Introduction to gradient-based optimization

First and second order methods

Naive gradient, stochastic gradient & accelerated gradient

Outline

Outline

Semi-parametric modelling - logistic regression

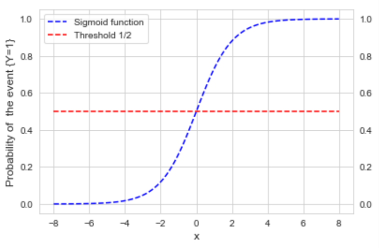
→ The objective is to **predict the label $Y \in \{0,1\}$** based on $X \in \mathbb{R}^d$.

→ Logistic regression **models the distribution of Y given X**.

$$\mathbb{P}(Y = 1|X) = \sigma(\langle w, X\rangle + b)\,,$$

where $w \in \mathbb{R}^d$ is a vector of model **weights** and $b \in \mathbb{R}$ is the **intercept**, and where σ is the **sigmoid** function.

$$\sigma : z \mapsto \frac{1}{1+e^{-z}}\,.$$



→ The sigmoid function is a **model choice to map $\mathbb{R}$ into $(0,1)$**.

→ Another widespread solution for σ is $\sigma : z \mapsto \mathbb{P}(Z \leqslant z)$ where $Z \sim \mathcal{N}(0,1)$, which leads to a **probit** regression model.

Logistic regression

→ $\{(X_i, Y_i)\}_{1 \leqslant i \leqslant n}$ are **i.i.d. with the same distribution as** (X, Y).

Likelihood

$$\prod_{i=1}^{n} \mathbb{P}(Y_i|X_i) = \prod_{i=1}^{n} \sigma(\langle w, X_i \rangle + b)^{Y_i} \big(1 - \sigma(\langle w, X_i \rangle + b)\big)^{1-Y_i}\,,$$
$$= \prod_{i=1}^{n} \sigma(\langle w, x_i \rangle + b)^{Y_i} \sigma(-\langle w, X_i \rangle - b)^{1-Y_i}$$

and the **normalized negative loglikelihood** is

$$f(w, b) = \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, \langle w, X_i \rangle + b)\,.$$

Logistic regression

Compute $\hat{w}_n$ and $\hat{b}_n$ as follows:

$$(\hat{w}_n, \hat{b}_n) \in \operatorname{argmin}_{w \in \mathbb{R}^d, b \in \mathbb{R}} \tfrac{1}{n} \textstyle\sum_{i=1}^n \left(-Y_i (X_i^T w + b) + \log(1 + e^{X_i^T w + b}) \right) .$$

→ It is an **average of losses**, one for each sample point.

→ It is a **convex and smooth problem**.

Using the **logistic loss** function

$$\ell : (y, y') \mapsto \log(1 + e^{-yy'})$$

yields

$$(\hat{w}_n, \hat{b}_n) \in \underset{w \in \mathbb{R}^d, b \in \mathbb{R}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^n \ell(Y_i, \langle w, X_i \rangle + b) .$$

Maximum likelihood estimate

Assume for now that the intercept is 0. Then, the likelihood is,

$$L_n(w) = \prod_{i=1}^{n} \left(\frac{e^{X_i^T w}}{1 + e^{X_i^T w}} \right)^{Y_i} \left(\frac{1}{1 + e^{X_i^T w}} \right)^{1-Y_i} = \prod_{i=1}^{n} \left(\frac{e^{X_i^T w Y_i}}{1 + e^{X_i^T w}} \right) .$$

And the **negative** log-**likelihood** is

$$\ell_n(w) = -\log(L_n(w)) = \sum_{i=1}^{n} \left(-Y_i X_i^T w + \log(1 + e^{X_i^T w}) \right) .$$

Derivatives

$$\begin{aligned} \frac{\partial \left(\log(L_n(w))\right)}{\partial w_j} &= \sum_{i=1}^{n} \left(Y_i X_{ij} - \frac{x_{ij} e^{X_i^T w}}{(1 + e^{X_i^T w})} \right) \\ &= \sum_{i=1}^{n} X_{ij} \left(Y_i - \sigma(\langle w, X_i \rangle) \right) . \end{aligned}$$

→ **No explicit solution** for the maximizer of the loglikelihood... Parameter estimate obtained using **gradient based optimization**.

Outline

General optimization problem

Parameter inference in machine learning often boils down to solving

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} f(w) + g(w)\,,$$

with f a **goodness-of-fit functio based on a loss ℓ**,

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle w, x_i \rangle)$$

and

$$g(w) = \lambda \mathrm{pen}(w)\,,$$

where $\lambda > 0$ and **$\mathrm{pen}(\cdot)$ is some penalization function**.

→ $\mathrm{pen}(w) = \|w\|_2^2$ (Ridge).

→ $\mathrm{pen}(w) = \|w\|_1$ (Lasso).

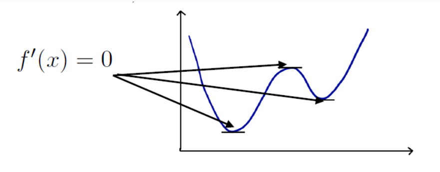
Outline

Outline

First order necessary condition

→ **In dimension one**.

Let $f : \mathbb{R} \to \mathbb{R}$ be a differentiable function. If $x^\star$ is a local extremum (minimum/maximum) then $f'(x^\star) = 0$.



→ **Generalization for $d > 1$**.

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a differentiable function. If $x^\star$ is a local extremum then $\nabla f(x^\star) = 0$.

Points such that $\nabla f(x^\star) = 0$ are called **critical points**.

Critical points are not always extrema (consider $x \mapsto x^3$).

Gradient

The gradient of a function $f : \mathbb{R}^d \to \mathbb{R}$ in $x \in \mathbb{R}^d$, denoted by $\nabla f(x)$, is **the vector of partial derivatives**:

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_d} \end{pmatrix}.$$

Some useful gradients

→ If $f : \mathbb{R} \to \mathbb{R}$, $\nabla f(x) = f'(x)$.

→ $f : x \mapsto \langle a, x \rangle$: $\nabla f(x) = a$.

→ $f : x \mapsto x^T A x$: $\nabla f(x) = (A + A^T)x$.

→ Particular case: $f : x \mapsto \|x\|^2$, $\nabla f(x) = 2x$.

Heuristic: why gradient descent works?

For a function $f : \mathbb{R}^d \to \mathbb{R}$, define the **level sets**:

$$\mathcal{C}_c = \{\mathbf{x} \in \mathbb{R}^d, f(\mathbf{x}) = c\}\,.$$

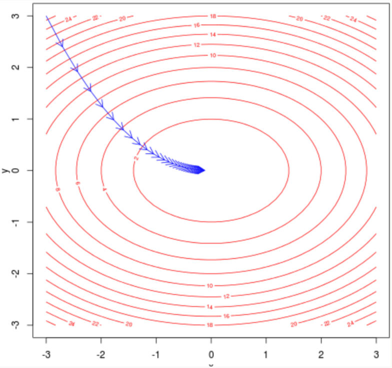


Figure 1: Gradient descent for function $f : (x, y) \mapsto x^2 + 2y^2$

$\to$ The gradient is **orthogonal to level sets**.

Convexity

Convexity - Definition

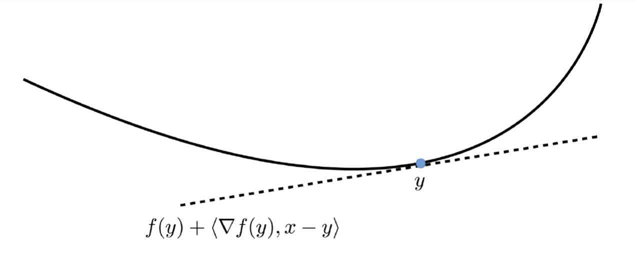
A function $f : \mathbb{R}^d \to \mathbb{R}$ is **convex** on $\mathbb{R}^d$ if, for all $x, y \in \mathbb{R}^d$, for all $\lambda \in [0, 1]$,

$$f(\lambda x + (1 - \lambda)y) \leqslant \lambda f(x) + (1 - \lambda)f(y).$$

Convexity - First derivative

A **differentiable function** $f : \mathbb{R}^d \to \mathbb{R}$ is convex if and only if, for all $x, y \in \mathbb{R}^d$,

$$f(x) \geqslant f(y) + \langle \nabla f(y), x - y \rangle .$$



Hessian

If $f : \mathbb{R}^d \to \mathbb{R}$ is **twice differentiable**, the **Hessian matrix in $x \in \mathbb{R}^d$** denoted by $\nabla^2 f(x)$ is given by

$$\nabla^2 f(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(x) & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_d}(x) \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(x) & \frac{\partial^2 f}{\partial x_2^2}(x) & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_d}(x) \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1}(x) & \frac{\partial^2 f}{\partial x_d \partial x_2}(x) & \dots & \frac{\partial^2 f}{\partial x_d^2}(x) \end{pmatrix}.$$

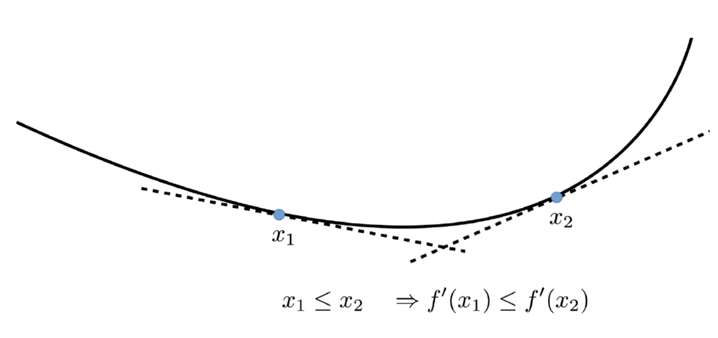
The **Hessian matrix is symmetric if f is twice continuously differentiable**.

Convexity - Hessian

A **twice differentiable function** $f : \mathbb{R}^d \to \mathbb{R}$ is convex if and only if, for all $x \in \mathbb{R}^d$,

$$\nabla^2 f(x) \succeq 0\,,$$

that is $h^T \nabla^2 f(x) h \geqslant 0$, for all $h \in \mathbb{R}^d$.



Optimality conditions: second order

Assume that f is twice continuously differentiable.

Necessary condition

If $x^\star$ is a local minimum, then $\nabla f(x^\star) = 0$ **and** $\nabla^2 f(x^\star)$ **is positive semi-definite**.

Sufficient condition

If $\nabla f(x^\star) = 0$ **and** $\nabla^2 f(x^\star)$ **is positive definite** then $x^\star$ is a strict local optimum.

For $d = 1$, this condition boils down to $f'(x^\star) = 0$ **and** $f''(x^\star) > 0$.

Gradient descent algorithms are **iterative procedures**. There are two classes of such algorithms, depending on the information that is used to compute the next iteration.

First-order algorithms that use f and ∇f. Standard algorithms when f is differentiable and convex.

Second-order algorithms that use f, ∇f and $\nabla^2 f$. They are useful when computing the Hessian matrix is not too costly.

Gradient descent algorithm

Gradient descent

Input: Function f to minimize, initial vector $w^{(0)}$, $k = 0$.

Parameters: step size $\eta > 0$.

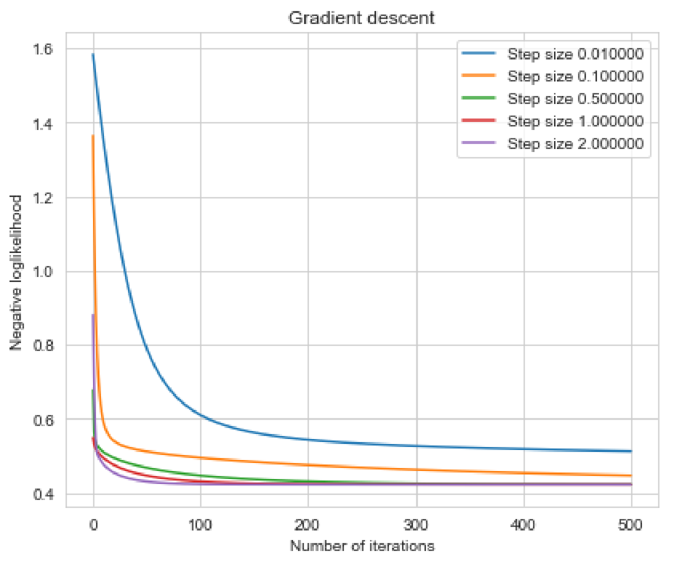
While *not converge* do

$$\rightharpoonup w^{(k+1)} = w^{(k)} - \eta_{k+1} \nabla f\big(w^{(k)}\big).$$

$$\rightharpoonup k = k + 1.$$

Output: $w^{(n_*)}$ where n_* is the last iteration.

Gradient descent in practice



When does gradient descent converge?

Convex function

A function $f : \mathbb{R}^d \to \mathbb{R}$ is **convex** on $\mathbb{R}^d$ if, for all $x, y \in \mathbb{R}^d$, for all $\lambda \in [0, 1]$,

$$f(\lambda x + (1 - \lambda)y) \leqslant \lambda f(x) + (1 - \lambda)f(y).$$

L-smooth function

A function f is said to be **L-smooth** if f is differentiable and if, for all $x, y \in \mathbb{R}^d$,

$$\|\nabla f(x) - \nabla f(y)\| \leqslant L\|x - y\|\,.$$

If f is **twice differentiable**, this is equivalent to writing that for all $x \in \mathbb{R}^d$,

$$\lambda_{max}(\nabla^2 f(x)) \leqslant L.$$

Theorem

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a **L-smooth convex function**. Let $w^\star$ be a minimum of f on $\mathbb{R}^d$. Then, Gradient Descent with step size $\eta \leqslant 1/L$ satisfies

$$\boxed{f(w^{(k)}) - f(w^\star) \leqslant \frac{\|w^{(0)} - w^\star\|_2^2}{2\eta k}\,.}$$

In particular, for $\eta = 1/L$,

$$L\|w^{(0)} - w^\star\|_2^2/2$$

iterations are sufficient to get an **ε-approximation of the minimal value of f**.

A key point: the descent lemma

If f is ***L*-smooth**, then for any $w, w' \in \mathbb{R}^d$,

$$\boxed{f(w') \leqslant f(w) + \langle \nabla f(w), w' - w \rangle + \tfrac{L}{2} \|w - w'\|_2^2 \,.}$$

Using the descent Lemma,

$$\begin{aligned} &\operatorname*{argmin}_{w \in \mathbb{R}^d} \left\{ f(w^k) + \langle \nabla f(w^k), w - w^k \rangle + \frac{L}{2} \|w - w^k\|_2^2 \right\} \\ &= \operatorname*{argmin}_{w \in \mathbb{R}^d} \left\| w - \left(w^k - \frac{1}{L} \nabla f(w^k) \right) \right\|_2^2 . \end{aligned}$$

Hence, it is natural to choose

$$w^{k+1} = w^k - \frac{1}{L} \nabla f(w^k) \,.$$

This is the most standard **gradient descent** algorithm.

Faster rate for strongly convex function

Strong convexity

A function $f : \mathbb{R}^d \to R$ is **μ-strongly convex** if

$$x \mapsto f(x) - \frac{\mu}{2}\|x\|_2^2$$

is convex.

If f is differentiable it is equivalent to, for all $x \in \mathbb{R}^d$,

$$\lambda_{min}(\nabla^2 f(x)) \geqslant \mu\,.$$

This is also equivalent to, for all $x, y \in \mathbb{R}^d$,

$$f(y) \geqslant f(x) + \langle \nabla f(x), y - x\rangle + \frac{\mu}{2}\|y - x\|_2^2.$$

Theorem

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a **L-smooth, μ strongly convex function**. Let $w^\star$ be a minimum of f on $\mathbb{R}^d$. Then, Gradient Descent with step size $\eta \leqslant 1/L$ satisfies

$$\boxed{f(w^{(k)}) - f(w^\star) \leqslant \Big(1 - \eta\mu\Big)^k \|f(w^{(0)}) - f(w^\star)\|_2^2\,.}$$

How to choose η?

Exact line search

At each step, choose the best η by optimizing

$$\eta^{(k)} = \underset{\eta > 0}{\text{argmin}}\, f(w - \eta \nabla f(w)) .$$

$\rightarrow$ **Computationally very intensive...**

Backtracking line search

Let $0 < \beta < 1$, then at each iteration, start with $\eta_k = 1$ and while

$$f(w^{(k)} - \eta_k \nabla f(w^{(k)})) - f(w^{(k)}) > -\frac{\eta_k}{2} \|\nabla f(w^{(k)})\|^2,$$

update $\eta_k \leftarrow \beta \eta_k$.

$\rightarrow$ **Simple and work pretty well in practice.**

If $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is a **L-smooth convex function**, then, Gradient Descent with backtracking line search satisfies

$$f(w^{(k)}) - f(w^\star) \leqslant \frac{\|w^{(0)} - w^\star\|_2^2}{2k \min(1, \beta/L)} .$$

Outline

Stochastic Gradient Descent (SGD)

Previous methods are based on **full gradients**, since each iteration requires the computation of

$$\nabla f(w) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w),$$

which depends on the whole dataset.

If **n is large, computing $\nabla f(w)$ is computationally expensive**.

If I is chosen uniformly at random in $\{1, \ldots, n\}$, then

$$\boxed{\mathbb{E}[\nabla f_I(w)] = \tfrac{1}{n}\textstyle\sum_{i=1}^{n} \nabla f_i(w) = \nabla f(w)\,,}$$

$\nabla f_I(w)$ is an **unbiased** but very noisy estimate of the full gradient $\nabla f(w)$.

Computation of $\nabla f_I(w)$ only requires the I-th observation.

Stochastic Gradient Descent (SGD)

Stochastic gradient descent algorithm

Input: starting point $w^{(0)}$, steps (learning rates) η_k

For $k = 1, 2, \ldots$ until *convergence* do

$\rightarrow$ Pick at random (uniformly) I_k in $\{1, \ldots, n\}$.

$\rightarrow$ compute

$$w^{(k)} = w^{(k-1)} - \eta_k \nabla f_{I_k}(w^{(k-1)}) .$$

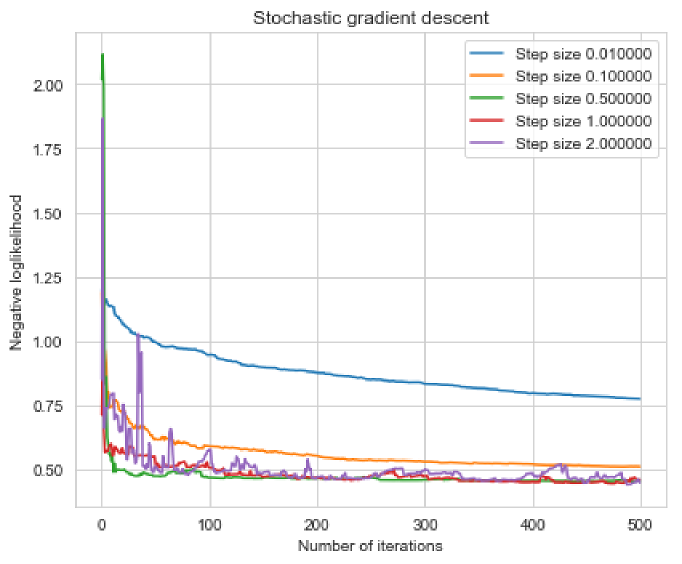
Return last $w^{(k)}$.

Remarks

$\rightarrow$ Each iteration **has complexity $O(d)$ instead of $O(nd)$ for full gradient methods**.

$\rightarrow$ Possible to reduce this to $O(s)$ when features are s-sparse using **lazy-updates**.

Stochastic gradient descent in practice (I)



Convergence rate of SGD

Project each estimate into the ball $B(0, R)$ with $R > 0$ fixed.

Let

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) .$$

Theorem

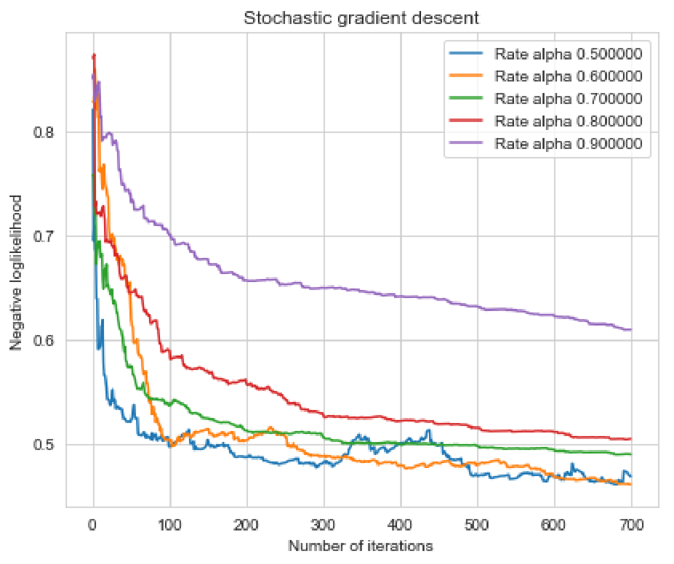
Assume that f **is convex** and that there exists $b > 0$ satisfying, for all $x \in B(0, R)$,

$$\|\nabla f_i(x)\| \leqslant b .$$

Assume also that **all minima of f belong to $B(0, R)$**. Then, setting $\eta_k = 2R/(b\sqrt{k})$,

$$\boxed{\mathbb{E}\left[f\left(\frac{1}{k} \sum_{j=1}^{k} w^{(j)} \right) \right] - f(w^\star) \leqslant \frac{3Rb}{\sqrt{k}} .}$$

Stochastic gradient descent in practice (II)



Outline

Nesterov Accelerated Gradient Descent

Input: starting point $w^{(0)}$, learning rate $\eta_k > 0$, initial velocity $v^{(0)} = 0$, momentum $\beta_k \in [0, 1]$.

While *not converge* do

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

$\rightharpoonup$ $w^{(k+1)} = v^{(k+1)} + \beta_{k+1}(v^{(k+1)} - v^{(k)})$.

Return last $w^{(k+1)}$.

Rate of convergence of Nesterov accelerated gradient (NAG)

Theorem

Assume that f is a **L-smooth, convex function whose minimum is reached at $w^\star$**. Then, if $\beta_{k+1} = k/(k+3)$,

$$\boxed{f(w^{(k)}) - f(w^\star) \leqslant \frac{2\|w^{(0)} - w^\star\|_2^2}{\eta(k+1)^2}\,.}$$

Theorem

Assume that f is a **L-smooth, μ strongly convex function whose minimum is reached at $w^\star$**. Then, choosing

$$\beta_k = \frac{1 - \sqrt{\mu/L}}{1 + \sqrt{\mu/L}},$$

yields

$$\boxed{f(w^{(k)}) - f(w^\star) \leqslant \frac{\|w^{(0)} - w^\star\|_2^2}{\eta}\left(1 - \sqrt{\frac{\mu}{L}}\right)^k\,.}$$

Rate of Coordinate Gradient Descent

Theorem - Nesterov (2012)

Assume that **f is convex and smooth and that each f^j is L_j-smooth**.

Consider a sequence $\{w^k\}$ given by CGD with $\eta_j = 1/L_j$ and coordinates chosen at random: i.i.d and uniform distribution in $\{1, \dots, d\}$. Then,

$$\boxed{\mathbb{E}[f(w^{k+1})] - f(w^\star) \leqslant \frac{n}{n+k}\Big(\big(1 - \tfrac{1}{n}\big)(f(w^0) - f(w^\star)) + \tfrac{1}{2}\|w^0 - w^\star\|_L^2\Big),}$$

with $\|w\|_L^2 = \sum_{j=1}^d L_j w_j^2$.

→ **Bound in expectation**, since coordinates are taken at random.

→ For **cycling cordinates $j = (k \mod d) + 1$ the bound is much worse**.