to move $\mathbf{p}^t$,
- step size α^t,
- when to stop (termination condition)?

Steepest descent

- How do we decide the direction $\mathbf{p}(t)$ to move?
- Steepest descent algorithm chooses the direction along which f decreases most rapidly.
- f increases most rapidly along the gradient ∇f direction: $\mathbf{p}(t) = -\nabla f(\mathbf{x}^i)$.



Steepest descent

How do we know that f increases most rapidly along ∇f?

Applying Taylor's theorem,

$$f(\mathbf{x} + \alpha \mathbf{p}) = f(\mathbf{x}) + \alpha \mathbf{p}^\top \nabla f(\mathbf{x}) + \frac{1}{2}\alpha^2 \mathbf{p}^\top \nabla^2 f(\mathbf{x} + t\mathbf{p})\mathbf{p},$$

for some $t \in (0, \alpha)$.

The rate of change in f along the direction $\mathbf{p}$ is α.

The optimum unit vector $\mathbf{p}^*$ ($\|\mathbf{p}^*\| = 1$) that minimizes $\mathbf{p}^\top \nabla f(\mathbf{x})$ is $-\frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}$.

Steepest descent

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \alpha^t \mathbf{p}^t;$$

- $\mathbf{p}^t = -\nabla f(\mathbf{x}^t)$.
- How do we decide the step size $\alpha^t > 0$?
  $\rightarrow$ Simple solution (among others): fix it to a constant value α.
- When to terminate the optimization process?
  $\rightarrow$ Simple solution (among others):
  terminate when $\frac{\|f(\mathbf{x}^t)\|}{\|f(\mathbf{x}^0)\|} < T$ for a constant $T > 0$.

Steepest descent algorithm

Input: the stopping condition parameter $T > 0$ and step size $\alpha > 0$;

(I) $t = 0$; Make an initial guess $\mathbf{x}^t$;

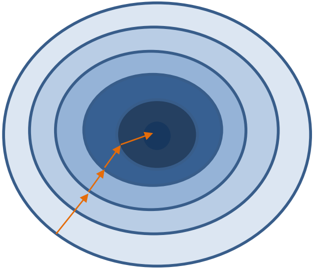
(II) Iterate until $\frac{\|f(\mathbf{x}^t)\|}{\|f(\mathbf{x}^0)\|} < T$.

   (i) $\mathbf{x}^{t+1} = \mathbf{x}^t - \alpha \nabla f(\mathbf{x}^t)$;

   (ii) $t = t + 1$;

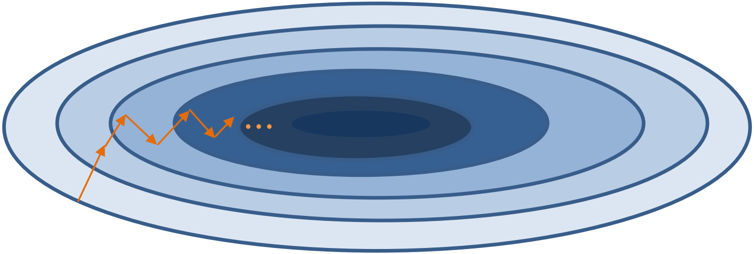
Steepest descent algorithm

Good for isotropic functions.



Steepest descent algorithm

Not good for anisotropic functions.

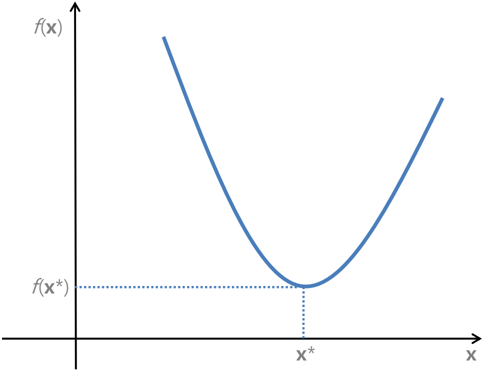


Steepest descent algorithm

Not good for anisotropic functions.

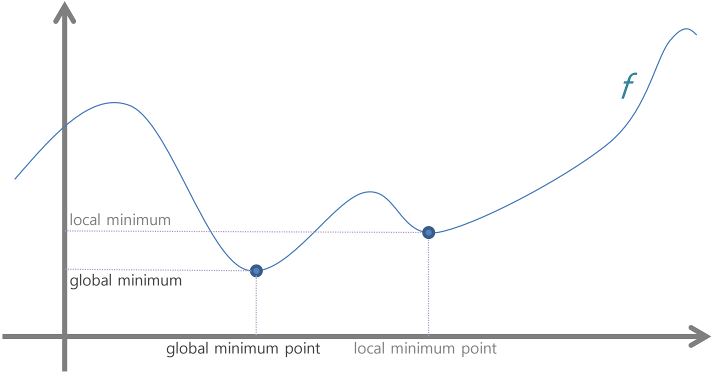
Steepest descent algorithm

finds the global minimum $\mathbf{x}^*$ for



Steepest descent algorithm

may not find the global minimum $\mathbf{x}^*$ when f looks like



Finding the global optimum is difficult even in 1D case.

- Global minimum (point) $\mathbf{x}^*$: $f(\mathbf{x}^*) \leq f(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$.
- Local minimum (point): a point which becomes a global minimum point when we restrict the domain to a (arbitrary small) neighborhood of itself.

Linear least-squares regression

Given a set of data points (pairs of input and output)

$$D = \{(\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^N, y^N)\} \subset \mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^n \times \mathbb{R},$$

Least-squares regression finds f^* that minimizes the sum of squared error:

$$f^* = \arg\min_f \sum_{i=1}^{N} (f(\mathbf{x}^i) - y^i)^2.$$

If f is linear

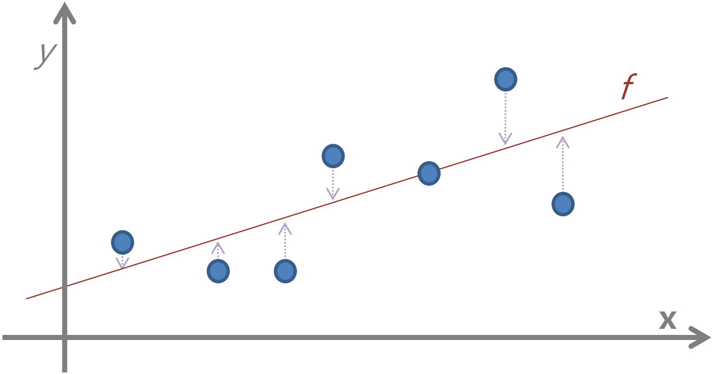
$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + c,$$

finding the optimum f^* is equivalent to finding the optimal parameter $\mathbf{w}^*$.
Offset c can be removed by replacing $\mathbf{x}$ and $\mathbf{w}$ with

$$\begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} \text{ and } \begin{pmatrix} \mathbf{w} \\ c \end{pmatrix},$$

respectively.

One-dimensional example (linear least-squares)



Linear least-squares regression

$$
\begin{aligned}
f(\mathbf{x}) &= \mathbf{w}^\top \mathbf{x}, \\
f^* &= \arg\min_{\text{Linear } f} \sum_{i=1}^{N} (f(\mathbf{x}^i) - y^i)^2 \\
\Leftrightarrow \mathbf{w}^* &= \arg\min_{\mathbf{w}\in\mathbb{R}^n} \sum_{i=1}^{N} (\mathbf{w}^\top \mathbf{x}^i - y^i)^2. \qquad (2)
\end{aligned}
$$

With data matrix $\mathbf{X} = (\mathbf{x}^1, \ldots, \mathbf{x}^N)$ and label vector $\mathbf{y} = \begin{pmatrix} y^1 \\ \vdots \\ y^N \end{pmatrix}$,

problem (2) can be rewritten as

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathbb{R}^n} \|\mathbf{X}^\top \mathbf{w} - \mathbf{y}\|^2.$$

Show that $\|\mathbf{X}^\top \mathbf{w} - \mathbf{y}\|^2 = \sum_{i=1}^{N} (f(\mathbf{x}^i) - y^i)^2$!

Steepest descent for linear least-squares regression

Objective:

$$\mathcal{O}(\mathbf{w}) = \|\mathbf{X}^\top \mathbf{w} - \mathbf{y}\|^2$$
$$\nabla_{\mathbf{w}} \mathcal{O}(\mathbf{w}) = 2\mathbf{X}(\mathbf{X}^\top \mathbf{w} - \mathbf{y})$$

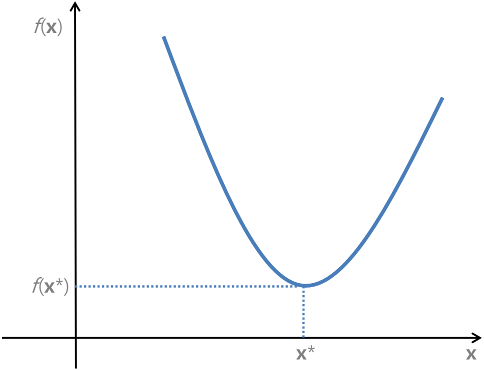
Optimization algorithm:
Given T and α,

(I) $t = 0$; Make an initial guess $\mathbf{w}^t$;

(II) Iterate until $\frac{\|f(\mathbf{w}^t)\|}{\|f(\mathbf{w}^0)\|} < T$.

  (i) $\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha \nabla \left(2\mathbf{X}(\mathbf{X}^\top \mathbf{w} - \mathbf{y})\right)$;
  (ii) $t = t + 1$;

Closed form solution for linear least-squares regression



When our optimization objective f does not have local minima other than the global minimum $\mathbf{x}^*$, $\mathbf{x}^*$ can be identified by setting the gradient ∇f to zero.

Closed form solution for linear least-squares regression

Objective:

$$\mathcal{O}(\mathbf{w}) = \|\mathbf{X}^\top \mathbf{w} - \mathbf{y}\|^2$$
$$\nabla_{\mathbf{w}} \mathcal{O}(\mathbf{w}) = 2\mathbf{X}(\mathbf{X}^\top \mathbf{w} - \mathbf{y}).$$

The minimum $\mathbf{w}^*$ satisfies

$$\begin{aligned}\nabla_{\mathbf{w}} \mathcal{O}(\mathbf{w}^*) &= 2\mathbf{X}(\mathbf{X}^\top \mathbf{w}^* - \mathbf{y}) = 0 \\ \Leftrightarrow & 2\mathbf{X}\mathbf{X}^\top \mathbf{w}^* - 2\mathbf{X}\mathbf{y} = 0 \\ \Leftrightarrow & \mathbf{X}\mathbf{X}^\top \mathbf{w}^* = \mathbf{X}\mathbf{y}.\end{aligned}$$

Two optimization approaches for linear least-squares regression

- Steepest descent:

$$\mathbf{w}^{t+1} = \mathbf{w}^{t} - \alpha\nabla\left(2\mathbf{X}\mathbf{X}^{\top}\mathbf{w} - 2\mathbf{X}\mathbf{y}\right).$$

  Complexity: $O(n \times N)$ per iteration.
  N: # data points, n: data dimensionality.

- Closed form solution:

$$\mathbf{X}\mathbf{X}^{\top}\mathbf{w}^{*} = \mathbf{X}\mathbf{y}.$$

  Complexity: $O(n^3 + N \times n^2)$.

Summary

- Almost all machine learning tasks are formulated as optimization problems.
- Steepest descent: one of the simplest optimization algorithms.
  - requires evaluating gradient.
  - good for isotropic functions.
  - Future topics: any alternatives to steepest descent, e.g. when f is anisotropic?
    How many iterations do we need (convergence analysis)?
- Iterative vs. closed form solution for linear least-squares regression.

Optimization (minimization) problem

Given a function $f : \mathcal{X} \subset \mathbb{R}^n \mapsto \mathbb{R}$, find an element $\mathbf{x}^*$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$.

- Maximization can be converted to a minimization by multiplying f by -1.
- Optimization problem can be accompanied by constraints (constrained optimization problem):

$$
\begin{aligned}
\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{X}} & f(\mathbf{x}), \\
\text{s.t. } & g_i(\mathbf{x}) \leq 0, i = \{1, \ldots, k\}, \\
& h_j(\mathbf{x}) = 0, j = \{1, \ldots, l\}.
\end{aligned}
$$

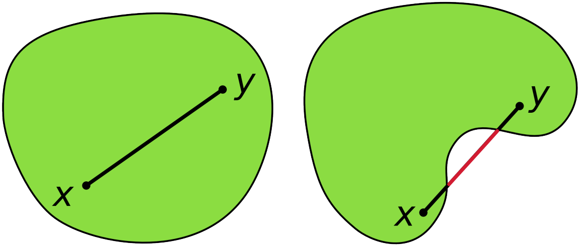
$\{g_i\}$ and $\{h_j\}$ are (inequality & equality) *constraint functions*.

Convex set

A subset C of a vector space is called convex if $\forall \mathbf{x}, \mathbf{y} \in C$ and $\lambda \in [0, 1]$

$$\lambda \mathbf{x} + (1 - \lambda)\mathbf{y} \in C.$$

examples:

- 

- $\mathbb{R}^n$
- a cone in $\mathbb{R}^n$

[Wikipedia]

Convex function

A function on a convex set C is

- convex if

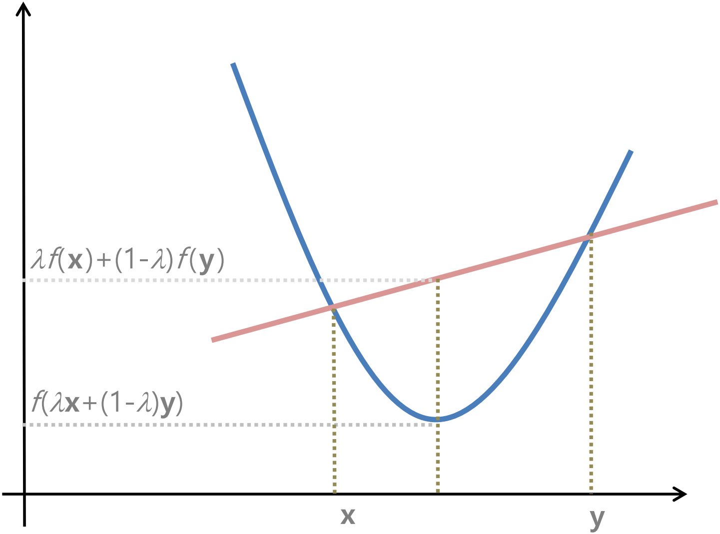
$$f(\lambda \mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}),$$

- strictly convex if

$$f(\lambda \mathbf{x} + (1-\lambda)\mathbf{y}) < \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y})$$

$\forall \mathbf{x}, \mathbf{y} \in C$ and $\lambda \in [0, 1]$.

Convex function



Convex function

If f is twice continuously differentiable,

- $f : \mathbb{R} \mapsto \mathbb{R}$ is convex if $\frac{d^2 f}{d\mathbf{x}^2}$ is positive everywhere.
- $f : \mathbb{R}^n \mapsto \mathbb{R}$ is convex if the Hessian matrix is positive definite everywhere.

Example: second-order polynomial

$$p(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$$

is convex if **A** is (symmetric) positive definite.

The Hessian matrix $\frac{d^2 p}{d\mathbf{x}^2}[\mathbf{x}]$ of p is A for all $\mathbf{x}$.

(symmetric) Positive definite matrix

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} > 0, \forall \mathbf{x} \neq 0$$

- $\mathbf{A}$ is invertible and $\mathbf{A}^{-1}$ is positive definite.
- Eigenvalues of $\mathbf{A}$ are positive:

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} = \mathbf{x}^\top(\mathbf{E}\Lambda\mathbf{E}^{-1})\mathbf{x} = \mathbf{x}^\top(\mathbf{E}\Lambda\mathbf{E}^\top)\mathbf{x} = (\mathbf{E}^\top\mathbf{x})^\top\Lambda(\mathbf{E}^\top\mathbf{x}).$$

$$\Rightarrow \mathbf{x}^\top \mathbf{A}\mathbf{x} > 0 \text{ if elements of } \Lambda \text{ are positive.}$$

Optimization problem types

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}),$$
$$\text{s.t. } g_i(\mathbf{x}) \leq 0, i = \{1, \dots, k\},$$
$$h_j(\mathbf{x}) = 0, j = \{1, \dots, l\}.$$

- Constrained vs. unconstrained optimization.
- Discrete vs. continuous optimization.
- Deterministic vs. stochastic optimization.

Linear programming

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^n} \mathbf{c}^\top \mathbf{x},$$
$$\text{s.t. } A\mathbf{x} = \mathbf{b},$$
$$\mathbf{x} \geq 0.$$

Example: a company manufactures two types of boxes, A and B.

- The boxes undergo two major processes: cutting and pinning operations.
- The profits per unit are 6 for A and 4 for B.
- A requires 2 minutes for cutting and 3 minutes for pinning.
- B requires 2 minutes for cutting and 1 minutes for pinning.
- Available operating time is 120 minutes and 60 minutes for cutting and pinning machines.
- We have to decide the optimum A and B quantities to maximize the profits.

Linear programming

- The boxes undergo two major processes: cutting and pinning operations.
- The profits per unit are 6 for A and 4 for B.
- A requires 2 minutes for cutting and 3 minutes for pinning.
- B requires 2 minutes for cutting and 1 minutes for pinning.
- Available operating time is 120 minutes and 60 minutes for cutting and pinning machines.
- We have to decide the optimum A and B quantities to maximize the profits.

$$
\begin{aligned}
&\mathbf{x} = [x_A, x_B]^\top \\
&\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathbb{R}^2} f(\mathbf{x}) = 5x_A + 4x_B \text{ (objective function)} \\
&\qquad\qquad \text{s.t. } 2x_A + 3x_B \leq 120 \text{ (cutting constraint)} \\
&\qquad\qquad\qquad 2x_A + x_B \leq 60 \text{ (pinning constraint)} \\
&\qquad\qquad\qquad x_A, x_B \geq 0 \text{ (box quantities cannot be negative)}\ .
\end{aligned}
$$

Integer programming

$$\mathbf{x} = [x_A, x_B]^\top$$

$$\begin{aligned}\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathbb{R}^2} f(\mathbf{x}) &= 5x_A + 4x_B \text{ (objective function)}\\ \text{s.t. } & 2x_A + 3x_B \leq 120 \text{ (cutting constraint)}\\ & 2x_A + x_B \leq 60 \text{ (pinning constraint)}\\ & x_A, x_B \geq 0 \text{ (box quantities cannot be negative)} .\end{aligned}$$

If box quantities need to be integers ...

$$\begin{aligned}\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathbb{Z}^2} \; & 5x_A + 4x_B,\\ \text{s.t. } & 2x_A + 3x_B \leq 120,\\ & 2x_A + x_B \leq 60\\ & x_A, x_B \geq 0.\end{aligned}$$

- Quadratic programming:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) = \mathbf{x}^\top Q\mathbf{x} + \mathbf{c}^\top\mathbf{x},$$
$$\text{s.t. } A\mathbf{x} \le \mathbf{b},$$

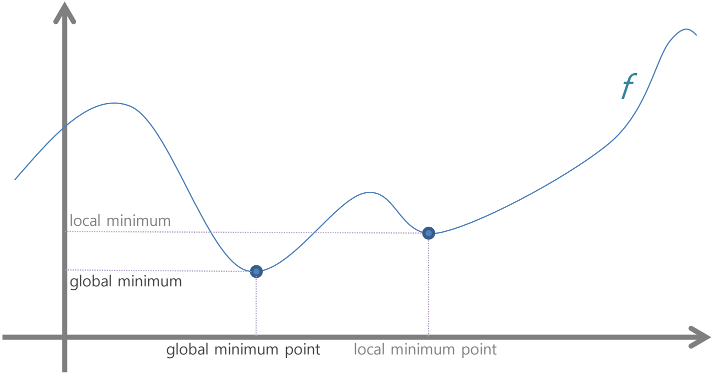
  where Q is a real symmetric matrix.

- Nonlinear programming:

$$\begin{aligned} \mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}),& \\ \text{s.t. } g(\mathbf{x}) &\le 0, \\ h(\mathbf{x}) &= 0, \end{aligned}$$

  where f, g, h are nonlinear.

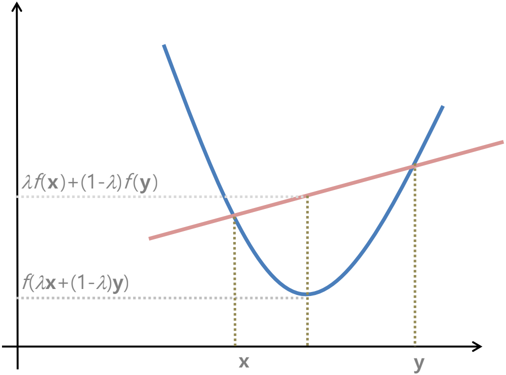
Optimization



Finding the global optimum is difficult even in 1D case.

- Global minimum (point) $\mathbf{x}^*$: $f(\mathbf{x}^*) \leq f(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$.
- Local minimum (point): a point which becomes a global minimum point when we restrict the domain to a (arbitrary small) neighborhood of itself.

Convex optimization



$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}),$$
$$\text{s.t. } g(\mathbf{x}) \leq 0,$$

where $f, g : \mathbb{R}^n \to \mathbb{R}$ are convex.

Convex function

- If f is twice continuously differentiable, $f : \mathbb{R} \mapsto \mathbb{R}$ is convex if $\frac{d^2 f}{d\mathbf{x}^2}$ is positive everywhere.
- If f is twice continuously differentiable, $f : \mathbb{R}^n \mapsto \mathbb{R}$ is convex if the Hessian matrix $Hf|_{\mathbf{x}}$ is positive definite everywhere.
- The second order polynomial

$$p(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$$

  is convex if $\mathbf{A}$ is positive definite.
- An increasing function of a convex function is convex, e.g. if $f(x)$ is convex, $\log(f(x))$ and $\exp(f(x))$ are convex.
- If f and g are convex, $f + g$ is convex.
- If a local optimum exists for a convex function f, it is a global minimum.
- A point $\mathbf{x}^*$ is a local minimum if $\nabla_f(\mathbf{x}^*) = 0$.

Positive definite matrix

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} > 0, \forall \mathbf{x} \neq 0$$

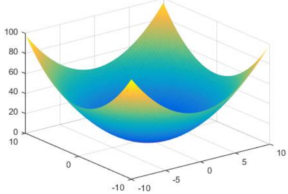
- $\mathbf{A}$ is invertible and $\mathbf{A}^{-1}$ is positive definite.
- Eigenvalues of $\mathbf{A}$ are positive:

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} = \mathbf{x}^\top (\mathbf{E}\Lambda\mathbf{E}^{-1})\mathbf{x} = \mathbf{x}^\top (\mathbf{E}\Lambda\mathbf{E}^\top)\mathbf{x} = (\mathbf{E}^\top \mathbf{x})^\top \Lambda (\mathbf{E}^\top \mathbf{x}).$$
$$\Rightarrow \mathbf{x}^\top \mathbf{A}\mathbf{x} > 0 \text{ if elements of } \Lambda \text{ are positive.}$$

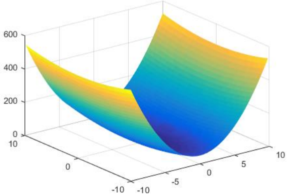
2D examples

$$p(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{A}\mathbf{x}.$$

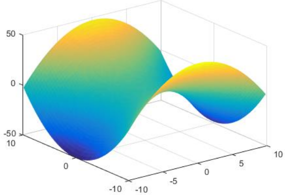
$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$



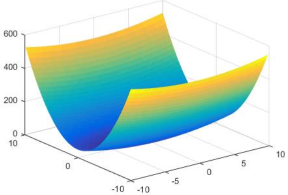
$\mathbf{A} = \begin{pmatrix} 10 & 0 \\ 0 & 1 \end{pmatrix}$



$\mathbf{A} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$



$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 10 \end{pmatrix}$



2D example 1

Eigen-decomposition of the diagonal matrix $\mathbf{A} = \begin{pmatrix} 10 & 0 \\ 0 & 1 \end{pmatrix}$:

$$\begin{aligned}
\mathbf{A} &= \mathbf{E}\Lambda\mathbf{E}^{-1} = \mathbf{E}\Lambda\mathbf{E}^{\top} \\
&= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 10 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\
&= (\mathbf{e}^1, \mathbf{e}^2)\begin{pmatrix} \lambda^1 & 0 \\ 0 & \lambda^2 \end{pmatrix}(\mathbf{e}^1, \mathbf{e}^2)^{\top} \\
&= \lambda^1 \mathbf{e}^1 {\mathbf{e}^1}^{\top} + \lambda^2 \mathbf{e}^2 {\mathbf{e}^2}^{\top},
\end{aligned}$$

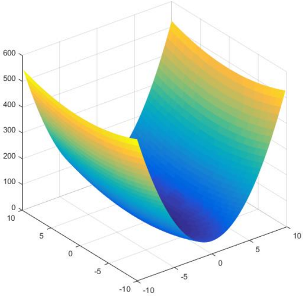
where

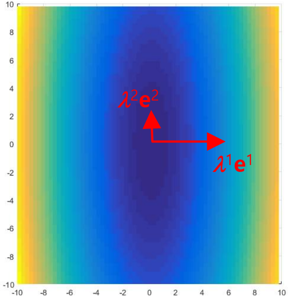
$$\mathbf{e}^1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \mathbf{e}^2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \lambda^1 = 10, \lambda^2 = 1.$$

Note: Eigenvectors of $\mathbf{A}$ form a basis.

2D example 1

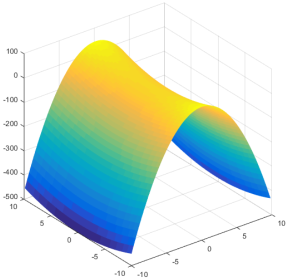
$$p(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{A}\mathbf{x}, \mathbf{A} = \begin{pmatrix} 10 & 0 \\ 0 & 1 \end{pmatrix} = \lambda^1 \mathbf{e}^1 {\mathbf{e}^1}^\top + \lambda^2 \mathbf{e}^2 {\mathbf{e}^2}^\top.$$

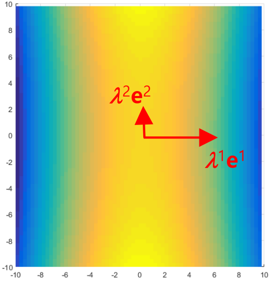




2D example 2

$$p(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{A}\mathbf{x}, \mathbf{A} = \begin{pmatrix} -10 & 0 \\ 0 & 1 \end{pmatrix} = \lambda^1 \mathbf{e}^1 {\mathbf{e}^1}^\top + \lambda^2 \mathbf{e}^2 {\mathbf{e}^2}^\top.$$





2D example 3

Eigen-decomposition of symmetric matrix $\mathbf{A} = \begin{pmatrix} 5.5 & 4.5 \\ 4.5 & 5.5 \end{pmatrix}$:

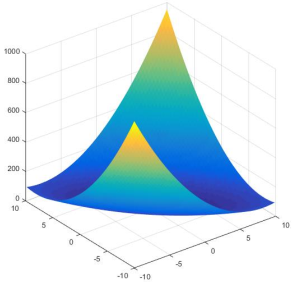
$$\begin{aligned}
\mathbf{A} &= \mathbf{E}\Lambda\mathbf{E}^{-1} = \mathbf{E}\Lambda\mathbf{E}^{\top} \\
&= \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 10 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \\
&= (\mathbf{e}^1, \mathbf{e}^2) \begin{pmatrix} \lambda^1 & 0 \\ 0 & \lambda^2 \end{pmatrix} (\mathbf{e}^1, \mathbf{e}^2)^{\top} \\
&= \lambda^1 \mathbf{e}^1 {\mathbf{e}^1}^{\top} + \lambda^2 \mathbf{e}^2 {\mathbf{e}^2}^{\top},
\end{aligned}$$

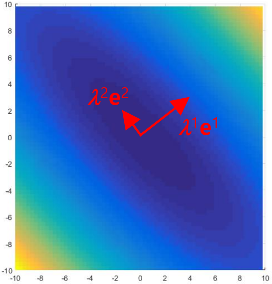
where

$$\mathbf{e}^1 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}, \mathbf{e}^2 = \begin{pmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}, \lambda^1 = 10, \lambda^2 = 1.$$

2D example 3

$$p(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{A}\mathbf{x}, \mathbf{A} = \begin{pmatrix} 5.5 & 4.5 \\ 4.5 & 5.5 \end{pmatrix} = \lambda^1 \mathbf{e}^1 {\mathbf{e}^1}^\top + \lambda^2 \mathbf{e}^2 {\mathbf{e}^2}^\top.$$





References

Noc J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer (second edition).

Ber D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific (second edition).

Teu Teukolsky, Vetterling, and Flannery, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press (any edition). `http://www.nr.com/`

Pet K. B. Petersen and M. S. Pedersen, *The Matrix Cookbook*.