How to Evolve Gradient Descent into Evolution Strategies and CMA-ES

Nikolaus Hansen
Inria
CMAP, CNRS, Ecole Polytechnique, IP Paris, France

Presented at CMAP 2019

Outline

- Preliminaries / Context
- From gradient descent to evolution strategies
- A second order (variable metric) evolution strategy: CMA-ES

Context: Objective

minimize an objective function

$$f : \mathbb{R}^n \to \mathbb{R},\ x \mapsto f(x)$$

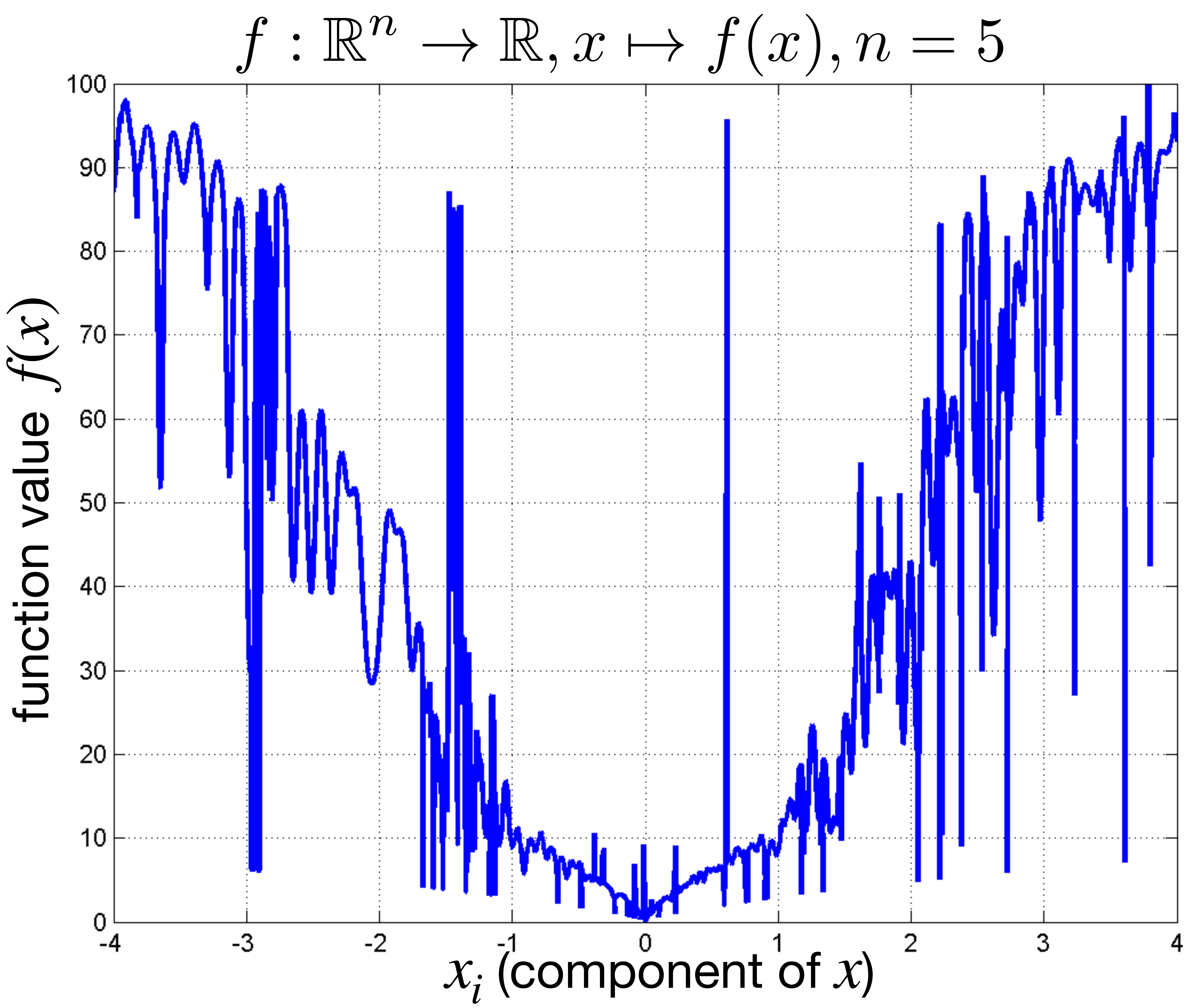
- in a black-box / direct search scenario

✓ no first order information (i.e. no gradient)

✓ unknown structure

- in theory: convergence to the global optimum
- in practice: find a good solution *iteratively* as quickly as possible

Section Through a 5-Dimensional Rugged Landscape



How can we modify gradient descent to solve this problem?

Flexible Muscle-Based Locomotion for Bipedal Creatures

SIGGRAPH ASIA 2013

Thomas Geijtenbeek
Michiel van de Panne
Frank van der Stappen

Flexible Muscle-Based Locomotion for Bipedal Creatures
T. Geijtenbeek, M van de Panne, F van der Stappen
https://youtu.be/pgaEE27nsQw

The Optimization/Search Algorithms

An Entirely Incomplete Landscape of Continuous Search Methods

Gradient-based (Taylor, local)

- Conjugate gradient methods [Fletcher & Reeves 1964]
- Quasi-Newton methods (BFGS) [Broyden et al 1970]

Derivative-free optimization (DFO)

- Trust-region methods (NEWUOA, BOBYQA) [Powell 2006, 2009]
- Simplex downhill [Nelder & Mead 1965]
- Pattern search [Hooke & Jeeves 1961, Audet & Dennis 2006]

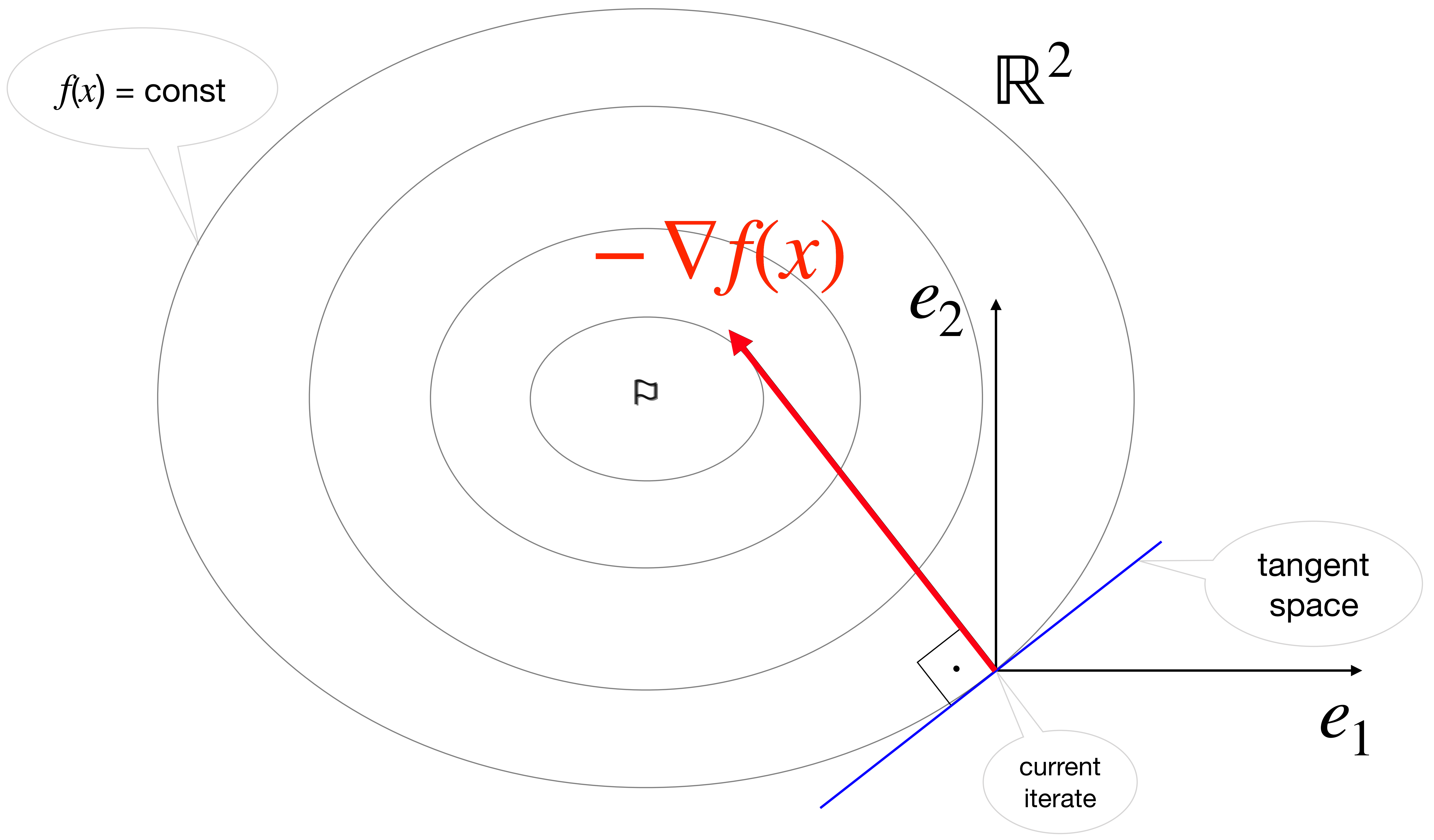
Stochastic (randomized) search methods

- Evolutionary algorithms (broader sense, continuous domain)
  - Differential Evolution [Storn & Price 1997]
  - Particle Swarm Optimization [Kennedy & Eberhart 1995]
  - **Evolution Strategies** [Rechenberg 1965, Hansen & Ostermeier 2001]
- Simulated annealing [Kirkpatrick et al 1983]
- Simultaneous perturbation stochastic approximation (SPSA) [Spall 2000]

From Gradient Descent to Evolution Strategies

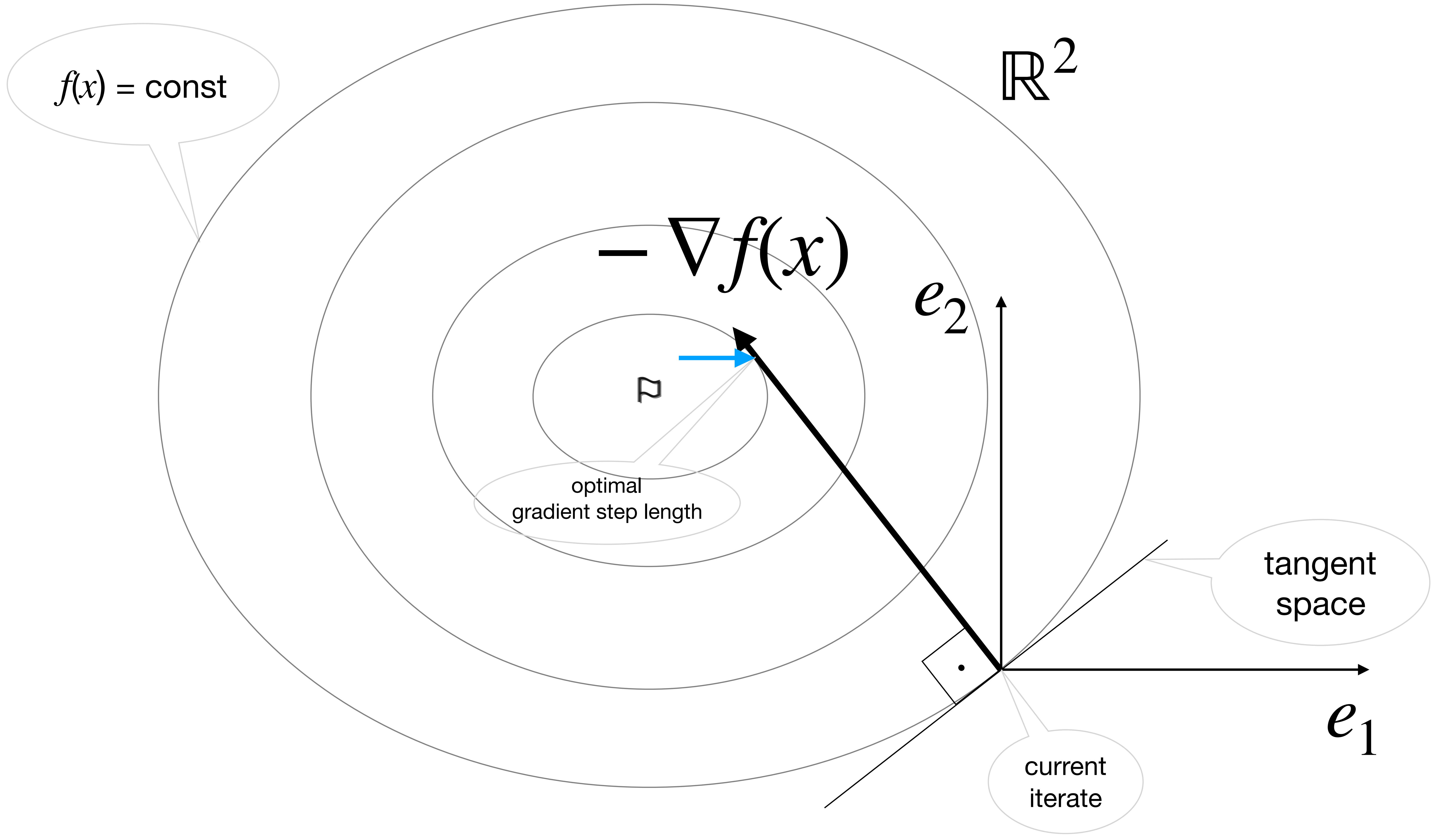
Basic Approach: Gradient Descent

The *gradient* is the local direction of the maximal f increase



Basic Approach: Gradient Descent

The *gradient* is the local direction of the maximal f increase



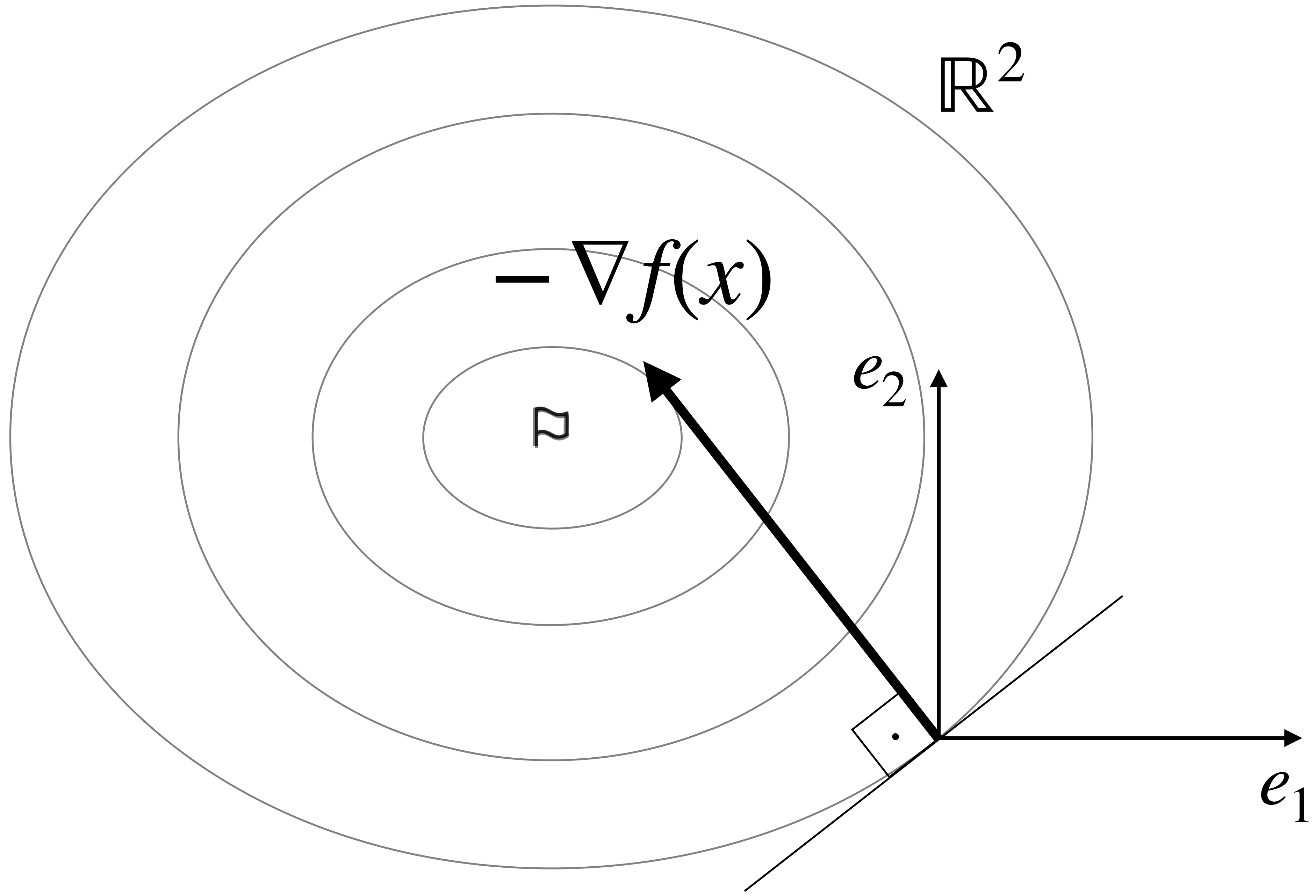
Basic Approach: Gradient Descent

The *gradient* is the local direction of the maximal f increase

$$\nabla f(x) = -\sum_{i=1}^{n} w_i e_i \qquad -w_i = lim_{\delta\to 0}\frac{f(x+\delta e_i)-f(x)}{\delta}$$

$$x \leftarrow x - \sigma \nabla f(x)$$

$$= x + \sigma \sum_{i=1}^{n} w_i e_i$$



Basic Approach: Gradient Descent

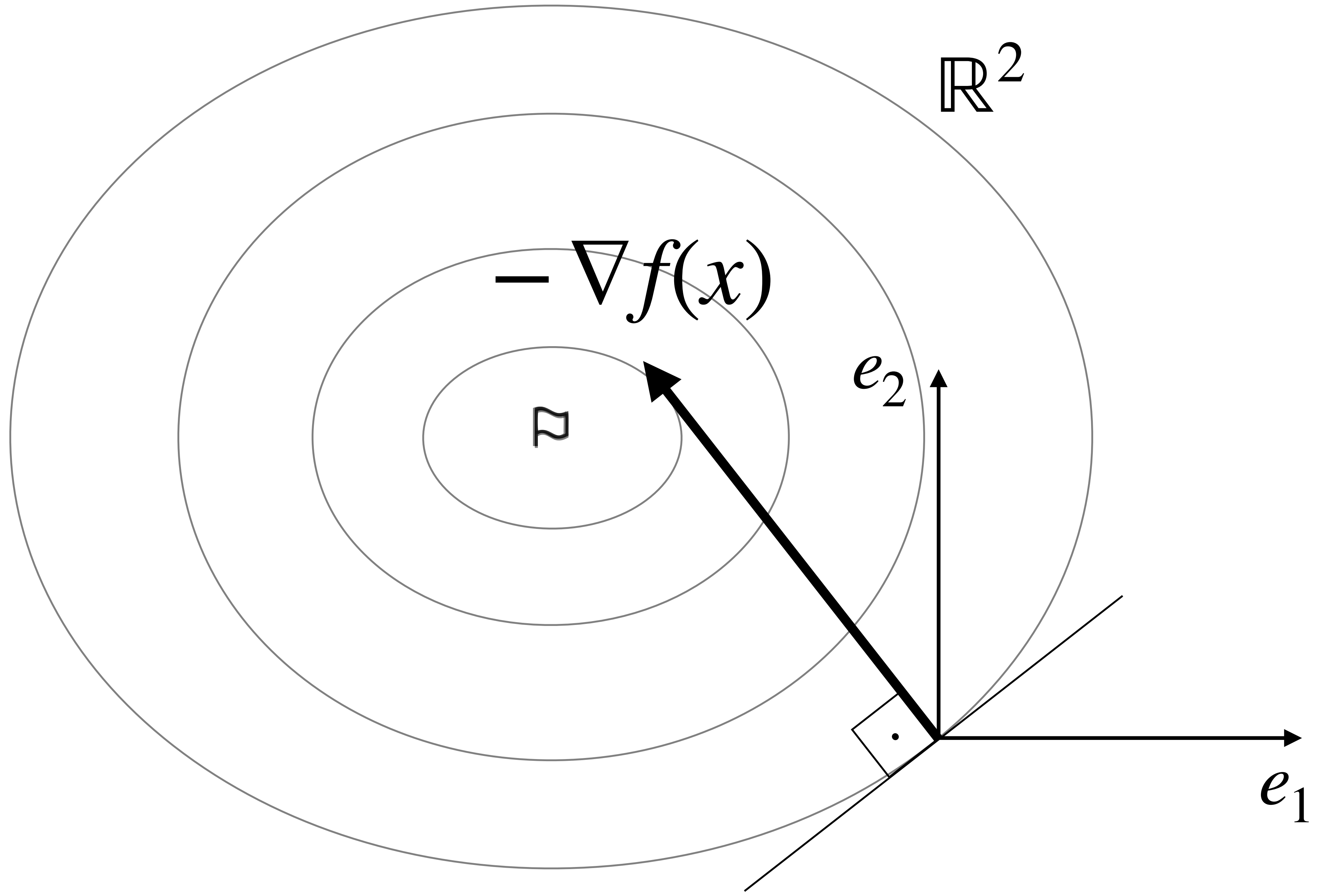
The *gradient* is the local direction of the maximal f increase

partial derivative $\frac{\partial f}{\partial x_i}(x)$

$$\nabla f(x) = -\sum_{i=1}^{n} w_i e_i \qquad -w_i = lim_{\delta\to 0}\frac{f(x+\delta e_i) - f(x)}{\delta}$$

$$x \leftarrow x - \sigma \nabla f(x)$$

$$= x + \sigma \sum_{i=1}^{n} w_i e_i$$



Basic Approach: Gradient Descent

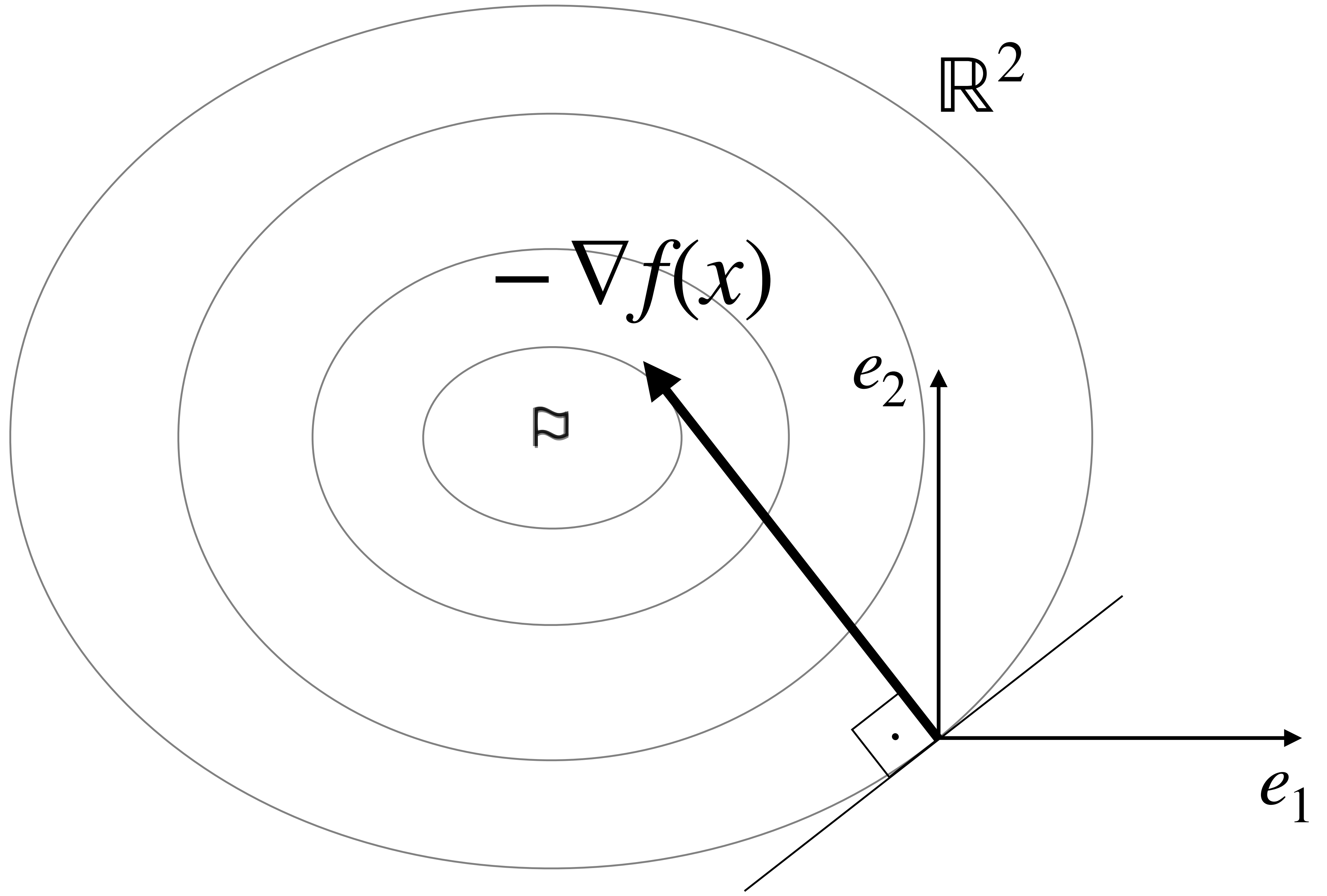
The *gradient* is the local direction of the maximal f increase

small test step

$$\nabla f(x) \approx -\sum_{i=1}^{n} w_i e_i \qquad -w_i = \cancel{\lim_{\delta\to 0}} \frac{f(x+\delta e_i) - f(x)}{\delta}$$

$$x \leftarrow x - \sigma \nabla f(x)$$

$$\approx x + \sigma \sum_{i=1}^{n} w_i e_i$$



Basic Approach: Gradient Descent

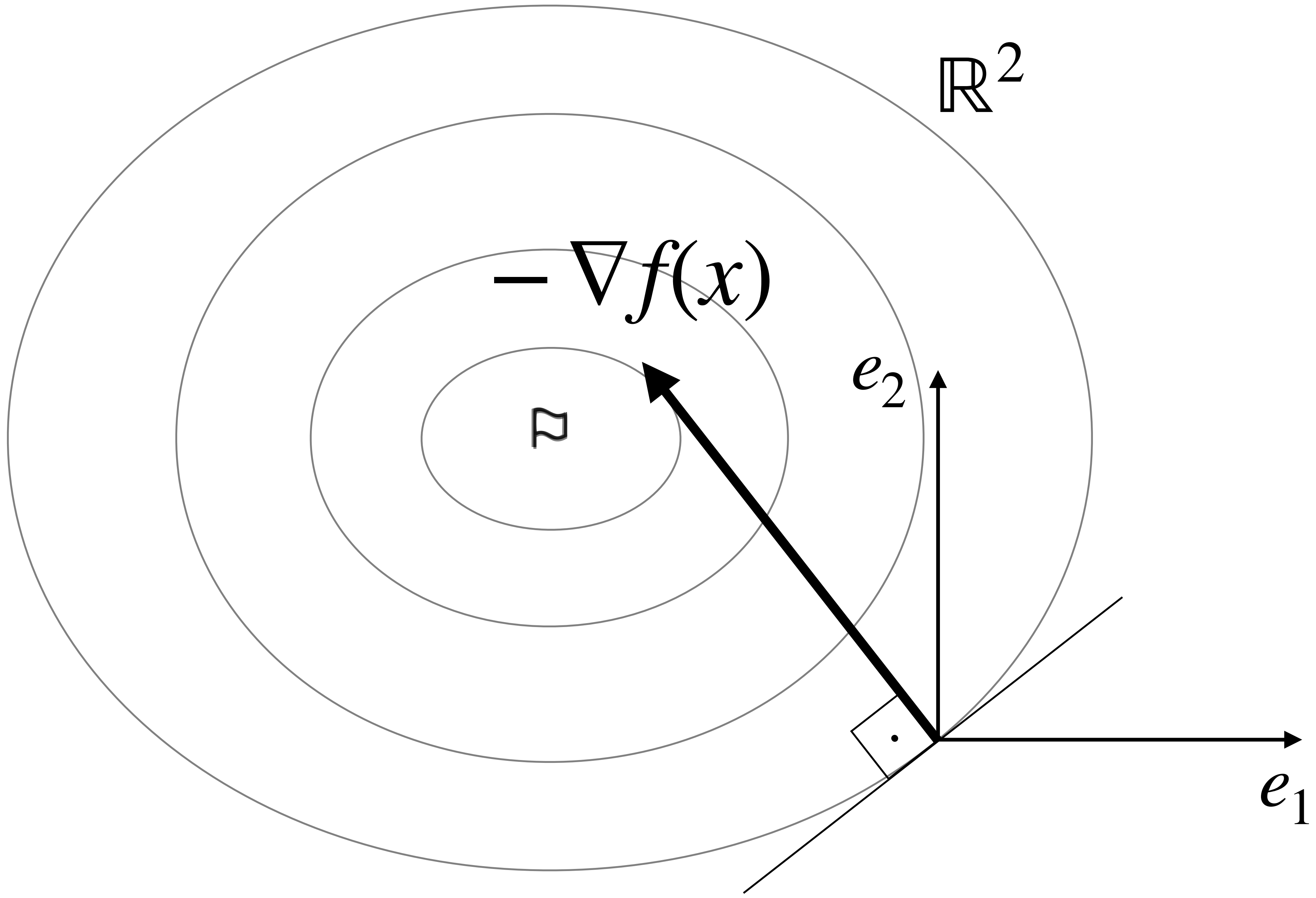
The *gradient* is the local direction of the maximal f increase

small test step

$$\nabla f(x) \approx -\sum_{i=1}^{n} w_i e_i \qquad -w_i = \cancel{\lim_{\delta\to 0}} \frac{f(x+\delta e_i) - f(x-\delta e_i)}{2\delta}$$

$$x \leftarrow x - \sigma \nabla f(x)$$

$$\approx x + \sigma \sum_{i=1}^{n} w_i e_i$$



Now we do three (small) changes

leading to an algorithm with very different behavior

Basic Approach: Gradient Descent

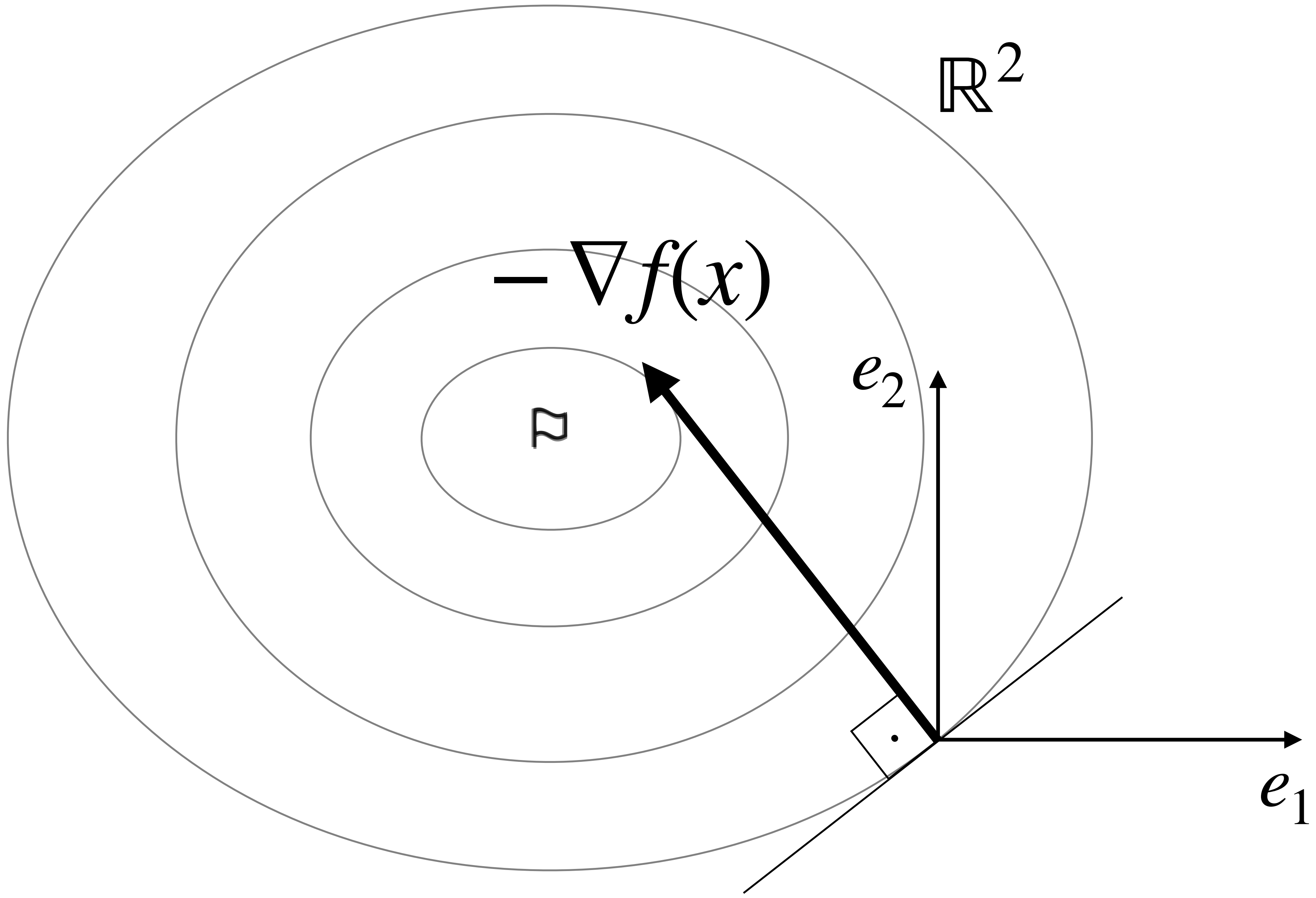
The *gradient* is the local direction of the maximal f increase

small test step

$$\nabla f(x) \approx -\sum_{i=1}^{n} w_i e_i \qquad -w_i = \cancel{\lim_{\delta\to 0}} \frac{f(x+\delta e_i) - f(x)}{\delta}$$

$$x \leftarrow \cancel{x - \sigma \nabla f(x)}$$

$$x + \sigma \sum_{i=1}^{n} w_i e_i$$



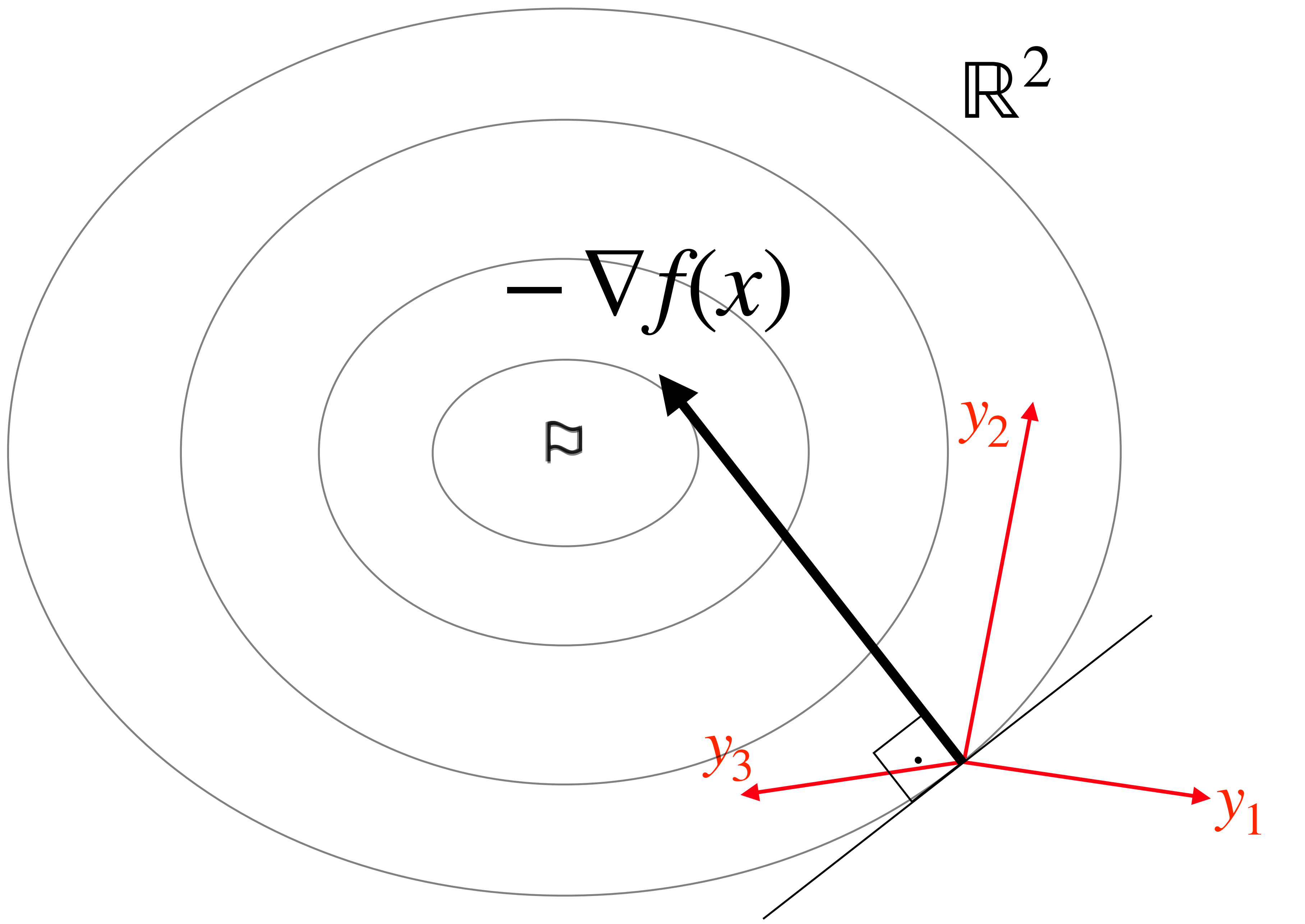
Basic Approach: Approximated Gradient Descent

We modify the gradient equation: (1) use y_i instead of e_i

$$\nabla f(x) \approx -\sum_{i=1}^{m} w_i y_i \qquad -w_i = \cancel{\lim_{\delta\to 0}} \frac{f(x+\delta y_i) - f(x)}{\delta}$$

$$x \leftarrow \cancel{x - \sigma \nabla f(x)}$$

$$x + \sigma \sum_{i=1}^{m} w_i y_i$$



Basic Approach: Approximated Gradient Descent

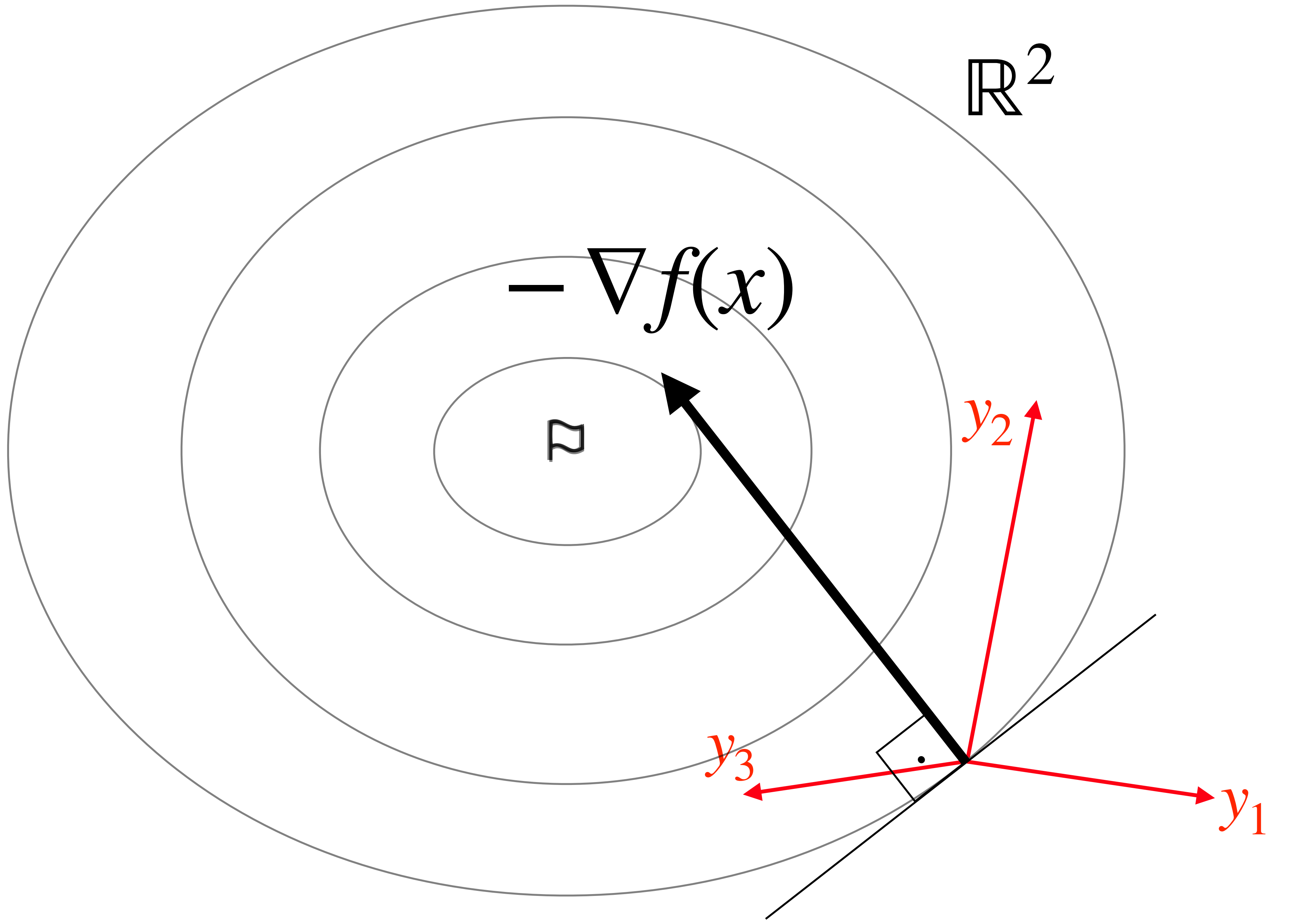
We modify the gradient equation: (1) use y_i instead of e_i

$$y_i \sim \mathcal{N}(0, I) \qquad -w_i = \cancel{\lim_{\delta \to 0}} \frac{f(x + \delta y_i) - f(x)}{\delta}$$

will become the *population size*

$$x \leftarrow \cancel{x - \sigma \nabla f(x)}$$

$$x + \sigma \sum_{i=1}^{m} w_i y_i$$



Basic Approach: Approximated Gradient Descent

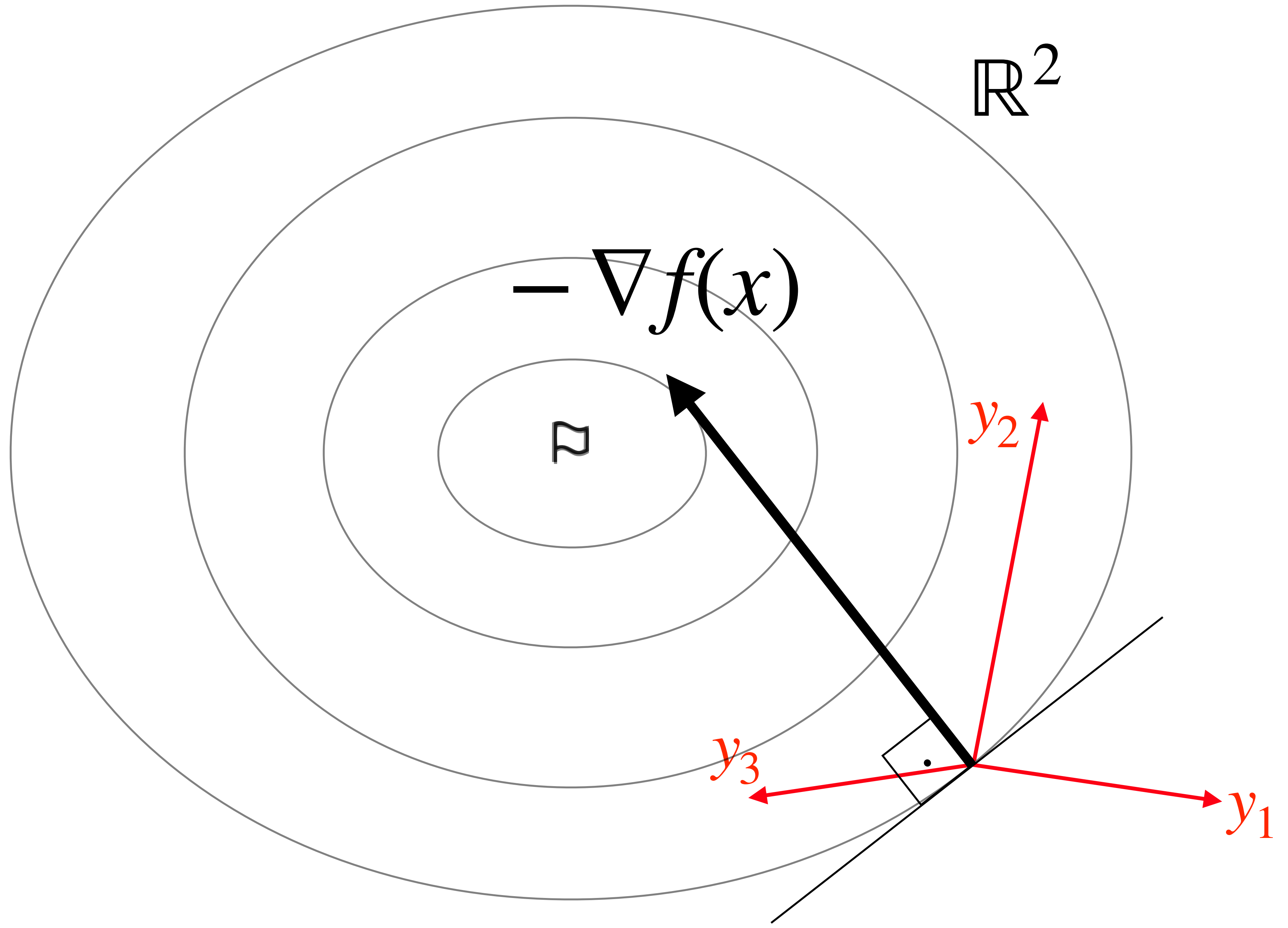
We modify the gradient equation: (2) make large test steps

~~small~~ test step ($\delta \approx \sigma$)

$$y_i \sim \mathcal{N}(0, I) \qquad -w_i = \cancel{\lim_{\delta \to 0}} \frac{f(x + \delta y_i) - f(x)}{\delta}$$

$$x \leftarrow \cancel{x - \sigma \nabla f(x)}$$

$$x + \sigma \sum_{i=1}^{m} w_i y_i$$



Evolutionary Gradient Search (EGS) [Salmon 1998, Arnold & Salomon 2007]

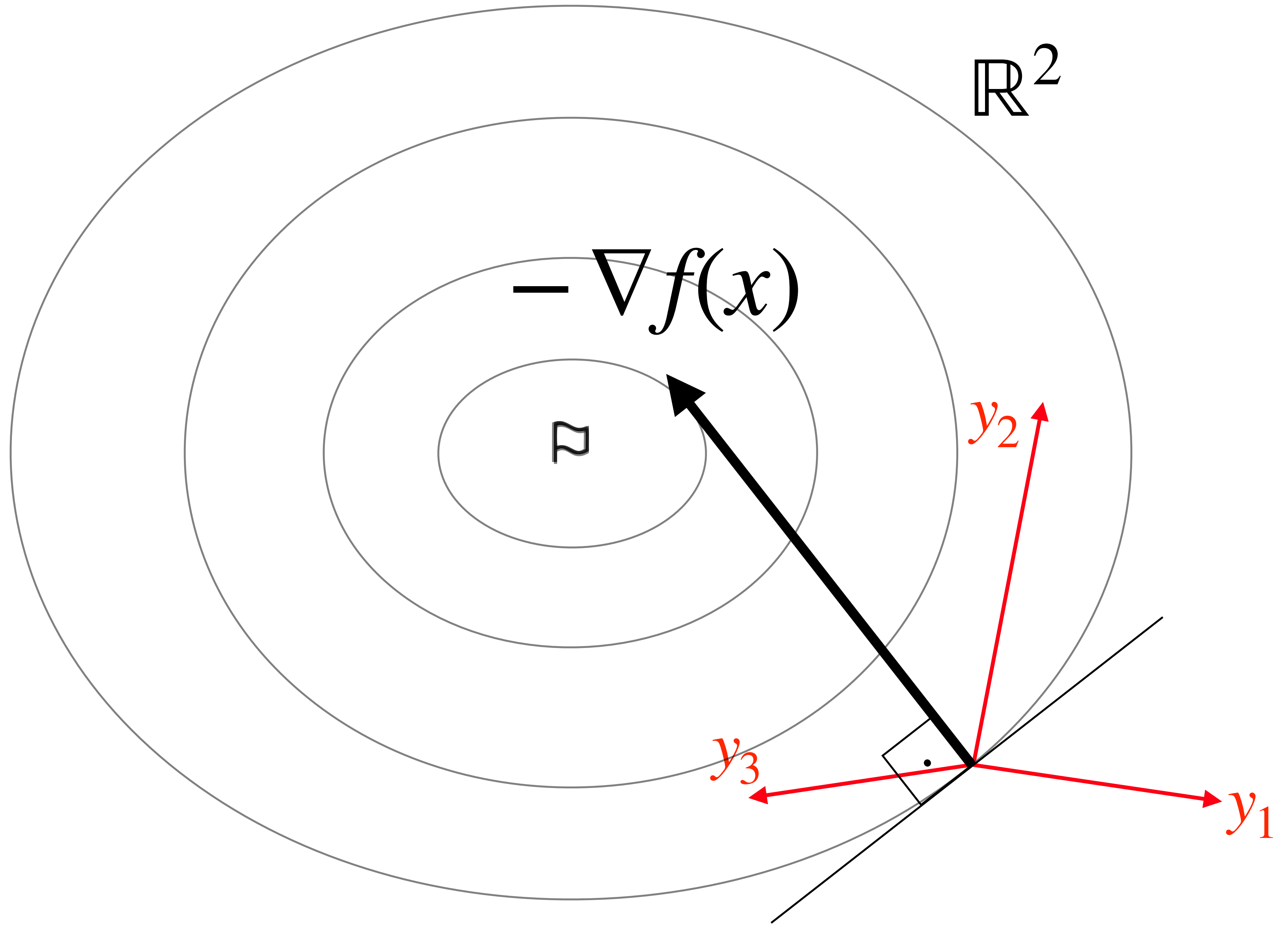
Rank-Based Approximated Gradient Descent

We modify the gradient equation: (3) use ranks instead of f-values

$$y_i \sim \mathcal{N}(0, I) \qquad -w_i \propto \overbrace{\textbf{rank}_i(f(x + \delta y_i))}^{\in\{1,\dots,m\}} - m/2$$

$$x \leftarrow \cancel{x - \sigma \nabla f(x)}$$

$$x + \sigma \sum_{w_i > 0} w_i y_i$$



Evolution Strategy (ES) [Rechenberg 1973, Schwefel 1981, Rudolph 1997, Hansen & Ostermeier 2001]

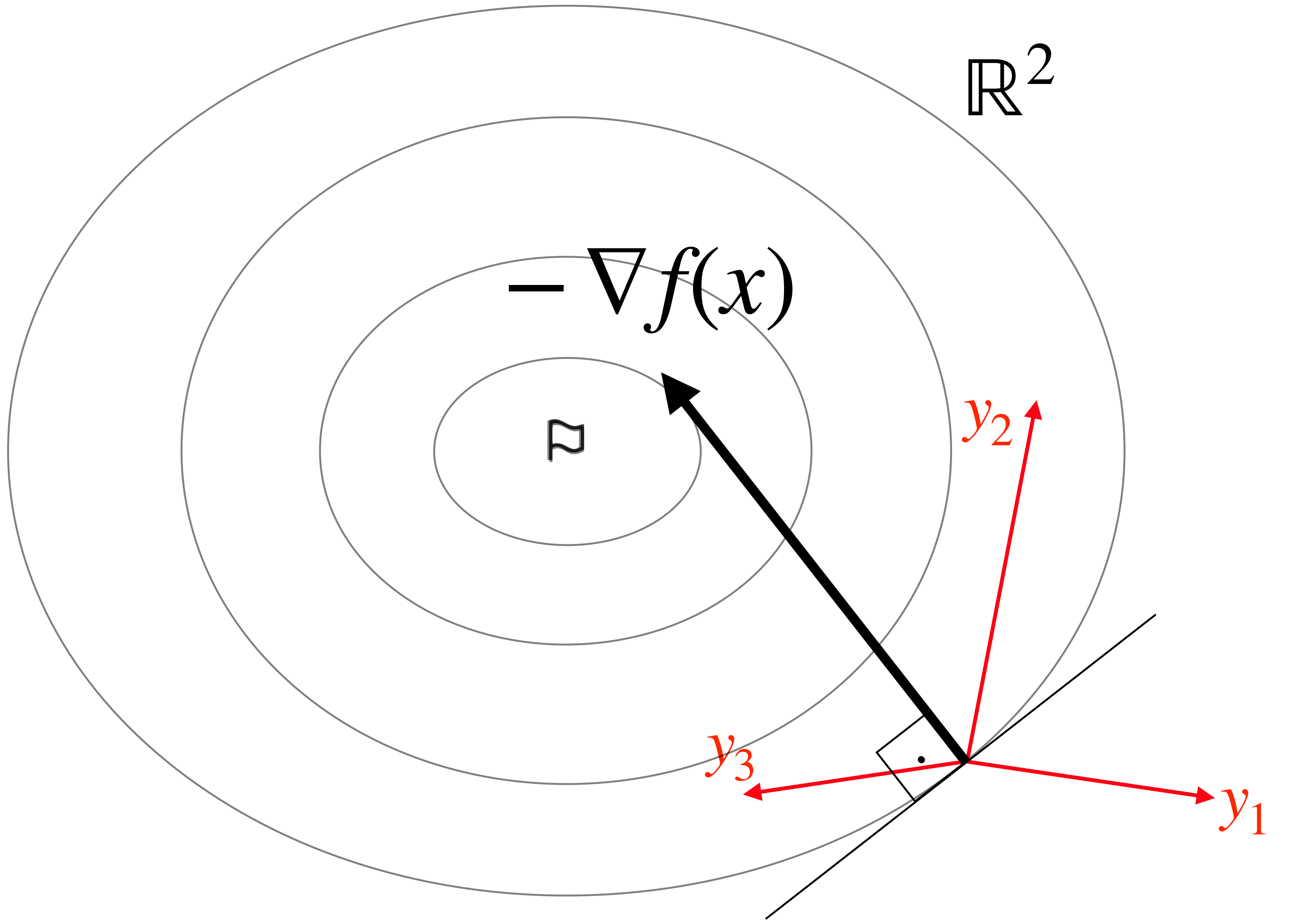
Rank-Based Approximated Gradient Descent = Evolution Strategy

We modify the gradient equation: (3) use ranks instead of f-values

$$y_i \sim \mathcal{N}(0, I) \qquad -w_i = \frac{\overbrace{\ln\big(\mathbf{rank}_i(f(x+\delta y_i))\big) - \ln\frac{m+1}{2}}^{\sum_{w_i>0} w_i \approx 1}}{m/2}$$

$$x \leftarrow \cancel{x - \sigma \nabla f(x)}$$

$$x + \sigma \sum_{w_i>0} w_i y_i$$



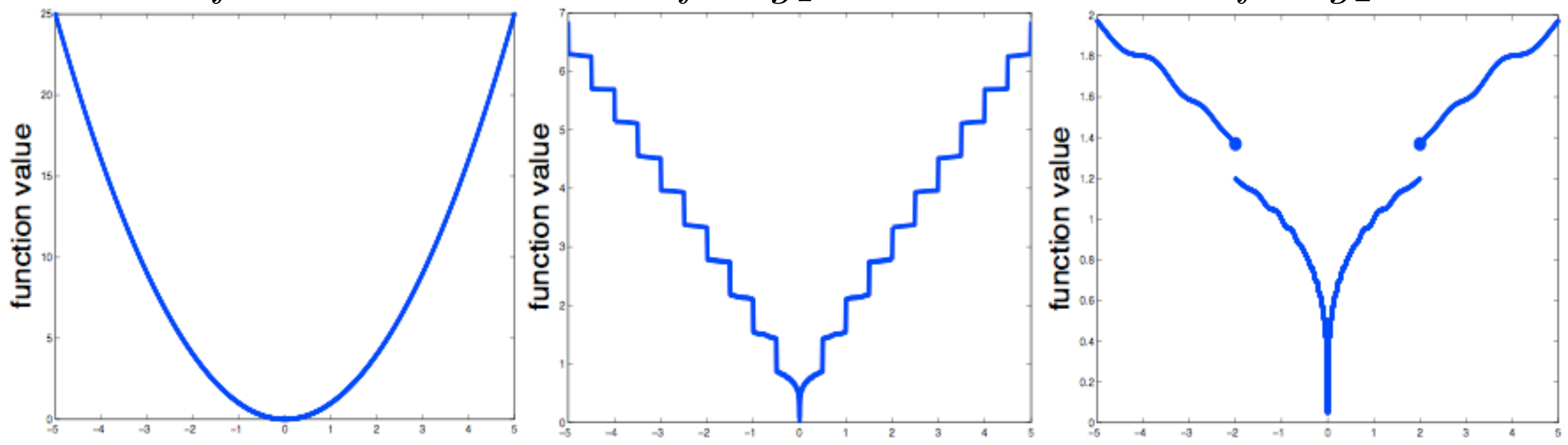
Evolution Strategy (ES) [Rechenberg 1973, Schwefel 1981, Rudolph 1997, Hansen & Ostermeier 2001]

Using Rank-Based Weights

- introduces robustness to (erroneously) f-value differences
- introduces invariance to
  - scaling of (the gradient of) f
  - strictly monotonous f-transformations

Invariance from Rank-Based Weights

$f = h$ $\qquad$ $f = g_1 \circ h$ $\qquad$ $f = g_2 \circ h$



Three functions belonging to the same equivalence class

A *rank-based search algorithm* is invariant under the transformation with any order preserving (strictly increasing) g.

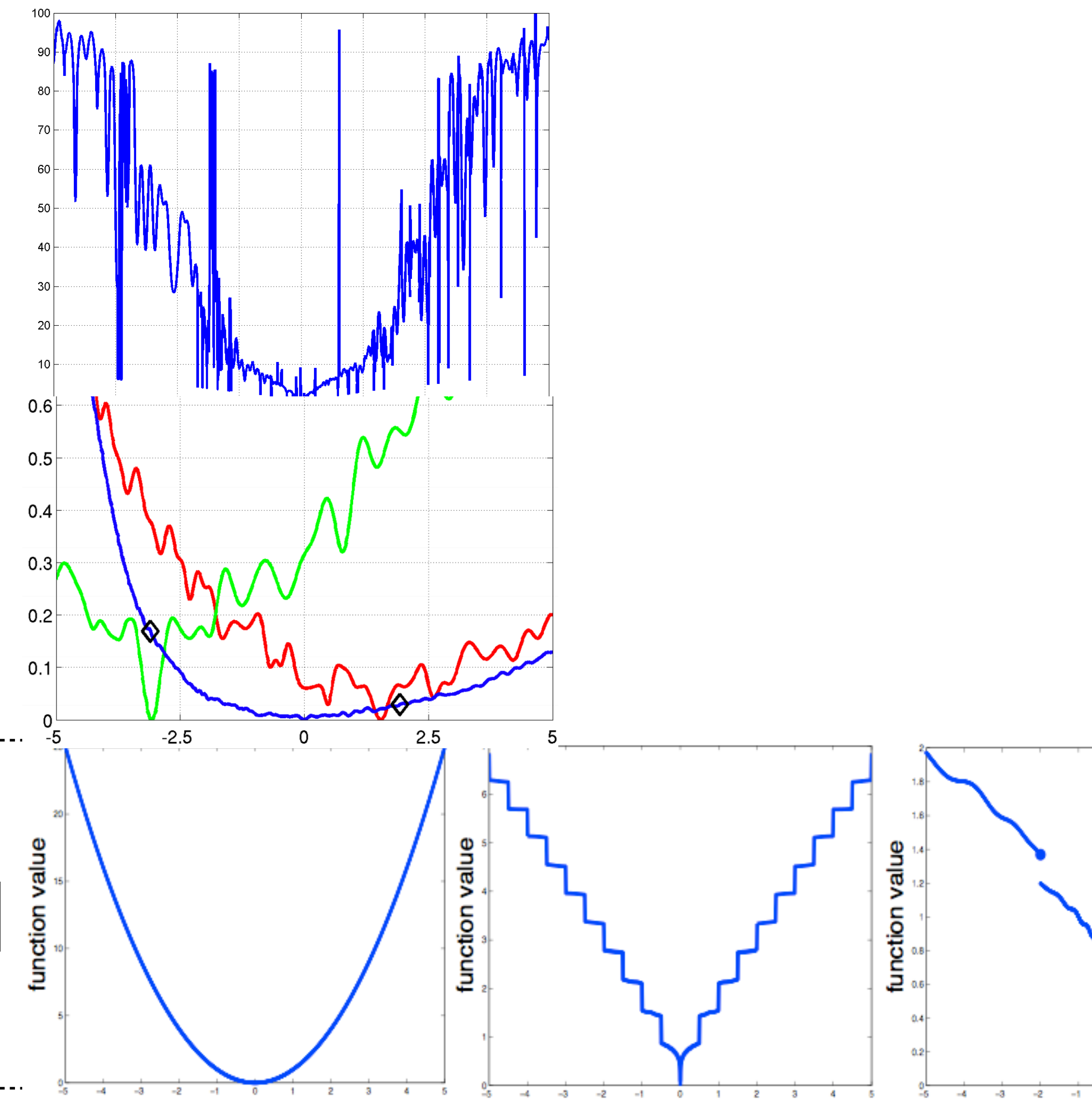
Invariances make

- observations meaningful
- algorithms predictable and/or ”robust”

as a rigorous notion of generalization

From Gradient Descent to Evolution Strategies

	Gradient Descent	**Evolution Strategy**
Test Steps:	unit vectors	(symmetric) random vectors
	dimension n or $2n$	any number > 1
	very small	(very) **large**
Weights:	partial derivatives (estimated)	**fixed** rank-based
Realized Step Length:	line search	step-size control (non-trivial)



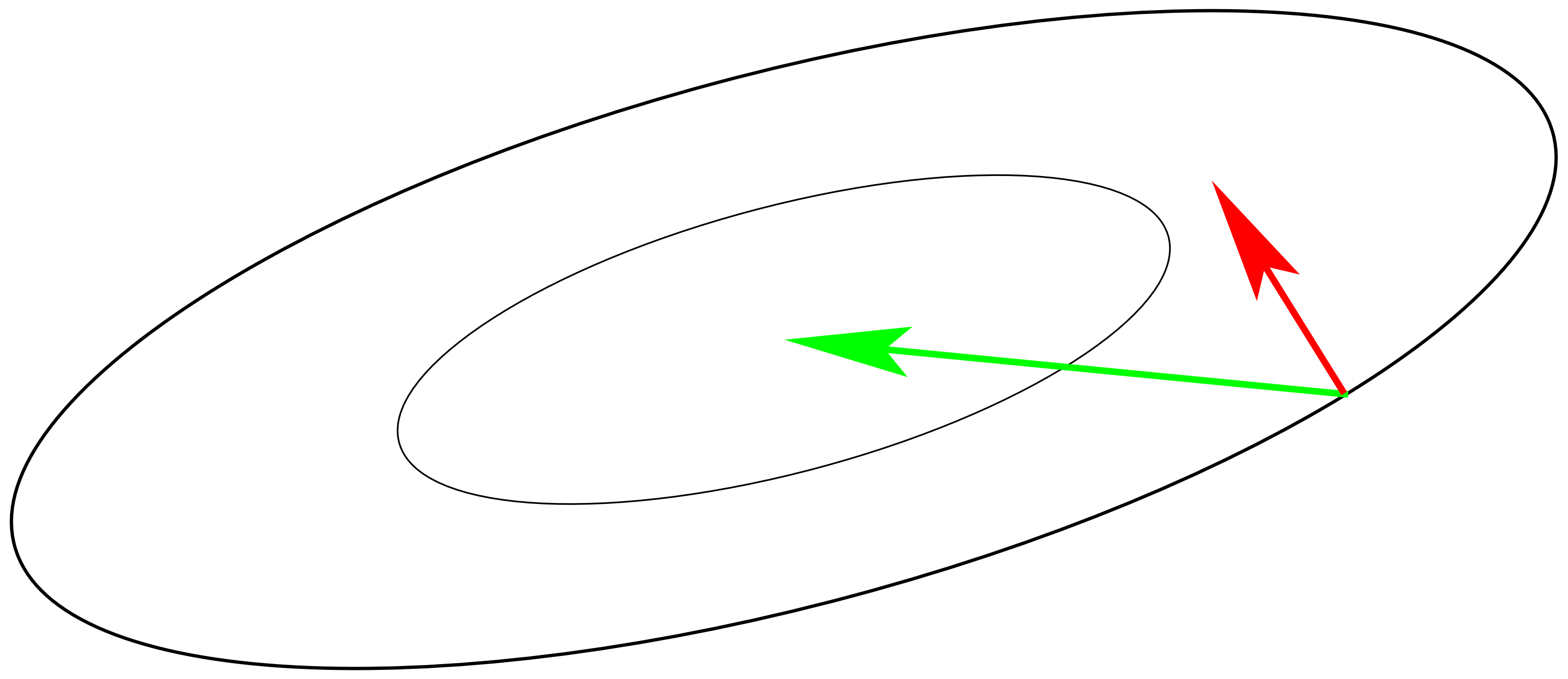
Ill-Conditioned Problems

Curvature of level sets

Consider the convex-quadratic function

$$f(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^*)^T\boldsymbol{H}(\boldsymbol{x}-\boldsymbol{x}^*) = \frac{1}{2}\sum_i h_{i,i}\,(x_i-x_i^*)^2 + \frac{1}{2}\sum_{i\neq j} h_{i,j}\,(x_i-x_i^*)(x_j-x_j^*)$$

$\boldsymbol{H}$ is Hessian matrix of f and symmetric positive definite



gradient direction $-f'(\boldsymbol{x})^{\mathrm{T}}$

Newton direction $-\boldsymbol{H}^{-1}f'(\boldsymbol{x})^{\mathrm{T}}$

Ill-conditioning means squeezed level sets (high curvature). Condition number equals nine here. Condition numbers up to 10^{10} are not unusual in real world problems.

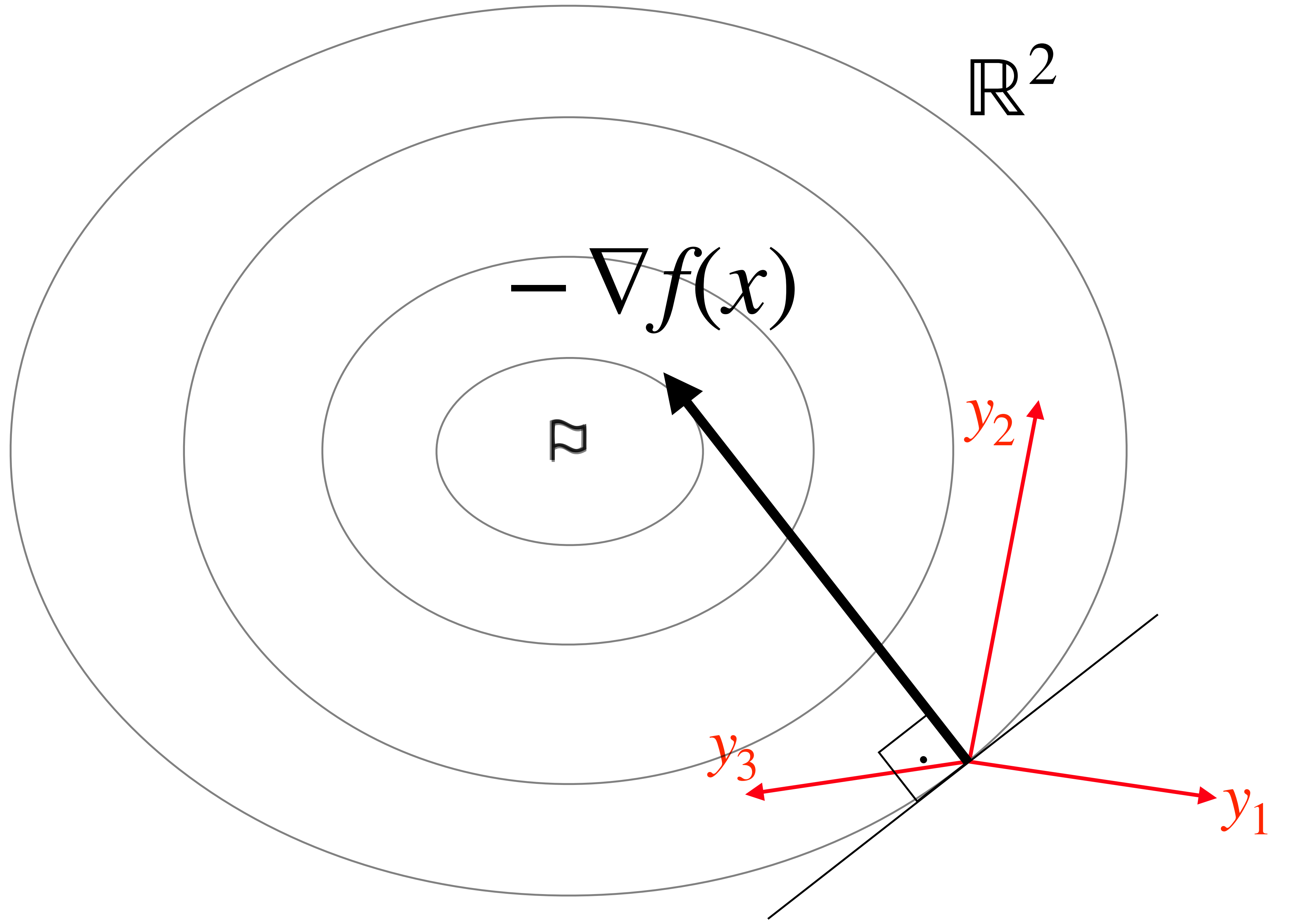
If $\boldsymbol{H} \approx \mathbf{I}$ (small condition number of $\boldsymbol{H}$) first order information (e.g. the gradient) is sufficient. Otherwise second order information (estimation of $\boldsymbol{H}^{-1}$) is necessary.

Rank-Based Approximated Gradient Descent = Evolution Strategy

$$y_i \sim \mathcal{N}(0, I) \qquad -w_i = \frac{\ln\left(\mathbf{rank}_i(f(x + \delta y_i))\right) - \ln\frac{m+1}{2}}{m/2}$$

$$x \leftarrow \cancel{x - \sigma \nabla f(x)}$$

$$x + \sigma \sum_{w_i > 0} w_i y_i$$



Evolution Strategy (ES) [Rechenberg 1973, Schwefel 1981, Rudolph 1997, Hansen & Ostermeier 2001]

Rank-Based Approximated Gradient Descent With Variable Metric

We estimate the shape of the level sets (without using f-values)

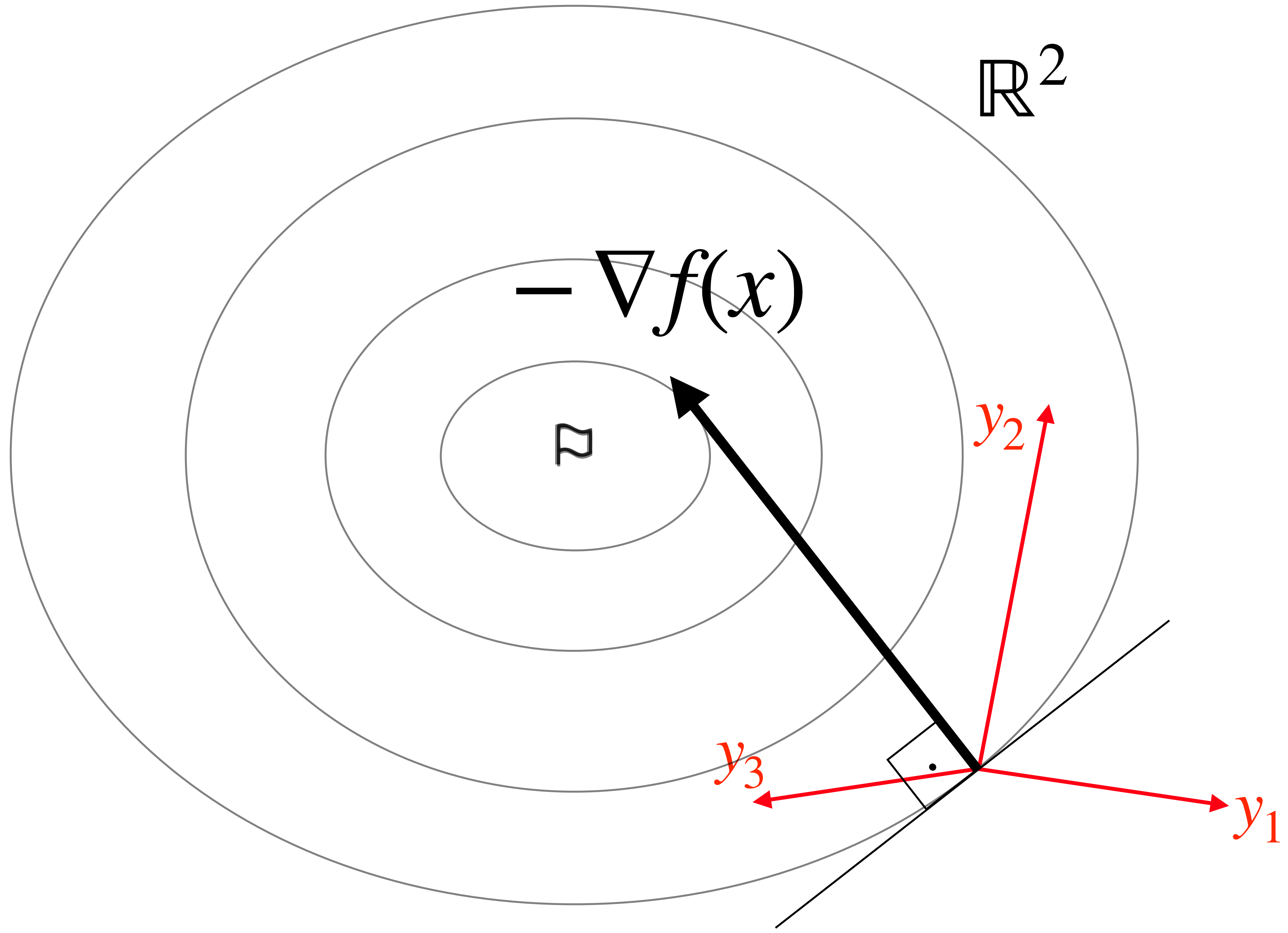
variable metric, updated to estimate H^{-1} up to a factor

$$y_i \sim \mathcal{N}(0, C)$$

$$-w_i = \frac{\ln\left(\mathbf{rank}_i(f(x + \delta y_i))\right) - \ln\frac{m+1}{2}}{m/2}$$

$$x \leftarrow \cancel{x - \sigma \nabla f(x)}$$

$$x + \sigma \sum_{w_i > 0} w_i y_i$$



Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [Hansen & Ostermeier 2001, Hansen et al 2003]

CMA-ES

Let $x \in \mathbb{R}^n$, $\sigma > 0$, $C = \mathbf{I}_n$, $y_0 = \mathbf{0}$

population size

$$x_k \sim \mathcal{N}(x, \sigma^2 C) = x + \sigma\,\mathcal{N}(0, C) \in \mathbb{R}^n, \quad k = 1 \dots \lambda$$

$$y_k = \frac{x_{\text{permute}_\lambda(k)} - x}{\sigma}$$ **sorted by** f $\quad y_k \sim \mathcal{N}(0, C)$

$$x \leftarrow x + c_m\,\sigma \sum_{w_k > 0,\, k \neq 0} w_k y_k,$$

$$c_m \approx \Sigma_{k=1}^{\mu} w_k \approx 1,\ \mu \approx \lambda/2$$

$$y_0 \leftarrow (1 - c_c)\,y_0 + \sqrt{c_c(2 - c_c)\mu_w} \sum_{k=1}^{\mu} w_k y_k,$$

$$c_c \approx \sqrt{c_\mu}, \quad \mu_w = \frac{(\Sigma_{i=1}^{\mu} w_k)^2}{\Sigma_{i=1}^{\mu} {w_k}^2}$$

$$C \leftarrow C + c_\mu \sum_{k=0}^{\lambda} w_k (y_k y_k^\top - C),$$

$$c_\mu \approx \mu_w/n^2,\ \Sigma_{k=0}^{\lambda} w_k \approx 0$$

$$\sigma \leftarrow \sigma \times \exp(\dots)$$

CMA-ES

Let $x \in \mathbb{R}^n$, $\sigma > 0$, $C = \mathbf{I}_n$, $y_0 = \mathbf{0}$

population size

$$x_k \sim \mathcal{N}(x, \sigma^2 C) = x + \sigma\,\mathcal{N}(0, C) \in \mathbb{R}^n, \quad k = 1 \ldots \lambda$$

$$y_k = \frac{x_{\mathrm{permute}_\lambda(k)} - x}{\sigma}$$ **sorted by** f $\quad y_k \sim \mathcal{N}(0, C)$

$$x \leftarrow x + c_m\,\sigma \sum_{w_k > 0,\, k \neq 0} w_k y_k, \qquad c_m \approx \Sigma_{k=1}^{\mu} w_k \approx 1,\ \mu \approx \lambda/2$$

$$y_0 \leftarrow (1 - c_c)\,y_0 + \sqrt{c_c(2 - c_c)\mu_w} \sum_{k=1}^{\mu} w_k y_k, \qquad c_c \approx \sqrt{c_\mu}, \quad \mu_w = \frac{(\Sigma_{i=1}^{\mu} w_k)^2}{\Sigma_{i=1}^{\mu} {w_k}^2}$$

$$C \leftarrow C + c_\mu \sum_{k=0}^{\lambda} w_k (y_k y_k^\top - C), \qquad c_\mu \approx \mu_w / n^2,\ \Sigma_{k=0}^{\lambda} w_k \approx 0$$

$$\sigma \leftarrow \sigma \times \exp(\ldots)$$

Summary

- There are many interesting applications for robust black-box optimization

- It takes three modifications to turn gradient descent into an evolution strategy
  - Test steps: replace unit vectors with a symmetrical *distribution* (of any number)
  - Test steps: replace small with *large* test steps (no limit to zero)
  - Replace f-value differences with *fixed weights* for linear combination of test steps

- We can reliably estimate the shape of the level sets (the inverse Hessian) in evolution strategies (CMA-ES) without using f-values pypi.org/project/cma/

cma 2.7.0

```
pip install cma
```

Latest version

Last released: Apr 25, 2019

CMA-ES, Covariance Matrix Adaptation Evolution Strategy for non-linear numerical optimization in Python

Navigation

- Project description
- Release history
- Download files

Project links

- Homepage

Statistics

GitHub statistics:

- Stars: 310
- Forks: 51
- Open issues/PRs: 25

View statistics for this project via Libraries.io, or by using Google BigQuery

Project description

A stochastic numerical optimization algorithm for difficult (non-convex, ill-conditioned, multi-modal, rugged, noisy) optimization problems in continuous search spaces, implemented in Python.

Typical domain of application are bound-constrained or unconstrained objective functions with:

- search space dimension between, say, 5 and (a few) 100,
- no gradients available,
- at least, say, 100 times dimension function evaluations needed to get satisfactory solutions,
- non-separable, ill-conditioned, or rugged/multi-modal landscapes.

The CMA-ES is quite reliable, however for small budgets (fewer function evaluations than, say, 100 times dimension) or in very small dimensions better (i.e. faster) methods are available.

The `pycma` module provides two independent implementations of the CMA-ES algorithm in the classes `cma.CMAEvolutionStrategy` and `cma.purecma.CMAES`.

Installation

There are several ways of installation:

- In the terminal command line type:

pypistats.org/packages/cma

cma

PyPI page
Home page
Author: Nikolaus Hansen
License: BSD
Summary: CMA-ES, Covariance Matrix Adaptation Evolution Strategy for non-linear numerical optimization in Python
Latest version: 2.7.0

Merci

Downloads last day: 69
Downloads last week: 2,560
Downloads last month: 20,572

Daily Download Quantity of cma package - Overall