Revisiting derivative-free optimization algorithms in the context of black-box adversarial attacks

Clément Royer (Université Paris Dauphine-PSL)

Séminaire Statistiques et Optimisation

Toulouse, 10 mars 2026

MILES project in Dauphine

What is MILES?

- Team-project within Dauphine.
- Focus: **Trustworthy AI**.
- Expertise: Machine learning, Game theory, Optimization,...

We have an MCF opening for 2026!

MILES project in Dauphine

What is MILES?

- Team-project within Dauphine.
- Focus: **Trustworthy AI**.
- Expertise: Machine learning, Game theory, Optimization,...

We have an MCF opening for 2026!

One of our strengths: Adversarial attacks

Perturb data to fool models

- Computing attacks (Meunier et al '19, Timoz et al '26).
- Game theoretical perspective (Meunier et al '21).
- Robustify models via randomization (Pinot et al. '19 '20 '22, Gnecco Heredia et al '24).

Where this talk fits in

COMPROMIS project (PEPR Cybersécurité)

- Thesis of Bastien Cavarretta.
- Goal: Develop structured adversarial attacks in a black-box setting using derivative-free optimization.

Where this talk fits in

COMPROMIS project (PEPR Cybersécurité)

- Thesis of Bastien Cavarretta.
- Goal: Develop structured adversarial attacks in a black-box setting using derivative-free optimization.

What we started with

- Zoo of black-box adversarial attacks.
- Derivative-free methods somewhat disregarded because of scalability issues.

Where this talk fits in

COMPROMIS project (PEPR Cybersécurité)

- Thesis of Bastien Cavarretta.
- Goal: Develop structured adversarial attacks in a black-box setting using derivative-free optimization.

What we started with

- Zoo of black-box adversarial attacks.
- Derivative-free methods somewhat disregarded because of scalability issues.

What we have now

- An algorithmic framework with enhanced scalability (theory).
- Preliminary numerical validation.

Roadmap

1. Adversarial attacks
2. Black-box adversarial attacks
3. Adversarial attacks via direct search

Roadmap

1. Adversarial attacks
2. Black-box adversarial attacks
3. Adversarial attacks via direct search

Classification setup

Basic classification paradigm

- Data points $\{\boldsymbol{x}_i, y_i\}_{i=1,\ldots,N}$ with
  - $\boldsymbol{x}_i$ input (e.g. image) viewed as a vector in $\mathcal{X} \subset \mathbb{R}^n$.
  - $y_i \in \{1, \ldots, C\}$ discrete (class) label.
- Goal: Predict y_i given $\boldsymbol{x}_i$ through a model h.
- Training: Compute best h within a model class $\mathcal{H}$.

Classification setup

Basic classification paradigm

- Data points $\{\boldsymbol{x}_i, y_i\}_{i=1,\dots,N}$ with
  - $\boldsymbol{x}_i$ input (e.g. image) viewed as a vector in $\mathcal{X} \subset \mathbb{R}^n$.
  - $y_i \in \{1, \dots, C\}$ discrete (class) label.
- Goal: Predict y_i given $\boldsymbol{x}_i$ through a model h.
- Training: Compute best h within a model class $\mathcal{H}$.

Empirical risk minimization

$$\underset{h \in \mathcal{H}}{\text{minimize}} \ \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(h(\boldsymbol{x}_i), y_i).$$

- Typical model: Logit function $h(\boldsymbol{x}) \in \mathbb{R}^C$ output of a neural network.
- Typical loss: Cross entropy $\mathcal{L}(h(\boldsymbol{x}), y) = -\log\left(\frac{\exp(h(\boldsymbol{x})^y)}{\sum_{c=1}^{C} \exp(h(\boldsymbol{x})^c)}\right)$.

What could go wrong in classification?

Suppose

- Data $\{(\boldsymbol{x}_i, y_i)\}$ (e.g. ImageNet).
- Model h obtained by training on this data?

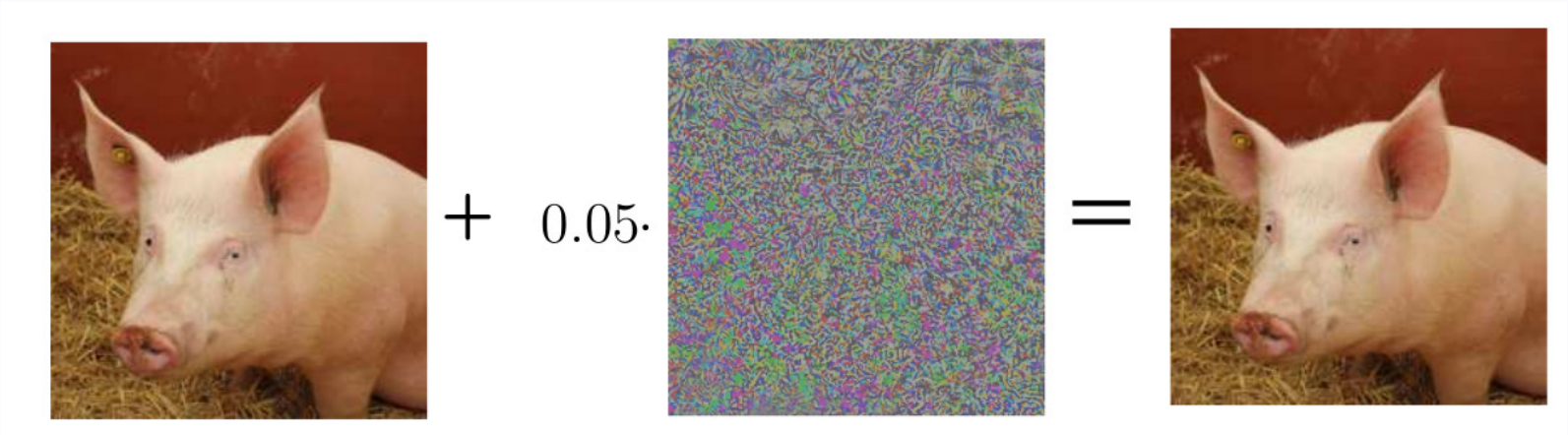
Can the model be fooled by small perturbations of data points?

What could go wrong in classification?

Suppose

- Data $\{(\boldsymbol{x}_i, y_i)\}$ (e.g. ImageNet).
- Model h obtained by training on this data?

Can the model be fooled by small perturbations of data points?



A hog misclassified as an airliner by a ResNet50 with high confidence
`https://adversarial-ml-tutorial.org/introduction/`

Adversarial examples for classification

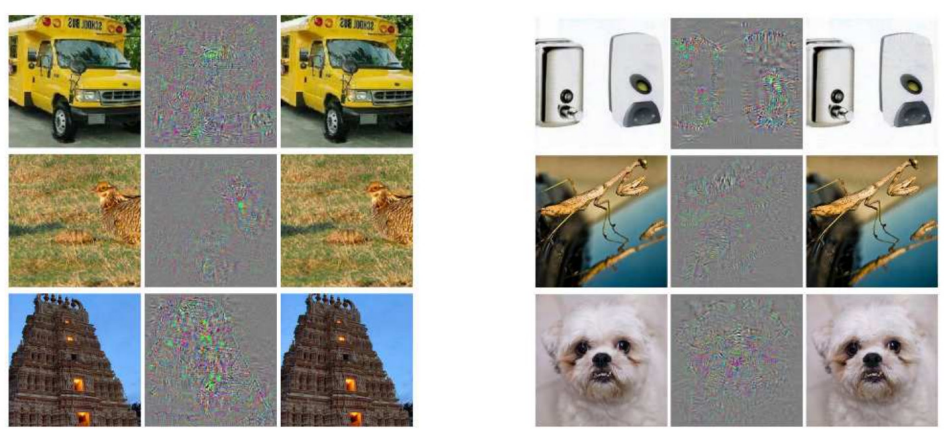
Adversarial examples/attacks

- Perturbation of data points that fool models (typically not humans).
- Observed in spam detection (Dalvi et al '04), speech recognition, language processing, etc.

Adversarial examples for classification

Adversarial examples/attacks

- Perturbation of data points that fool models (typically not humans).
- Observed in spam detection (Dalvi et al '04), speech recognition, language processing, etc.



Perturbed data points from ImageNet classified as Ostrich by AlexNet (Szegedy et al '14).

How are adversarial attacks computed?

How are adversarial attacks computed?

Adversarial attack from data point $(\boldsymbol{x}, y)$

$$\underset{\boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}, \epsilon) \cap \mathcal{X}}{\text{maximize}} \; \mathcal{L}(h(\boldsymbol{z}), \boldsymbol{y}).$$

- $\mathcal{B}(\boldsymbol{x}, \epsilon) = \{\boldsymbol{z} \mid \|\boldsymbol{z} - \boldsymbol{x}\| \leq \epsilon\}$, $\|\cdot\|$ norm on $\mathbb{R}^n$.
- h either un-trained (data poisoning) or **trained (evasion attack) model**.

$\rightarrow$ Focus for this talk: Evasion attacks.

How are adversarial attacks computed?

Adversarial attack from data point $(\boldsymbol{x}, y)$

$$\underset{\boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}, \epsilon) \cap \mathcal{X}}{\text{maximize}} \; \mathcal{L}(h(\boldsymbol{z}), \boldsymbol{y}).$$

- $\mathcal{B}(\boldsymbol{x}, \epsilon) = \{\boldsymbol{z} \mid \|\boldsymbol{z} - \boldsymbol{x}\| \leq \epsilon\}$, $\|\cdot\|$ norm on $\mathbb{R}^n$.
- h either un-trained (data poisoning) or **trained (evasion attack) model**.

$\rightarrow$ Focus for this talk: Evasion attacks.

Original attack (Szegedy et al '14)

$$\underset{\boldsymbol{z}}{\text{minimize}} \, \|\boldsymbol{z} - \boldsymbol{x}\|_2 \quad \text{s.t.} \quad h(\boldsymbol{z}) = y_{adv}, \boldsymbol{z} \in \mathcal{X}.$$

$\rightarrow$ Targeted attack (label y_{adv})
$\rightarrow$ Nonlinear optimization problem (solved with L-BFGS).

Gradient-based attacks

Idea Solve the attack problem inexactly using gradient-based methods.

Gradient-based attacks

Idea Solve the attack problem inexactly using gradient-based methods.

Fast Gradient Method (Goodfellow et al '15)

$$\underset{\boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}, \epsilon)}{\text{maximize}} \quad (\boldsymbol{z} - \boldsymbol{x})^{\mathrm{T}} \nabla_{\boldsymbol{x}} \mathcal{L}(h(\boldsymbol{x}), y).$$

- First-order approximation of $\mathcal{L}(h(\boldsymbol{z}), y) - \mathcal{L}(h(\boldsymbol{x}), y)$.
- Compute $\nabla_{\boldsymbol{x}} \mathcal{L}(h(\boldsymbol{x}), y)$ through automatic differentiation.
- Closed-form expressions when using $\|\cdot\|_2$ and $\|\cdot\|_\infty$.

Gradient-based attacks

Idea Solve the attack problem inexactly using gradient-based methods.

Fast Gradient Method (Goodfellow et al '15)

$$\underset{\boldsymbol{z}\in\mathcal{B}(\boldsymbol{x},\epsilon)}{\text{maximize}} \quad (\boldsymbol{z}-\boldsymbol{x})^{\mathrm{T}} \nabla_{\boldsymbol{x}} \mathcal{L}(h(\boldsymbol{x}), y).$$

- First-order approximation of $\mathcal{L}(h(\boldsymbol{z}), y) - \mathcal{L}(h(\boldsymbol{x}), y)$.
- Compute $\nabla_{\boldsymbol{x}} \mathcal{L}(h(\boldsymbol{x}), y)$ through automatic differentiation.
- Closed-form expressions when using $\|\cdot\|_2$ and $\|\cdot\|_\infty$.

Projected Gradient Descent/PGD (Kurakin et al '17, Madry et al '18)

$$\boldsymbol{z}_{t+1} = \Pi_{\mathcal{B}(\boldsymbol{x},\epsilon)} \left(\boldsymbol{z}_t + \alpha \nabla_x \mathcal{L}(h(\boldsymbol{z}_t), y)\right)$$

- $\Pi_{\mathcal{B}(\boldsymbol{x},\epsilon)}$: Projection.
- Multiple steps of gradient ascent for the problem.

More attacks, and defenses (sample!)

- **CW Attack** (Carlini & Wagner '17b): Modified version of (Szegedy et al '14) using changes of variables.
  $\rightarrow$ (Carlini & Wagner '17a): Circumvent detection of adversarial examples.

More attacks, and defenses (sample!)

- **CW Attack** (Carlini & Wagner '17b): Modified version of (Szegedy et al '14) using changes of variables.
  $\rightarrow$ (Carlini & Wagner '17a): Circumvent detection of adversarial examples.
- **Adversarial training** (Goodfellow et al '15, Madry et al '18): Train classifier with adversarial examples+training data.

More attacks, and defenses (sample!)

- **CW Attack** (Carlini & Wagner '17b): Modified version of (Szegedy et al '14) using changes of variables.
  $\rightarrow$ (Carlini & Wagner '17a): Circumvent detection of adversarial examples.
- **Adversarial training** (Goodfellow et al '15, Madry et al '18): Train classifier with adversarial examples+training data.
- **Circumventing existing defenses**
  - (Athalye et al '18) 7/9 defenses from ICLR 2018.
  - (Tramer et al '20) 13 defenses from ICML/ICLR/NeurIPS.
  - (Sitawarin et al '22) breaks (Raff et al '19).

More attacks, and defenses (sample!)

- **CW Attack** (Carlini & Wagner '17b): Modified version of (Szegedy et al '14) using changes of variables.
  $\rightarrow$ (Carlini & Wagner '17a): Circumvent detection of adversarial examples.
- **Adversarial training** (Goodfellow et al '15, Madry et al '18): Train classifier with adversarial examples+training data.
- **Circumventing existing defenses**
  - (Athalye et al '18) 7/9 defenses from ICLR 2018.
  - (Tramer et al '20) 13 defenses from ICML/ICLR/NeurIPS.
  - (Sitawarin et al '22) breaks (Raff et al '19).
- **Methodology for evaluating defenses**
  - (Carlini et al '19): Good practice.
  - **RobustBench** (Croce et al '21): Standardized benchmarks.

More attacks, and defenses (sample!)

- **CW Attack** (Carlini & Wagner '17b): Modified version of (Szegedy et al '14) using changes of variables.
  $\rightarrow$ (Carlini & Wagner '17a): Circumvent detection of adversarial examples.
- **Adversarial training** (Goodfellow et al '15, Madry et al '18): Train classifier with adversarial examples+training data.
- **Circumventing existing defenses**
  - (Athalye et al '18) 7/9 defenses from ICLR 2018.
  - (Tramer et al '20) 13 defenses from ICML/ICLR/NeurIPS.
  - (Sitawarin et al '22) breaks (Raff et al '19).
- **Methodology for evaluating defenses**
  - (Carlini et al '19): Good practice.
  - **RobustBench** (Croce et al '21): Standardized benchmarks.

- A never-ending game of cat and mouse...
- Randomized defenses (popular topic in MILES) not necessarily better (Lucas et al '23).

Summing up

Adversarial examples

- Built through an optimization problem.
- Attacks help build defenses/robust models.
- Classical tools are gradient-based.

Summing up

Adversarial examples

- Built through an optimization problem.
- Attacks help build defenses/robust models.
- Classical tools are gradient-based.

What if the model is not available?

Roadmap

1. Adversarial attacks
2. Black-box adversarial attacks
3. Adversarial attacks via direct search

Black-box adversarial attacks

Recall: Adversarial attack problem

$$\underset{\boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}, \epsilon)}{\text{maximize}} \; \mathcal{L}(h(\boldsymbol{z}), \boldsymbol{y})$$

Black-box adversarial attacks

Recall: Adversarial attack problem

$$\underset{\boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}, \epsilon)}{\text{maximize}} \; \mathcal{L}(h(\boldsymbol{z}), \boldsymbol{y})$$

Black-box setup

- h model (e.g. neural net) only available through queries.
- Different from previous *white-box* models.

First attempts at black-box attacks

Proxy-based attacks (Papernot et al. '17, Tu et al '18)

- Replace h with a proxy network $\tilde{h}$.
- Build white-box attacks for such a network!

$\rightarrow$ Efficient for computing many adversarial examples.

First attempts at black-box attacks

Proxy-based attacks (Papernot et al. '17, Tu et al '18)

- Replace h with a proxy network $\tilde{h}$.
- Build white-box attacks for such a network!

$\rightarrow$ Efficient for computing many adversarial examples.

ZOO/Zeroth-Order Optimization (Chen et al '17)

- Use finite-difference schemes to approximate gradient ascent.
- Efficient for computing a small number of attacks.

First attempts at black-box attacks

Proxy-based attacks (Papernot et al. '17, Tu et al '18)

- Replace h with a proxy network $\tilde{h}$.
- Build white-box attacks for such a network!

$\rightarrow$ Efficient for computing many adversarial examples.

ZOO/Zeroth-Order Optimization (Chen et al '17)

- Use finite-difference schemes to approximate gradient ascent.
- Efficient for computing a small number of attacks.

$\rightarrow$ A derivative-free optimization technique!

Derivative-free optimization

$$\text{maximize}_{\boldsymbol{z} \in \mathbb{R}^n} \quad f(\boldsymbol{z}).$$

Derivative-free algorithms

(Audet & Hare '25, Conn et al '09, Larson et al '21)

- **Derivatives unavailable for algorithmic use**.
- Function calls typically considered expensive.

Derivative-free optimization

$$\text{maximize}_{\boldsymbol{z} \in \mathbb{R}^n} \quad f(\boldsymbol{z}).$$

Derivative-free algorithms

(Audet & Hare '25, Conn et al '09, Larson et al '21)

- **Derivatives unavailable for algorithmic use**.
- Function calls typically considered expensive.

Main algorithmic families

- **Zeroth-order** Estimate gradient via finite differences.

Derivative-free optimization

$$\text{maximize}_{\boldsymbol{z} \in \mathbb{R}^n} \quad f(\boldsymbol{z}).$$

Derivative-free algorithms

(Audet & Hare '25, Conn et al '09, Larson et al '21)

- **Derivatives unavailable for algorithmic use**.
- Function calls typically considered expensive.

Main algorithmic families

- **Zeroth-order** Estimate gradient via finite differences.
- **Genetic algorithms** Population-based methods that sample.

Derivative-free optimization

$$\text{maximize}_{\boldsymbol{z} \in \mathbb{R}^n} \quad f(\boldsymbol{z}).$$

Derivative-free algorithms

(Audet & Hare '25, Conn et al '09, Larson et al '21)

- **Derivatives unavailable for algorithmic use**.
- Function calls typically considered expensive.

Main algorithmic families

- **Zeroth-order** Estimate gradient via finite differences.
- **Genetic algorithms** Population-based methods that sample.
- **Direct search** Explore the space through directions.

Derivative-free optimization

$$\text{maximize}_{\boldsymbol{z} \in \mathbb{R}^n} \quad f(\boldsymbol{z}).$$

Derivative-free algorithms

(Audet & Hare '25, Conn et al '09, Larson et al '21)

- **Derivatives unavailable for algorithmic use**.
- Function calls typically considered expensive.

Main algorithmic families

- **Zeroth-order** Estimate gradient via finite differences.
- **Genetic algorithms** Population-based methods that sample.
- **Direct search** Explore the space through directions.
- **Model based** Use surrogate for f to compute new points (similar to proxy but iterative).

A numerical comparison (from Ughi et al '22)

Goal ℓ_∞ attacks

$$\underset{\boldsymbol{z}\in\mathcal{B}_\infty(\boldsymbol{x},\epsilon_\infty)}{\text{maximize}} \quad \mathcal{L}(h(\boldsymbol{x}), y).$$

- $\mathcal{B}_\infty(\boldsymbol{x}, \epsilon_\infty) = \{\boldsymbol{z} \mid \max_i |z_i - x_i| \leq \epsilon\}$.
- h: ResNet50 with or without adversarial training (Engstrom et al '19).
- Dataset: CIFAR-10, $32 \times 32 \times 3$ images, 10 classes.
- 900 attacks (Assign all other labels to one of 100 images).

A numerical comparison (from Ughi et al '22)

Goal ℓ_∞ attacks

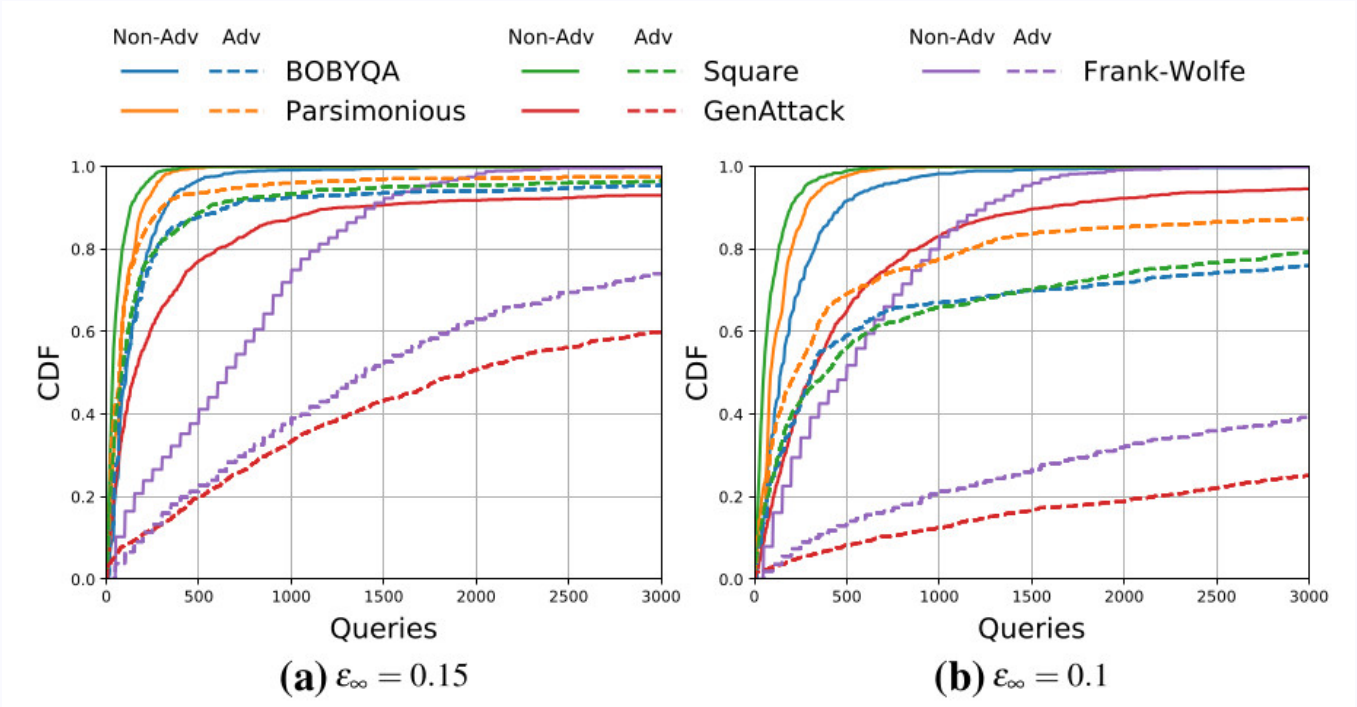
$$\underset{\boldsymbol{z} \in \mathcal{B}_\infty(\boldsymbol{x}, \epsilon_\infty)}{\text{maximize}} \quad \mathcal{L}(h(\boldsymbol{x}), y).$$

- $\mathcal{B}_\infty(\boldsymbol{x}, \epsilon_\infty) = \{\boldsymbol{z} \mid \max_i |z_i - x_i| \leq \epsilon\}$.
- h: ResNet50 with or without adversarial training (Engstrom et al '19).
- Dataset: CIFAR-10, $32 \times 32 \times 3$ images, 10 classes.
- 900 attacks (Assign all other labels to one of 100 images).

Compared methods

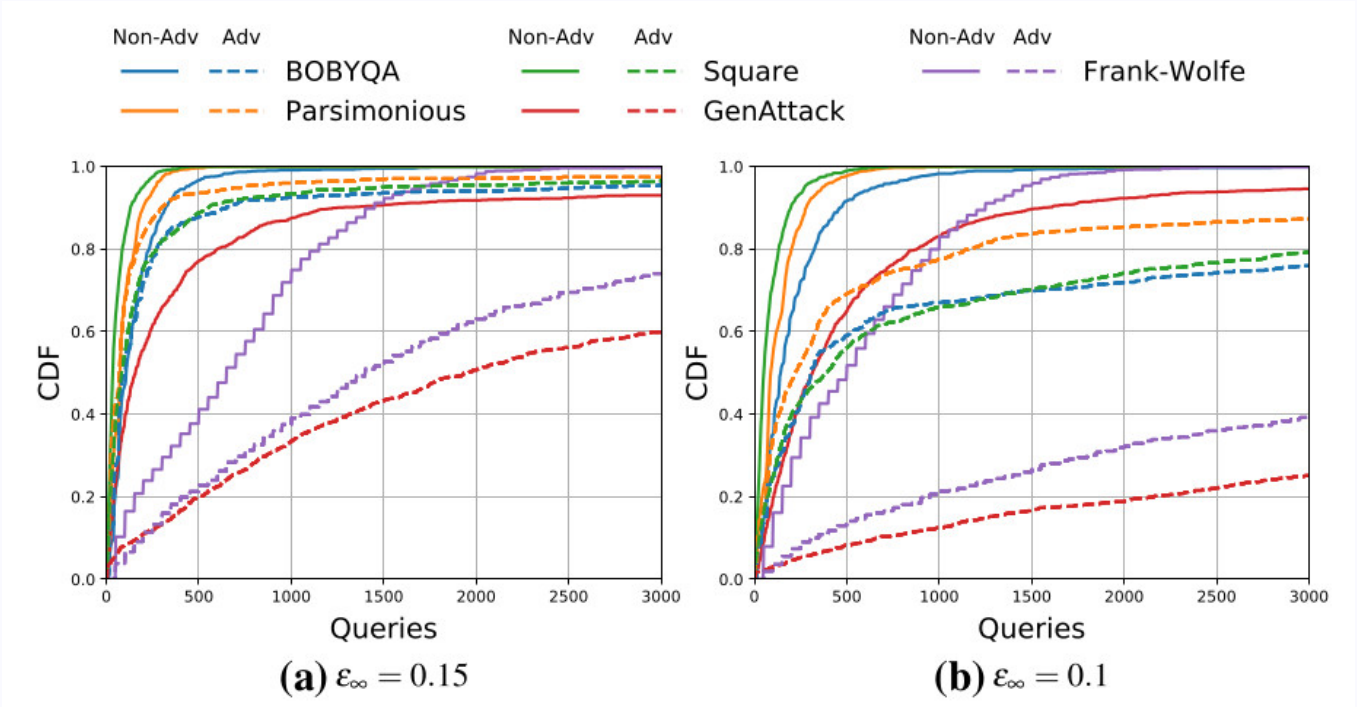
- *Frank-Wolfe* (Chen et al '20): Randomized zeroth-order method.
- *GenAttack* (Alzantot et al '19): Genetic algorithm.
- *Parsimonious* (Moon et al '19), *Square* (Andriushchenko et al '20): Direct search over combinatorial domain.
- BOBYQA (Powell '09) general-purpose DFO solver (in Fortran!)

A numerical comparison (from Ughi et al '22)



(a) $\varepsilon_\infty = 0.15$ **(b)** $\varepsilon_\infty = 0.1$

A numerical comparison (from Ughi et al '22)



(a) $\varepsilon_\infty = 0.15$ **(b)** $\varepsilon_\infty = 0.1$

Good performance for off-the-shelf BOBYQA?

Off-the-shelf solver and adversarial

BOBYQA solver (Original, Powell '09)

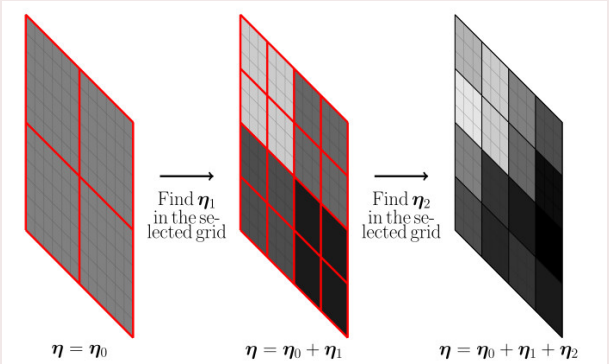
- Typically used up to a few hundred variables.
- Requires to build n-dimensional models.
- Here $n = 3072$!

Off-the-shelf solver and adversarial

BOBYQA solver (Original, Powell '09)

- Typically used up to a few hundred variables.
- Requires to build n-dimensional models.
- Here $n = 3072$!

BOBYQA for adversarial (Ughi et al '19)



- Random sub-sampling of variables (also used in ZOO/Frank-Wolfe).
- Hierarchical lifting, built through randomness (also used in GenAttack).

Other derivative-free techniques

True label:	brambling	cannon	pug-dog	fly	armadillo	balloon
Perturbed image **(ZO-PSGD):**						
Prediction label:	goldfinch	plow	bucket	longicorn	croquet ball	parachute
ℓ_2 Distortion:	35.2137	87.7304	72.3397	111.068	172.719	35.9330
# of queries:	7640	150	130	50	210	5830
Perturbed image **(ZO-SMD):**						
Prediction label:	goldfinch	plow	bucket	cicada	croquet ball	parachute
ℓ_2 distortion:	8.9708	26.0126	20.9504	30.0968	45.097	10.9023
# of queries:	29980	350	260	140	570	15830
Perturbed image **(ZO-AdaMM):**						
Prediction label:	goldfinch	plow	bucket	cicada	croquet ball	parachute
ℓ_2 distortion:	8.0502	5.7359	4.5753	4.4456	6.3149	7.7405
# of queries:	53300	1710	790	540	2780	32900
Perturbed image **(ZO-NES):**						
Prediction label:	goldfinch	plow	bucket	cicada	croquet ball	parachute
ℓ_2 distortion:	54.9956	34.5852	28.7035	28.9158	40.1483	51.7116
# of queries:	22110	1080	830	430	2280	15340

Zeroth-order with ImageNet on InceptionV3 (Liu et al '20).

$\rightarrow$ Mostly randomized variants with convergence rates.

Other derivative-free techniques ('ed)

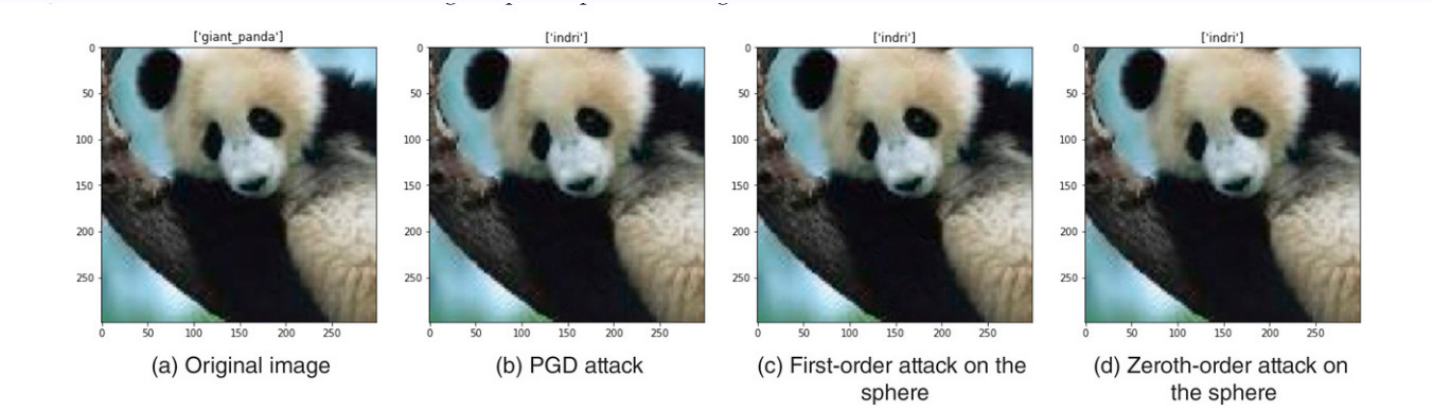
Riemannian Zeroth-Order (Li et al '23)

- Compute attacks on the sphere

$$\underset{\|\boldsymbol{z}-\boldsymbol{x}\|=\epsilon}{\text{maximize}} \ \mathcal{L}(h(\boldsymbol{z}), y).$$

  $\rightarrow$ Riemannian structure!

- Randomized zeroth-order scheme
  $\rightarrow$ Aggregate finite-difference estimates.



(a) Original image (b) PGD attack (c) First-order attack on the sphere (d) Zeroth-order attack on the sphere

Attack on ImageNet, $\|\cdot\| = \|\cdot\|_2$, $\epsilon = 0.05$.

Recap: Black-box adversarial attacks

- Query-based methods, akin to DFO.
- Numerous variants and benchmarking tasks.
- Mostly of zeroth-order type.

Recap: Black-box adversarial attacks

- Query-based methods, akin to DFO.
- Numerous variants and benchmarking tasks.
- Mostly of zeroth-order type.

Patterns

- Randomization plays a key role for scalability.
- Exploiting structure is also important.
- Challenging to apply DFO solvers other than zeroth-order.

Recap: Black-box adversarial attacks

- Query-based methods, akin to DFO.
- Numerous variants and benchmarking tasks.
- Mostly of zeroth-order type.

Patterns

- Randomization plays a key role for scalability.
- Exploiting structure is also important.
- Challenging to apply DFO solvers other than zeroth-order.

Our objectives

- Adapt general-purpose direct search to adversarial attacks.
- Develop structured attacks with randomness.

Roadmap

1. Adversarial attacks
2. Black-box adversarial attacks
3. Adversarial attacks via direct search

Derivative-free optimization via direct search

$$\underset{z \in \mathbb{R}^n}{\text{maximize}} \quad f(z)$$

Blackbox/Derivative-free optimization

- **Derivatives unavailable for algorithmic use.**
- Only access to values of f.
- **Cost** Number of function evaluations.

Derivative-free optimization via direct search

$$\underset{z \in \mathbb{R}^n}{\text{maximize}} \quad f(z)$$

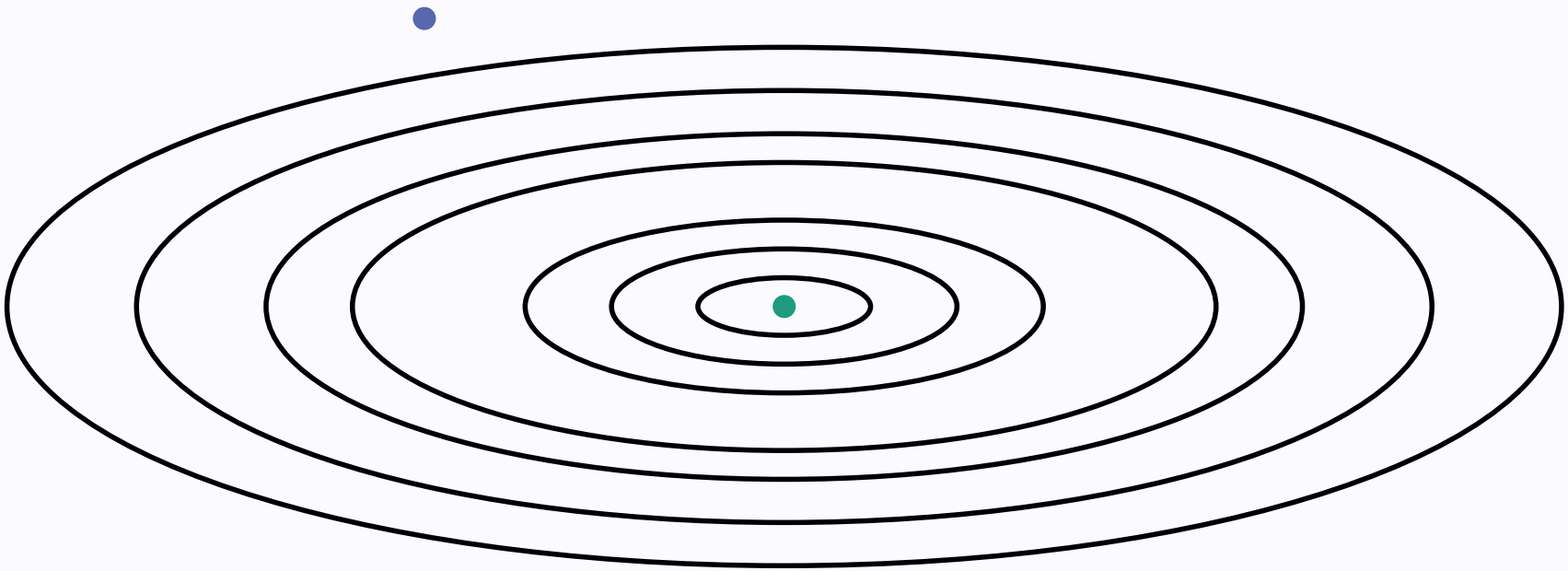
Blackbox/Derivative-free optimization

- **Derivatives unavailable for algorithmic use.**
- Only access to values of f.
- **Cost** Number of function evaluations.

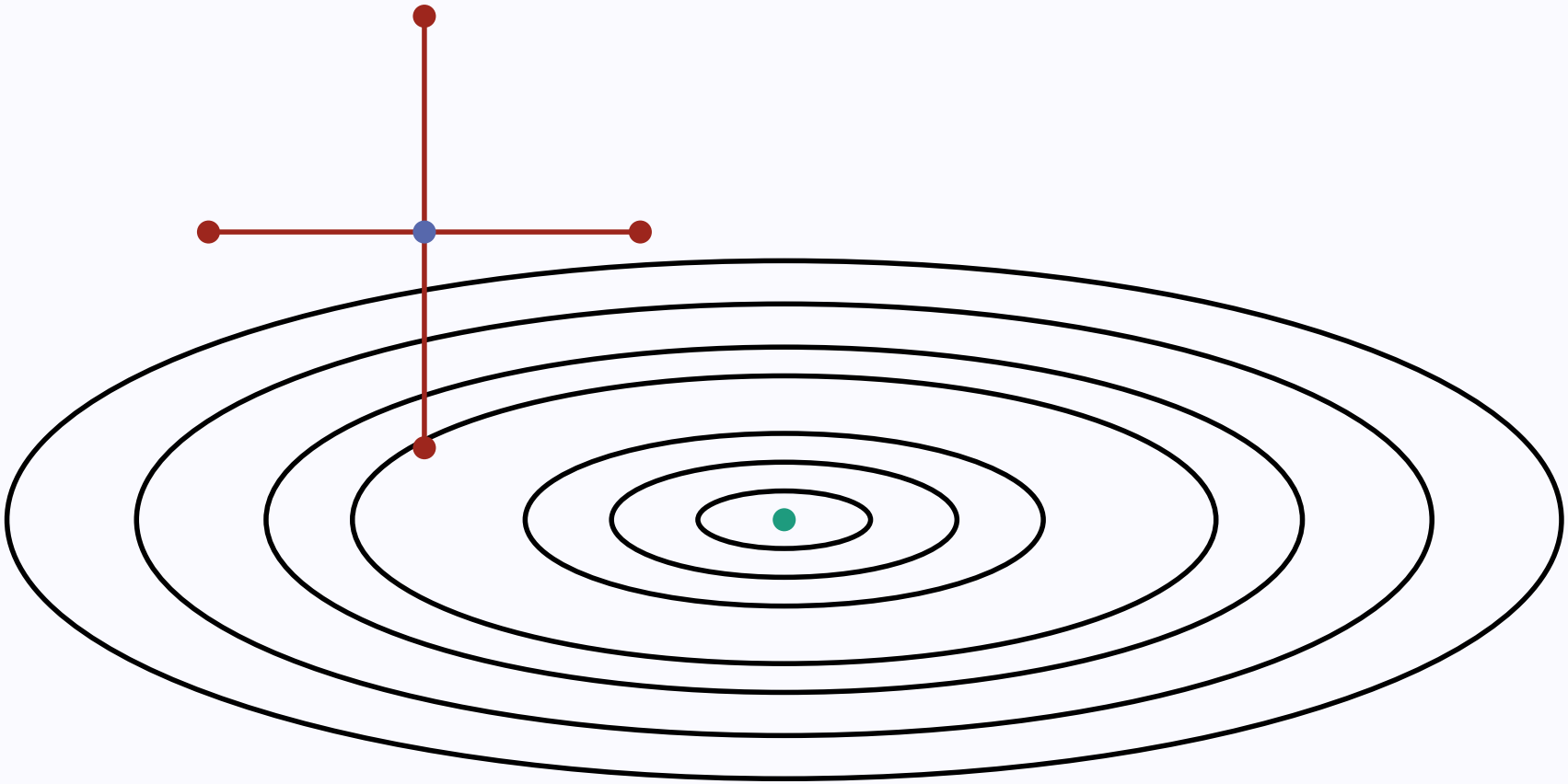
Direct search

- Proceed by polling the space along suitable directions.
- Rich convergence theory (Kolda et al '03, Dzahini et al. '25).

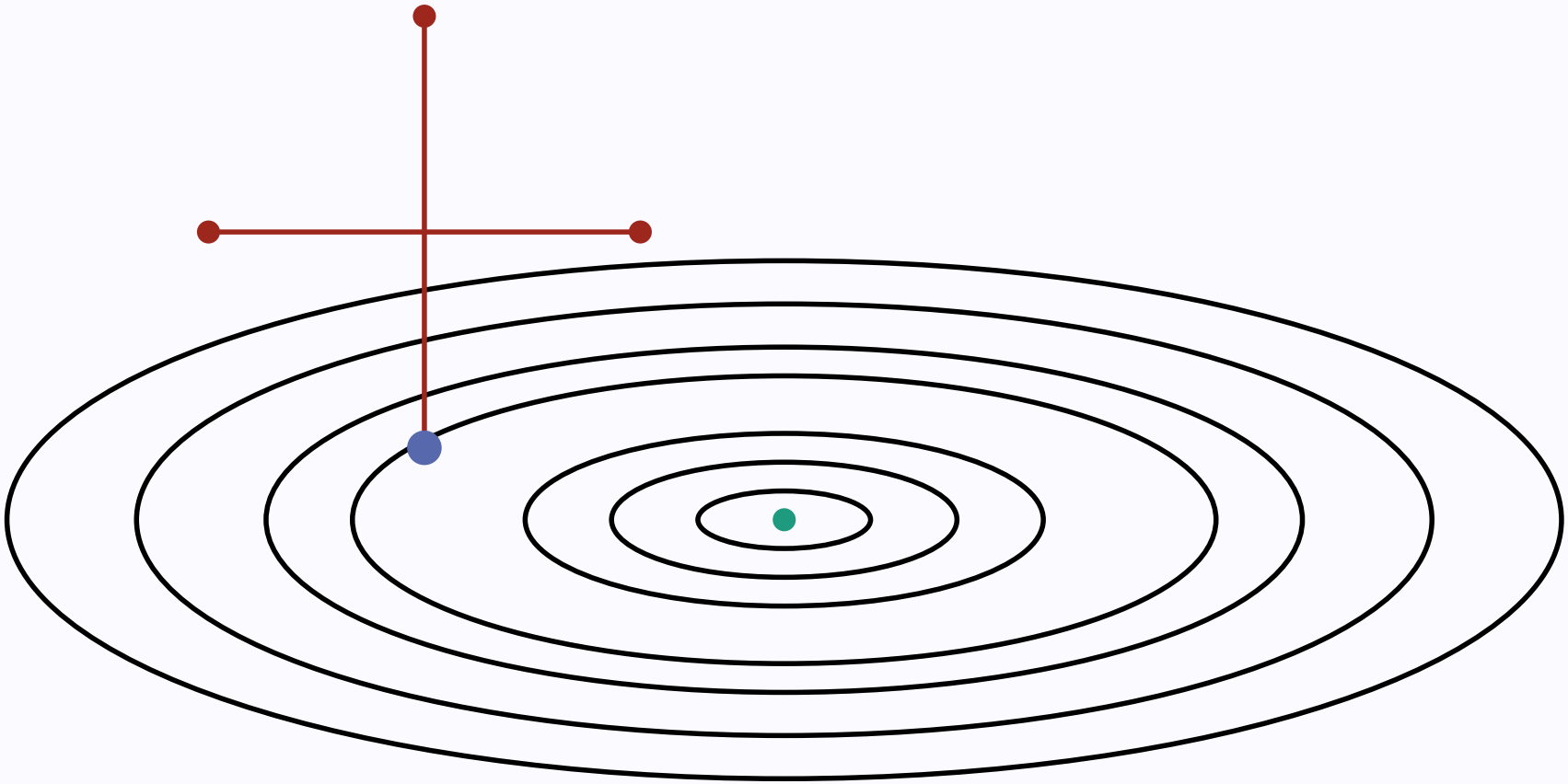
Example: Coordinate search



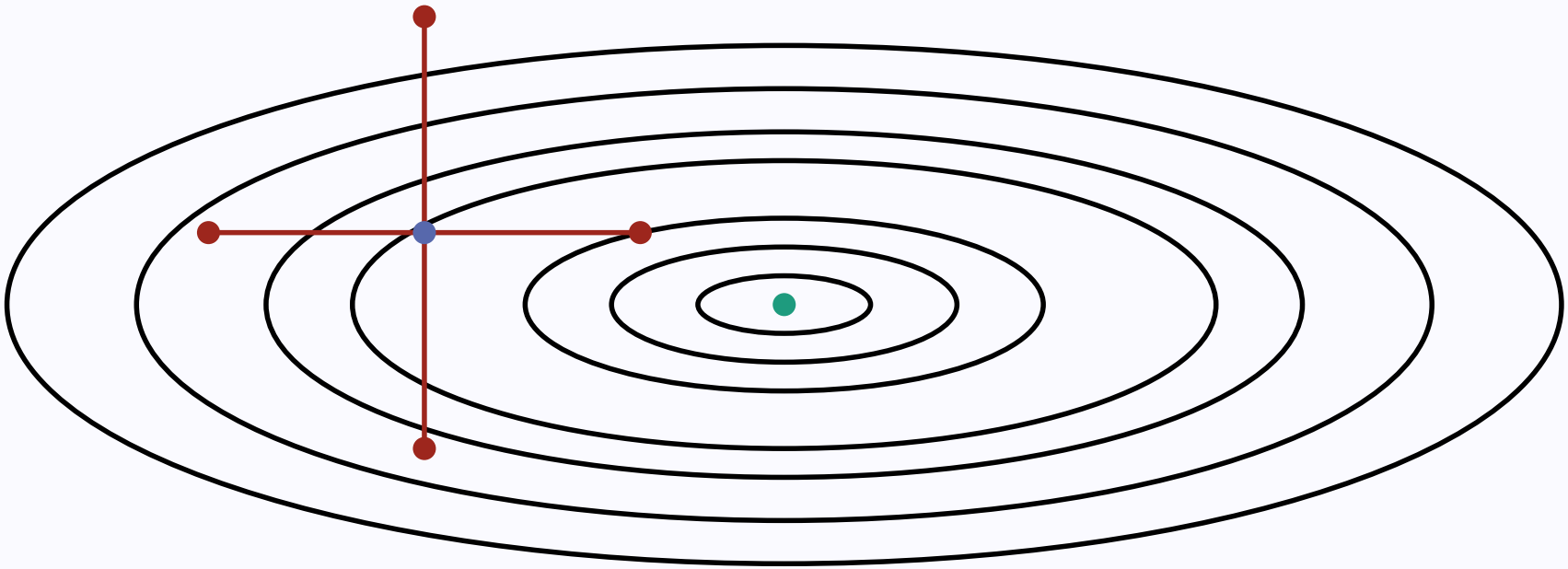
Example: Coordinate search



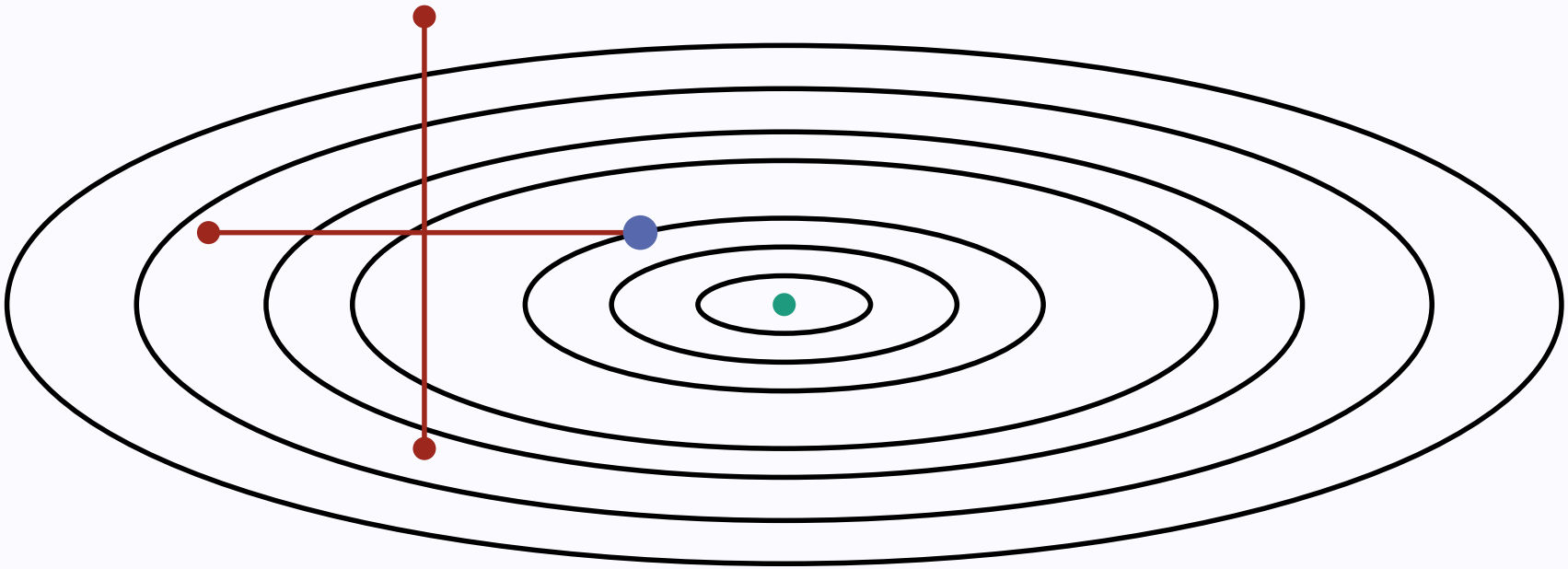
Example: Coordinate search



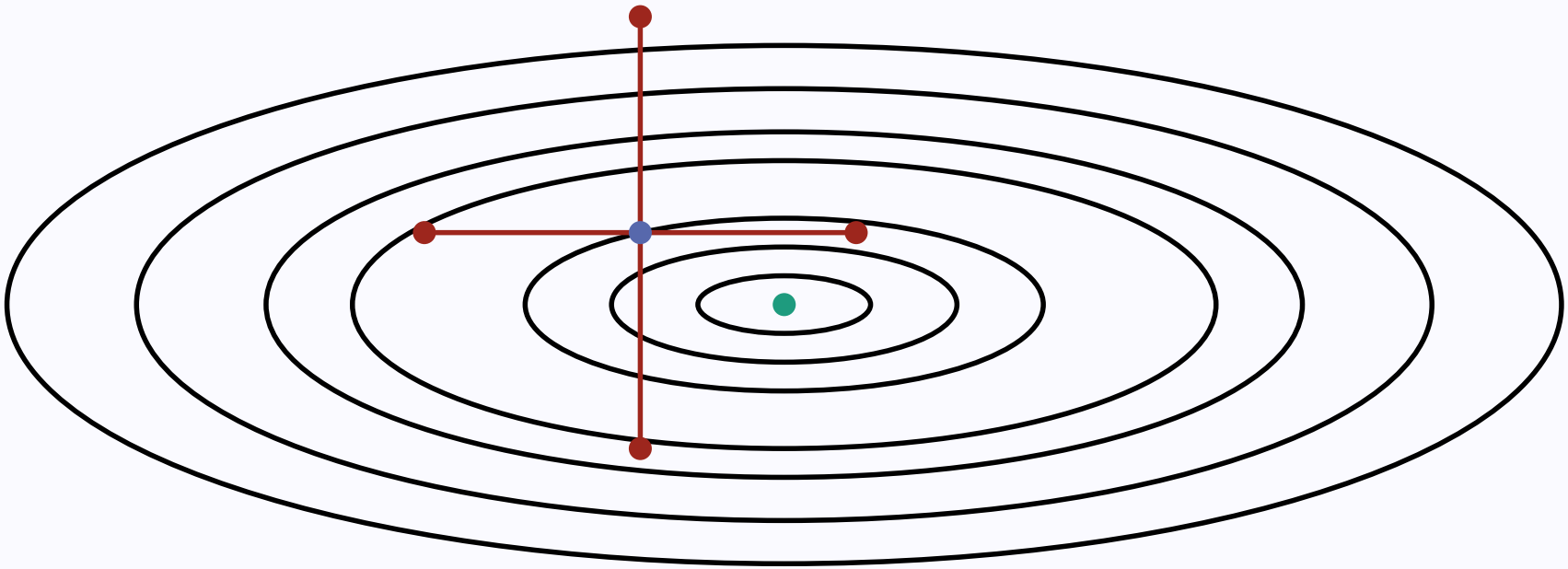
Example: Coordinate search



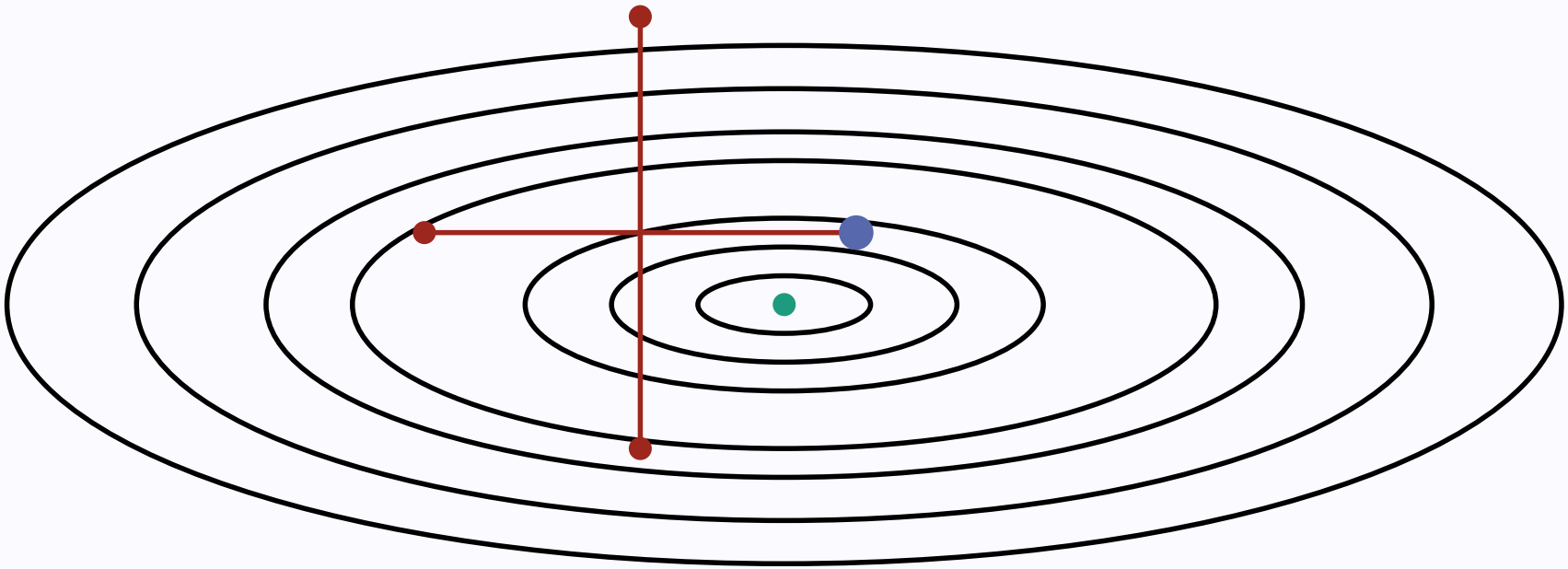
Example: Coordinate search



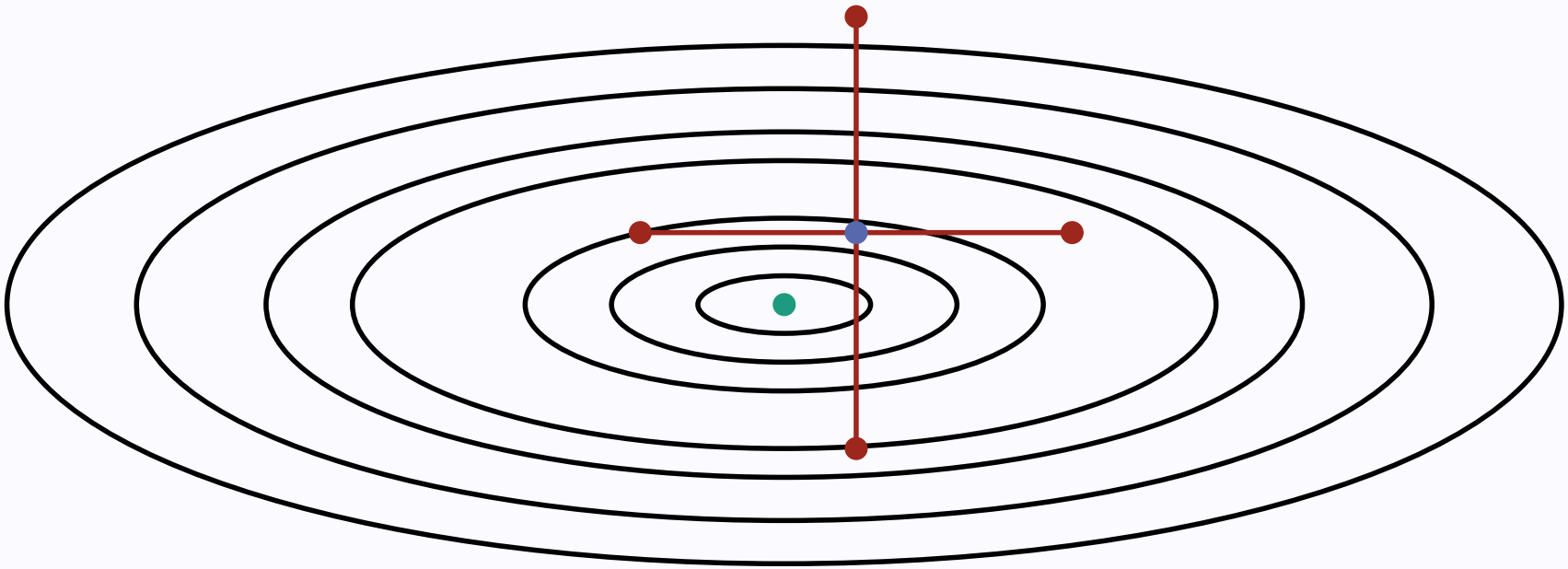
Example: Coordinate search



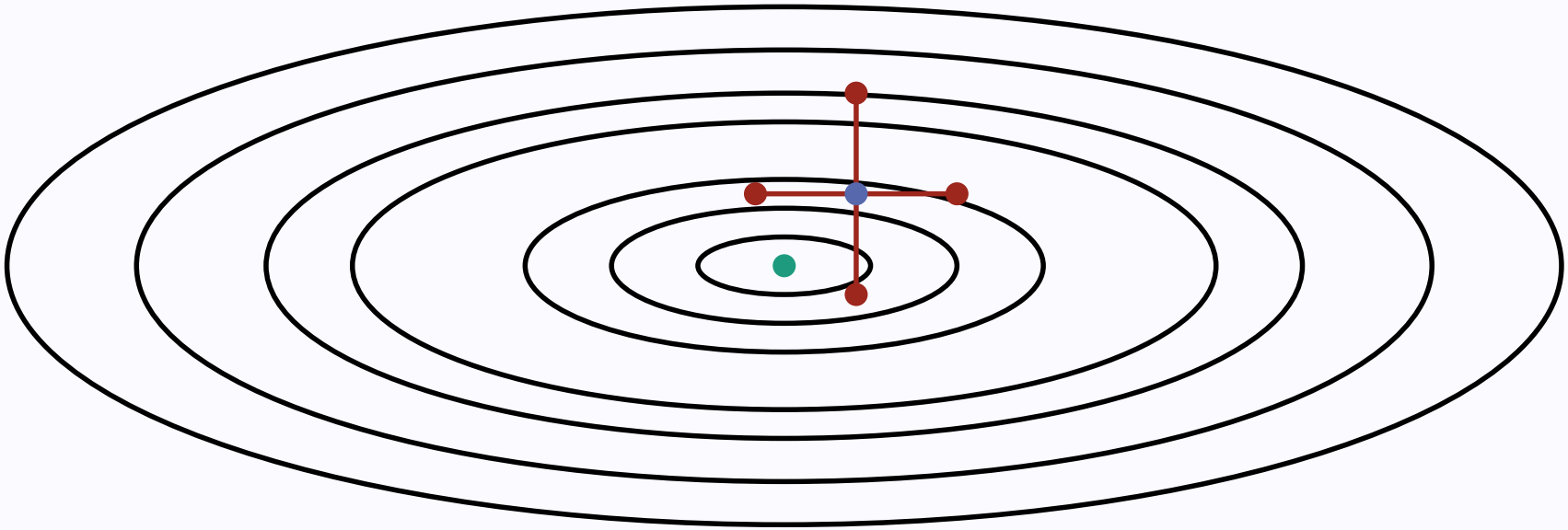
Example: Coordinate search



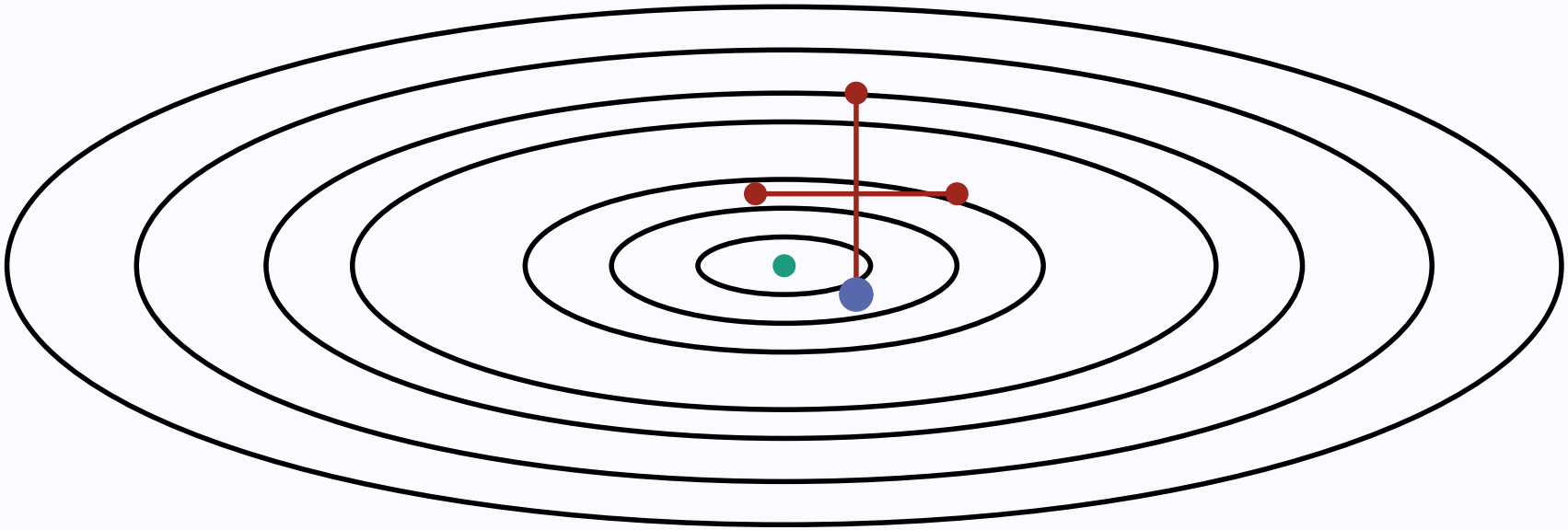
Example: Coordinate search



Example: Coordinate search



Example: Coordinate search



Basic direct-search framework

Problem $\text{maximize}_{z \in \mathbb{R}^n} \quad f(z)$.

Inputs $z_0 \in \mathbb{R}^n$, $\alpha_0 > 0$, $\eta > 0$.
For $k = 0, 1, \ldots$

- Choose $\mathcal{D}_k \subset \mathbb{R}^n$.

Basic direct-search framework

Problem $\text{maximize}_{\boldsymbol{z} \in \mathbb{R}^n} \quad f(\boldsymbol{z})$.

Inputs $\boldsymbol{z}_0 \in \mathbb{R}^n$, $\alpha_0 > 0$, $\eta > 0$.
For $k = 0, 1, \ldots$

- Choose $\mathcal{D}_k \subset \mathbb{R}^n$.
- If $\exists\ \boldsymbol{d}_k \in \mathcal{D}_k$ such that

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) > f(\boldsymbol{x}_k) + \eta \alpha_k^2 \|\boldsymbol{d}_k\|^2$$

set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, $\alpha_{k+1} := 2\alpha_k$.

Basic direct-search framework

Problem $\text{maximize}_{\boldsymbol{z} \in \mathbb{R}^n} \quad f(\boldsymbol{z})$.

Inputs $\boldsymbol{z}_0 \in \mathbb{R}^n$, $\alpha_0 > 0$, $\eta > 0$.
For $k = 0, 1, \ldots$

- Choose $\mathcal{D}_k \subset \mathbb{R}^n$.
- If $\exists\ \boldsymbol{d}_k \in \mathcal{D}_k$ such that

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) > f(\boldsymbol{x}_k) + \eta \alpha_k^2 \|\boldsymbol{d}_k\|^2$$

  set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, $\alpha_{k+1} := 2\alpha_k$.
- Otherwise, set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k$, $\alpha_{k+1} := 0.5\alpha_k$.

Basic direct-search framework

Problem $\text{maximize}_{\boldsymbol{z} \in \mathbb{R}^n} \quad f(\boldsymbol{z})$.

Inputs $\boldsymbol{z}_0 \in \mathbb{R}^n$, $\alpha_0 > 0$, $\eta > 0$.
For $k = 0, 1, \ldots$

- **Choose a PSS** $\mathcal{D}_k \subset \mathbb{R}^n$.
- If $\exists\ \boldsymbol{d}_k \in \mathcal{D}_k$ such that

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) > f(\boldsymbol{x}_k) + \eta \alpha_k^2 \|\boldsymbol{d}_k\|^2$$

  set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, $\alpha_{k+1} := 2\alpha_k$.
- Otherwise, set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k$, $\alpha_{k+1} := 0.5\alpha_k$.

Positive Spanning Set (PSS)

$$\mathcal{D} \subset \mathbb{R}^n \text{ PSS} \qquad \Longleftrightarrow \qquad \text{cm}(\mathcal{D}) = \max_{\boldsymbol{d} \in \mathcal{D}} \min_{\|\boldsymbol{v}\|=1} \frac{\boldsymbol{d}^{\mathrm{T}} \boldsymbol{v}}{\|\boldsymbol{d}\|} > 0.$$

$\text{cm}(\mathcal{D})$: Cosine measure.

Complexity of direct search (smooth setting)

Goal $\|\nabla f(\boldsymbol{z}_K)\| \leq \epsilon_g$.

Theorem (Vicente '12)

If $\mathcal{D}_k = \mathcal{D}$ PSS $\forall k$, the method takes at most

$$\mathcal{O}\left(|\mathcal{D}| \, \mathrm{cm}(\mathcal{D})^{-2} \, \epsilon_g^{-2} \right)$$

calls to f.

Complexity of direct search (smooth setting)

Goal $\|\nabla f(z_K)\| \leq \epsilon_g$.

Theorem (Vicente '12)

If $\mathcal{D}_k = \mathcal{D}$ PSS $\forall k$, the method takes at most

$$\mathcal{O}\left(|\mathcal{D}| \, \mathrm{cm}(\mathcal{D})^{-2} \, \epsilon_g^{-2} \right)$$

calls to f.

- $|\mathcal{D}|$ and $\mathrm{cm}(\mathcal{D})$ depend on dimension n!
- Best known bound:

$$|\mathcal{D}| = \mathcal{O}(n), \mathrm{cm}(\mathcal{D}) = \mathcal{O}(n^{-1/2}) \quad \Rightarrow \quad \mathcal{O}(n^2 \, \epsilon_g^{-2}).$$

Complexity of direct search (smooth setting)

Goal $\|\nabla f(z_K)\| \leq \epsilon_g$.

Theorem (Vicente '12)

If $\mathcal{D}_k = \mathcal{D}$ PSS $\forall k$, the method takes at most

$$\mathcal{O}\left(|\mathcal{D}| \, \mathrm{cm}(\mathcal{D})^{-2} \, \epsilon_g^{-2} \right)$$

calls to f.

- $|\mathcal{D}|$ and $\mathrm{cm}(\mathcal{D})$ depend on dimension n!
- Best known bound:

$$|\mathcal{D}| = \mathcal{O}(n), \mathrm{cm}(\mathcal{D}) = \mathcal{O}(n^{-1/2}) \quad \Rightarrow \quad \mathcal{O}(n^2 \, \epsilon_g^{-2}).$$

Can we improve using randomness?

Using randomness

Our approach (w/ Lindon Roberts)

- Consider a random subspace of dimension $r \leq n$;
- Use a PSS to approximate the projected gradient in the subspace;
- Guarantee sufficient gradient information **in probability**.

What it brings us

- Use random directions.
- Much less than ambient dimension n!

Not the only game in town (1/2)

Probabilistic descent (Gratton et al '15)

- Use directions $[\boldsymbol{d} \ -\boldsymbol{d}]$ with $\boldsymbol{d} \sim \mathcal{U}(\mathbb{S}^{n-1})$.
- Complexity improves from $\mathcal{O}(n^2\epsilon^{-2})$ to $\mathcal{O}(n\epsilon^{-2})$ $(m = 2)$.
- Limited to one distribution.

Not the only game in town (1/2)

Probabilistic descent (Gratton et al '15)

- Use directions $[\boldsymbol{d} \ -\boldsymbol{d}]$ with $\boldsymbol{d} \sim \mathcal{U}(\mathbb{S}^{n-1})$.
- Complexity improves from $\mathcal{O}(n^2\epsilon^{-2})$ to $\mathcal{O}(n\epsilon^{-2})$ $(m = 2)$.
- Limited to one distribution.

Gaussian smoothing approach: Draw $\boldsymbol{d} \sim \mathcal{N}(0, \boldsymbol{I})$ and use

$$\frac{f(\boldsymbol{x} + \delta\boldsymbol{d}) - f(\boldsymbol{x})}{\delta}\boldsymbol{d} \quad \text{or} \quad \frac{f(\boldsymbol{x} + \delta\boldsymbol{d}) - f(\boldsymbol{x} - \delta\boldsymbol{d})}{\delta}\boldsymbol{d}.$$

Random gradient-free method (Nesterov and Spokoiny 2017),
Stochastic three-point method (Bergou et al, 2020).

- Also achieve $\mathcal{O}(n\epsilon^{-2})$ bound.
- Use one-dimensional subspace based on Gaussian vectors.
- Use fixed or decreasing stepsizes.

Not the only game in town (2/2)

Zeroth-order (Kozak et al '21, '22)

- Estimate directional derivatives directly.
- Use orthogonal random directions $\boldsymbol{Q} \in \mathbb{R}^{n \times r}$, $\boldsymbol{Q}^{\mathrm{T}} \boldsymbol{Q} = \boldsymbol{I}$.
- Complexity results for convex/PL functions.

Not the only game in town (2/2)

Zeroth-order (Kozak et al '21, '22)

- Estimate directional derivatives directly.
- Use orthogonal random directions $\boldsymbol{Q} \in \mathbb{R}^{n \times r}$, $\boldsymbol{Q}^{\mathrm{T}} \boldsymbol{Q} = \boldsymbol{I}$.
- Complexity results for convex/PL functions.

Our approach

- General, subspace-based framework.
- Inspiration: Model-based methods (Cartis and Roberts '23, Dzahini and Wild '22a).

Direct-search algorithm (in subspaces)

Inputs: $\boldsymbol{z}_0 \in \mathbb{R}^n$, $\alpha_0 > 0$, $\eta > 0$.
Iteration k**:** Given $(\boldsymbol{z}_k, \alpha_k)$,

- Choose $\boldsymbol{P}_k \in \mathbb{R}^{r \times n}$ **at random**.
- Choose $\mathcal{D}_k \subset \mathbb{R}^r$ having m vectors.
- If $\exists\ \boldsymbol{d}_k \in \mathcal{D}_k$ such that

$$f(\boldsymbol{z}_k + \alpha_k \boldsymbol{P}_k^{\mathrm{T}} \boldsymbol{d}_k) > f(\boldsymbol{z}_k) + \eta \alpha_k^2 \|\boldsymbol{P}_k^{\mathrm{T}} \boldsymbol{d}_k\|^2,$$

  set $\boldsymbol{z}_{k+1} := \boldsymbol{z}_k + \alpha_k \boldsymbol{P}_k^{\mathrm{T}} \boldsymbol{d}_k$, $\alpha_{k+1} := 2\alpha_k$.
- Otherwise, set $\boldsymbol{z}_{k+1} := \boldsymbol{z}_k$, $\alpha_{k+1} := \alpha_k/2$.

Complexity analysis (smooth case)

Goal $\|\nabla f(\boldsymbol{z}_K)\| \leq \epsilon_g$.

Theorem (Roberts, R. '23)

Assume:

- $\{\boldsymbol{P}_k\}$ $(p, \sigma, P_{\max})$-well aligned with $p > 1/2$.
- $\mathcal{D}_k$ best PSS in r-dimensional subspace.

Let N_{ϵ_g} the number of function evaluations needed to have $\|\nabla f(\boldsymbol{z}_K)\| \leq \epsilon_g$. Then,

$$\mathbb{P}\left(N_\epsilon \leq \mathcal{O}\left(r^2 \phi \epsilon_g^{-2}\right)\right) \geq 1 - \exp\left(-\mathcal{O}\left(\phi \epsilon_g^{-2}\right)\right).$$

where $\phi = \sigma^{-2} P_{\max}^4$.

Complexity analysis (smooth case)

Goal $\|\nabla f(\boldsymbol{z}_K)\| \leq \epsilon_g$.

Theorem (Roberts, R. '23)

Assume:

- $\{\boldsymbol{P}_k\}$ $(p, \sigma, P_{\max})$-well aligned with $p > 1/2$.
- $\mathcal{D}_k$ best PSS in r-dimensional subspace.

Let N_{ϵ_g} the number of function evaluations needed to have $\|\nabla f(\boldsymbol{z}_K)\| \leq \epsilon_g$. Then,

$$\mathbb{P}\left(N_\epsilon \leq \mathcal{O}\left(r^2 \phi \epsilon_g^{-2}\right)\right) \geq 1 - \exp\left(-\mathcal{O}\left(\phi \epsilon_g^{-2}\right)\right).$$

where $\phi = \sigma^{-2} P_{\max}^4$.

How does $r^2\phi$ depend on n?

Randomized subspace techniques

Setup

- Use random $\boldsymbol{P}_k \in \mathbb{R}^{r \times n}$.
- Complexity scales as $r^2 \sigma^{-2} P_{\max}^{-4}$.

Setup

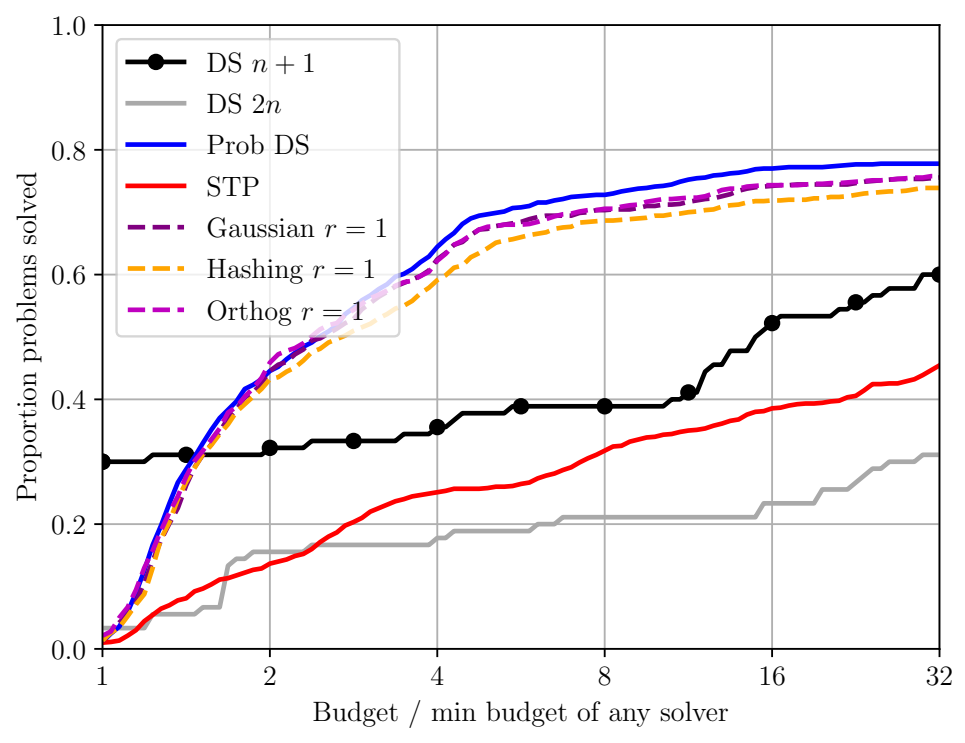
- Use random $\boldsymbol{P}_k \in \mathbb{R}^{r \times n}$.
- Complexity scales as $r^2 \sigma^{-2} P_{\max}^{-4}$.

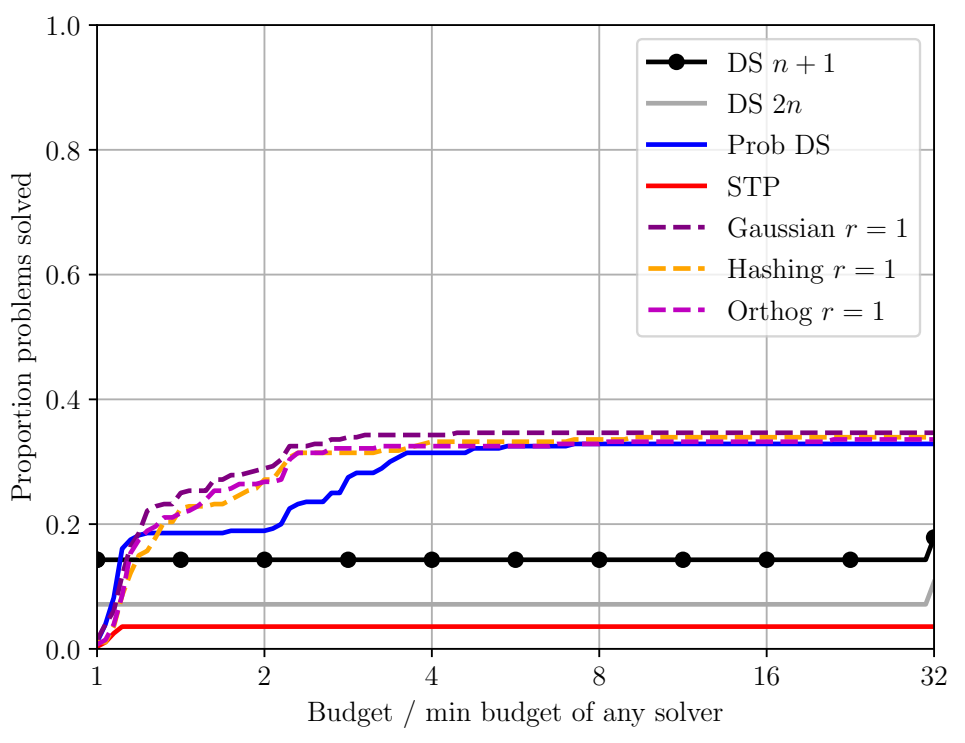
$\boldsymbol{P}_k$	σ	$P_{\max}$
Gaussian	$\Theta(\sqrt{n/r})$	$\Theta(\sqrt{n/r})$
Hashing	$\Theta(\sqrt{n/r})$ (Dzahini & Wild '22b)	$\sqrt{n}$
Orthogonal	$\sqrt{n/r}$	$\sqrt{n/r}$.

Best bound

$$r = \mathcal{O}(1) \quad \Rightarrow \quad \mathcal{O}(n\epsilon_g^{-2}).$$

Numerical illustration





Left: Medium scale; Right: Large scale.

- Operating in random subspaces works!
- But always a (hidden) dependency on n!

$\rightarrow$ `pip install directsearch`

Takeways

Direct search

- Simple to implement.
- Theoretical guarantees$\rightarrow$ Dependency on n.
- Competitor: Zeroth-order schemes!

Random subspaces

- Help scale things up (on standard DFO benchmarks).
- Still dependent on problem dimension n (not enough for adversarial attacks).

Riemannian optimization setup

$$\underset{\boldsymbol{Z} \in \mathcal{M}}{\text{maximize}}\, f(\boldsymbol{Z}).$$

- f: Black-box objective.
- $\boldsymbol{Z}$: Matrix/Tensor object (e.g. $32 \times 32 \times 3$ images).
- $\mathcal{M}$: Known Riemannian manifold.

Motivating examples

- Spheres in Matrix/Tensor spaces: $\mathcal{M} = \{\boldsymbol{Z}, \|\boldsymbol{Z} - \boldsymbol{X}\| = \epsilon\}$.
- Symmetric Positive Definite Matrices (SPDNet).
- Low-rank matrix/tensors.

Riemannian direct search

Problem: maximize$_{\boldsymbol{Z} \in \mathcal{M}} f(\boldsymbol{Z})$.

Inputs $\boldsymbol{Z}_0 \in \mathcal{M}$, $\alpha_0 > 0$, $\eta > 0$.
For $k = 0, 1, \ldots$

- Choose a PSS $\mathcal{D}(\boldsymbol{Z}_k)$ for $\mathcal{T}_{\boldsymbol{Z}_k}\mathcal{M}$.
- If $\exists\ \boldsymbol{D}_k \in \mathcal{D}_k$ such that

$$f(\mathcal{R}_{\boldsymbol{Z}_k}(\alpha_k \boldsymbol{D}_k)) > f(\boldsymbol{Z}_k) + \eta \alpha_k^2 \|\boldsymbol{D}_k\|_{\boldsymbol{Z}_k}^2$$

  set $\boldsymbol{Z}_{k+1} := \mathcal{R}_{\boldsymbol{Z}_k}(\alpha_k \boldsymbol{D}_k)$, $\alpha_{k+1} := 2\alpha_k$.
- Otherwise, set $\boldsymbol{Z}_{k+1} := \boldsymbol{Z}_k$, $\alpha_{k+1} := 0.5\alpha_k$.

Riemannian direct search

Problem: maximize$_{\boldsymbol{Z} \in \mathcal{M}} f(\boldsymbol{Z})$.

Inputs $\boldsymbol{Z}_0 \in \mathcal{M}$, $\alpha_0 > 0$, $\eta > 0$.
For $k = 0, 1, \ldots$

- Choose a PSS $\mathcal{D}(\boldsymbol{Z}_k)$ for $\mathcal{T}_{\boldsymbol{Z}_k}\mathcal{M}$.
- If $\exists\ \boldsymbol{D}_k \in \mathcal{D}_k$ such that

$$f(\mathcal{R}_{\boldsymbol{Z}_k}(\alpha_k \boldsymbol{D}_k)) > f(\boldsymbol{Z}_k) + \eta \alpha_k^2 \|\boldsymbol{D}_k\|_{\boldsymbol{Z}_k}^2$$

  set $\boldsymbol{Z}_{k+1} := \mathcal{R}_{\boldsymbol{Z}_k}(\alpha_k \boldsymbol{D}_k)$, $\alpha_{k+1} := 2\alpha_k$.
- Otherwise, set $\boldsymbol{Z}_{k+1} := \boldsymbol{Z}_k$, $\alpha_{k+1} := 0.5\alpha_k$.

- $\mathcal{T}_{\boldsymbol{Z}_k}\mathcal{M}$: Tangent space for $\mathcal{M}$ at $\boldsymbol{Z}_k$ with norm $\|\cdot\|_{\boldsymbol{Z}_k}$ $\rightarrow$ Rigorous extension of PSS possible!
- $\mathcal{R}_{\boldsymbol{Z}_k}(\alpha_k \boldsymbol{D}_k)$: Retraction (brings back onto the manifold) $\rightarrow$ Guarantees feasible iterates!

A theoretically sound approach

Goal $\|\text{grad} f(\boldsymbol{Z}_K)\| \leq \epsilon_g$ (Riemannian gradient).

Theorem (Cavarretta, Goyens, R., Yger '25)

The method takes at most

$$\mathcal{O}\left(\max_{\boldsymbol{Z} \in \mathcal{M}} |\mathcal{D}(\boldsymbol{Z})| \, \text{cm}(\mathcal{D}(\boldsymbol{Z}))^{-2} \, \epsilon_g^{-2} \right)$$

calls to f.

A theoretically sound approach

Goal $\|\text{grad} f(\boldsymbol{Z}_K)\| \leq \epsilon_g$ (Riemannian gradient).

Theorem (Cavarretta, Goyens, R., Yger '25)

The method takes at most

$$\mathcal{O}\left(\max_{\boldsymbol{Z} \in \mathcal{M}} |\mathcal{D}(\boldsymbol{Z})| \, \text{cm}(\mathcal{D}(\boldsymbol{Z}))^{-2} \, \epsilon_g^{-2} \right)$$

calls to f.

Extension: probabilistic variant

- Sample tangent subspaces (Gutman & Ho-Nguyen '22).
- Generalization of Euclidean case (Roberts, R. '23).

Test case: Low-rank attacks

Recall: Adversarial attack problem (matrix version)

$$\underset{\boldsymbol{Z} \in \mathcal{B}(\boldsymbol{X}, \epsilon)}{\text{maximize}} \ \mathcal{L}(h(\boldsymbol{Z}), y)$$

$\rightarrow \mathcal{B}(\cdot)$ ℓ_2 or nuclear norm ball.

Test case: Low-rank attacks

Recall: Adversarial attack problem (matrix version)

$$\underset{\boldsymbol{Z} \in \mathcal{B}(\boldsymbol{X}, \epsilon)}{\text{maximize}} \ \mathcal{L}(h(\boldsymbol{Z}), y)$$

$\rightarrow \mathcal{B}(\cdot)$ ℓ_2 or nuclear norm ball.

Low-rank PGD attacks (Savostianova et al '24)

- Observation: Projected Gradient Descent computes approximately low-rank attacks ($\boldsymbol{Z} - \boldsymbol{X}$ has many small singular values).

Test case: Low-rank attacks

Recall: Adversarial attack problem (matrix version)

$$\underset{\boldsymbol{Z} \in \mathcal{B}(\boldsymbol{X}, \epsilon)}{\text{maximize}} \ \mathcal{L}(h(\boldsymbol{Z}), y)$$

$\rightarrow \mathcal{B}(\cdot)$ ℓ_2 or nuclear norm ball.

Low-rank PGD attacks (Savostianova et al '24)

- Observation: Projected Gradient Descent computes approximately low-rank attacks ($\boldsymbol{Z} - \boldsymbol{X}$ has many small singular values).
- Approach: Enforce low-rank perturbation:

$$\underset{\boldsymbol{U}, \boldsymbol{V}}{\text{maximize}} \ \mathcal{L}(h(\boldsymbol{X} + \boldsymbol{U} \otimes \boldsymbol{V}), y) \quad \text{s.t.} \quad \boldsymbol{X} + \boldsymbol{U} \otimes \boldsymbol{V} \in \mathcal{B}(\boldsymbol{X}, \epsilon).$$

and solve with PGD applied to $\boldsymbol{U}$ and $\boldsymbol{V}$.

Test case: Low-rank attacks

Recall: Adversarial attack problem (matrix version)

$$\underset{\boldsymbol{Z} \in \mathcal{B}(\boldsymbol{X}, \epsilon)}{\text{maximize}} \; \mathcal{L}(h(\boldsymbol{Z}), y)$$

$\rightarrow \mathcal{B}(\cdot)$ ℓ_2 or nuclear norm ball.

Low-rank PGD attacks (Savostianova et al '24)

- Observation: Projected Gradient Descent computes approximately low-rank attacks ($\boldsymbol{Z} - \boldsymbol{X}$ has many small singular values).
- Approach: Enforce low-rank perturbation:

$$\underset{\boldsymbol{U}, \boldsymbol{V}}{\text{maximize}} \; \mathcal{L}(h(\boldsymbol{X} + \boldsymbol{U} \otimes \boldsymbol{V}), y) \quad \text{s.t.} \quad \boldsymbol{X} + \boldsymbol{U} \otimes \boldsymbol{V} \in \mathcal{B}(\boldsymbol{X}, \epsilon).$$

  and solve with PGD applied to $\boldsymbol{U}$ and $\boldsymbol{V}$.

What about low-rank black-box attacks?

Black-box low-rank attacks: Setup

Four methods (applied to factors $\boldsymbol{U}$, $\boldsymbol{V}$)

- **DS** Deterministic, uses fixed directions.
  $\rightarrow \mathcal{O}(\dim(\boldsymbol{U}) + \dim(\boldsymbol{V}))$ evaluations/iteration.
- **DS-PD(1)** Probabilistic, 1-dimensional subspace.
  $\rightarrow \mathcal{O}(1)$ evaluations/iteration.
- **DS-PD(rank)** Probabilistic, r-dimensional subspace for rank-r attacks.
  $\rightarrow \mathcal{O}(r)$ evaluations/iteration.
- **GFM** Gradient-Free Method, based on zeroth-order formula

$$\boldsymbol{W}_{k+1} = \boldsymbol{W}_k + \sigma_k \frac{f(\boldsymbol{W}_k + \mu_k \boldsymbol{D}_k) - f(\boldsymbol{W}_k - \mu_k \boldsymbol{D}_k)}{2\mu_k}$$

  over 1-dimensional subspaces.
  $\rightarrow \mathcal{O}(1)$ evaluations/iteration.

Black-box low-rank attacks: Setup

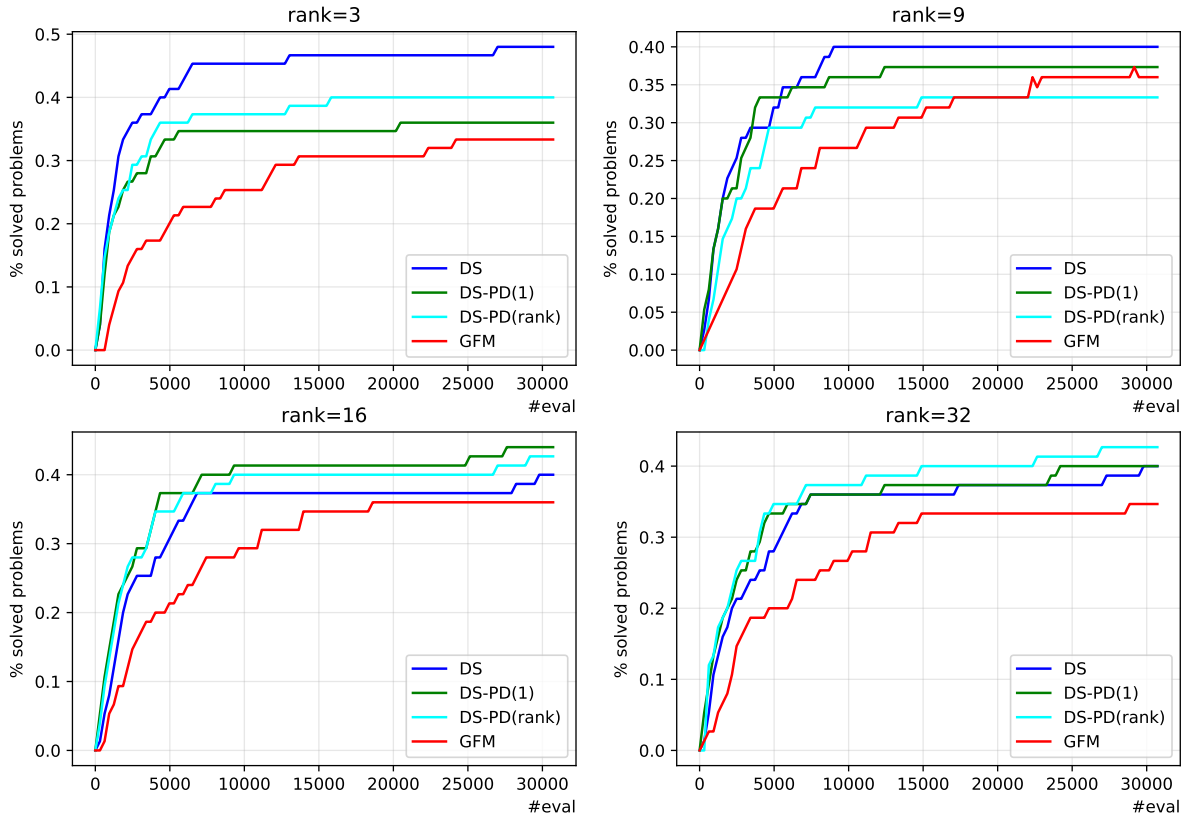
Model

- From RobustBench (Croce et al '21).
- WideResNet-28-10 trained on CIFAR-10.
- Problem dimension: $32 \times 32 \times 3 = 3072$.

Task

- Attack 75 images from CIFAR-10 (Target loss value).
- Budget: 10 PGD iterations with finite differences.
  $\rightarrow 32 \times 32 \times 3 \times 10 = 30720$ evaluations.

Black-box low-rank attacks: Results (fresh!)



Data profiles (top left is better)

- Small rank: Deterministic direct search is good.
- Increasing rank: Probabilistic variants for scalability!

Summary

Black-box adversarial attacks

- Hugely popular topic.
- Challenge: High dimensionality (but structure).

Summary

Black-box adversarial attacks

- Hugely popular topic.
- Challenge: High dimensionality (but structure).

Derivative-free optimization

- Classical techniques for low dimensions.
- Recent effort to scale up through randomization.

Summary

Black-box adversarial attacks

- Hugely popular topic.
- Challenge: High dimensionality (but structure).

Derivative-free optimization

- Classical techniques for low dimensions.
- Recent effort to scale up through randomization.

Our approach: Direct search

- Theory for Riemannian+Randomized versions.
- Promising results with low-rank perturbation of images.

References

- B. Cavarretta, F. Goyens, C. W. Royer and F. Yger, *Complexity guarantees and polling strategies for Riemannian direct-search methods*, arXiv:2511.15360, under review.
- K. J. Dzahini, F. Rinaldi, C. W. Royer and D. Zeffiro, *Direct search methods in the year 2025: Theoretical guarantees and algorithmic paradigms*, EURO Journal on Computational Optimization, 2025.
- L. Roberts and C. W. Royer, *Direct search based on probabilistic descent in reduced spaces*, SIAM Journal on Optimization, 2023.

→ Stay tuned for more!

References

- B. Cavarretta, F. Goyens, C. W. Royer and F. Yger, *Complexity guarantees and polling strategies for Riemannian direct-search methods*, arXiv:2511.15360, under review.
- K. J. Dzahini, F. Rinaldi, C. W. Royer and D. Zeffiro, *Direct search methods in the year 2025: Theoretical guarantees and algorithmic paradigms*, EURO Journal on Computational Optimization, 2025.
- L. Roberts and C. W. Royer, *Direct search based on probabilistic descent in reduced spaces*, SIAM Journal on Optimization, 2023.

→ Stay tuned for more!

Merci!

`clement.royer@lamsade.dauphine.fr`