Coin Sampling: Gradient-Based Bayesian Inference without Learning Rates

Louis Sharrock[1,2] Christopher Nemeth[1]

[1]Lancaster University

[2]University of Bristol

Outline

Background

The Problem

Sampling Problem: Generate samples from a target probability distribution π on $\mathbb{R}^d$, given some information about π.

Two common scenarios:

(i) The target admits a density $\pi(x)$ w.r.t. the Lebesgue measure, which is known up to some intractable normalising constant. That is,

$$\pi(x) := \frac{e^{-U(x)}}{Z},$$

where the potential $U : \mathbb{R}^d \to \mathbb{R}$ is known, and the constant $Z = \int_{\mathbb{R}^d} e^{-U(x)} \mathrm{d}x$ is unknown.

(ii) We have access to samples from the target π: $x_1, \ldots, x_n \sim \pi$.

We will focus on the first setting.

The Problem

Sampling Problem: Generate samples from a target probability distribution π on $\mathbb{R}^d$, given some information about π.

Two common scenarios:

(i) The target admits a density $\pi(x)$ w.r.t. the Lebesgue measure, which is known up to some intractable normalising constant. That is,

$$\pi(x) := \frac{e^{-U(x)}}{Z},$$

where the potential $U : \mathbb{R}^d \to \mathbb{R}$ is known, and the constant $Z = \int_{\mathbb{R}^d} e^{-U(x)} \mathrm{d}x$ is unknown.

(ii) We have access to samples from the target π: $x_1, \ldots, x_n \sim \pi$.

We will focus on the first setting.

The Problem

Sampling Problem: Generate samples from a target probability distribution π on $\mathbb{R}^d$, given some information about π.

Two common scenarios:

(i) The target admits a density $\pi(x)$ w.r.t. the Lebesgue measure, which is known up to some intractable normalising constant. That is,

$$\pi(x) := \frac{e^{-U(x)}}{Z},$$

where the potential $U : \mathbb{R}^d \to \mathbb{R}$ is known, and the constant $Z = \int_{\mathbb{R}^d} e^{-U(x)} \mathrm{d}x$ is unknown.

(ii) We have access to samples from the target π: $x_1, \ldots, x_n \sim \pi$.

We will focus on the first setting.

The Problem

Sampling Problem: Generate samples from a target probability distribution π on $\mathbb{R}^d$, given some information about π.

Two common scenarios:

(i) The target admits a density $\pi(x)$ w.r.t. the Lebesgue measure, which is known up to some intractable normalising constant. That is,

$$\pi(x) := \frac{e^{-U(x)}}{Z},$$

where the potential $U : \mathbb{R}^d \to \mathbb{R}$ is known, and the constant $Z = \int_{\mathbb{R}^d} e^{-U(x)} \mathrm{d}x$ is unknown.

(ii) We have access to samples from the target π: $x_1, \ldots, x_n \sim \pi$.

We will focus on the first setting.

The Problem

Sampling Problem: Generate samples from a target probability distribution π on $\mathbb{R}^d$, given some information about π.

Two common scenarios:

(i) The target admits a density $\pi(x)$ w.r.t. the Lebesgue measure, which is known up to some intractable normalising constant. That is,

$$\pi(x) := \frac{e^{-U(x)}}{Z},$$

where the potential $U : \mathbb{R}^d \to \mathbb{R}$ is known, and the constant $Z = \int_{\mathbb{R}^d} e^{-U(x)} \mathrm{d}x$ is unknown.

(ii) We have access to samples from the target π: $x_1, \ldots, x_n \sim \pi$.

We will focus on the first setting.

Motivation: Bayesian Inference

Bayesian Inference: learn the posterior distribution of a parameter $x \in \mathbb{R}^d$, given some observed data $\mathcal{D}$.

- Let $\mathcal{D} = (w_i, y_i)_{i=1}^p$ be a dataset of i.i.d. examples, with features w_i and labels y_i.
- Assume some underlying model for the data, parameterised by $x \in \mathbb{R}^d$, e.g.,

$$p(\mathcal{D}|x) = \prod_{i=1}^p p(y_i|w_i, x) \propto \exp\left(-\tfrac{1}{2}\sum_{i=1}^p ||y_i - f(w_i, x)||^2\right).$$

- Assume also some prior distribution $x \sim p_0$ for the parameter $x \in \mathbb{R}^d$, e.g.,

$$p_0(x) \propto \exp\left(-\frac{||x||^2}{2}\right).$$

Motivation: Bayesian Inference

Bayesian Inference: learn the posterior distribution of a parameter $x \in \mathbb{R}^d$, given some observed data $\mathcal{D}$.

- Let $\mathcal{D} = (w_i, y_i)_{i=1}^p$ be a dataset of i.i.d. examples, with features w_i and labels y_i.
- Assume some underlying model for the data, parameterised by $x \in \mathbb{R}^d$, e.g.,

$$p(\mathcal{D}|x) = \prod_{i=1}^p p(y_i|w_i, x) \propto \exp\left(-\tfrac{1}{2}\sum_{i=1}^p ||y_i - f(w_i, x)||^2\right).$$

- Assume also some prior distribution $x \sim p_0$ for the parameter $x \in \mathbb{R}^d$, e.g.,

$$p_0(x) \propto \exp\left(-\frac{||x||^2}{2}\right).$$

Motivation: Bayesian Inference

Bayesian Inference: learn the posterior distribution of a parameter $x \in \mathbb{R}^d$, given some observed data $\mathcal{D}$.

- Let $\mathcal{D} = (w_i, y_i)_{i=1}^p$ be a dataset of i.i.d. examples, with features w_i and labels y_i.
- Assume some underlying model for the data, parameterised by $x \in \mathbb{R}^d$, e.g.,

$$p(\mathcal{D}|x) = \prod_{i=1}^p p(y_i|w_i, x) \propto \exp\left(-\tfrac{1}{2}\sum_{i=1}^p ||y_i - f(w_i, x)||^2\right).$$

- Assume also some prior distribution $x \sim p_0$ for the parameter $x \in \mathbb{R}^d$, e.g.,

$$p_0(x) \propto \exp\left(-\frac{||x||^2}{2}\right).$$

Motivation: Bayesian Inference

Bayesian Inference: learn the posterior distribution of a parameter $x \in \mathbb{R}^d$, given some observed data $\mathcal{D}$.

- Let $\mathcal{D} = (w_i, y_i)_{i=1}^p$ be a dataset of i.i.d. examples, with features w_i and labels y_i.
- Assume some underlying model for the data, parameterised by $x \in \mathbb{R}^d$, e.g.,

$$p(\mathcal{D}|x) = \prod_{i=1}^p p(y_i|w_i, x) \propto \exp\left(-\tfrac{1}{2}\sum_{i=1}^p ||y_i - f(w_i, x)||^2\right).$$

- Assume also some prior distribution $x \sim p_0$ for the parameter $x \in \mathbb{R}^d$, e.g.,

$$p_0(x) \propto \exp\left(-\frac{||x||^2}{2}\right).$$

Motivation: Bayesian Inference

Bayesian Inference: learn the posterior distribution of a parameter $x \in \mathbb{R}^d$, given some observed data $\mathcal{D}$.

- By Bayes' rule, the **posterior distribution** is then given by

$$p(x|\mathcal{D}) = \frac{p(\mathcal{D}|x)p_0(x)}{Z}, \quad Z = \int_{\mathbb{R}^d} p(\mathcal{D}|x)p_0(x)\mathrm{d}x.$$

- In particular, writing $\pi := p(\cdot|\mathcal{D})$ for the posterior over the parameter $x \in \mathbb{R}^d$, we have

$$\pi(x) = \frac{e^{-U(x)}}{Z}, \quad U(x) = \frac{1}{2}\sum_{i=1}^{p} ||y_i - f(w_i, x)||^2 + \frac{||x||^2}{2}.$$

Motivation: Bayesian Inference

Bayesian Inference: learn the posterior distribution of a parameter $x \in \mathbb{R}^d$, given some observed data $\mathcal{D}$.

- By Bayes' rule, the **posterior distribution** is then given by

$$p(x|\mathcal{D}) = \frac{p(\mathcal{D}|x)p_0(x)}{Z}, \quad Z = \int_{\mathbb{R}^d} p(\mathcal{D}|x)p_0(x)\mathrm{d}x.$$

- In particular, writing $\pi := p(\cdot|\mathcal{D})$ for the posterior over the parameter $x \in \mathbb{R}^d$, we have

$$\pi(x) = \frac{e^{-U(x)}}{Z}, \quad U(x) = \frac{1}{2}\sum_{i=1}^{p} ||y_i - f(w_i, x)||^2 + \frac{||x||^2}{2}.$$

Sampling as Optimisation

Let's assume that $\pi \in \mathcal{P}_2(\mathbb{R}^d) := \{\mu : \int_{\mathbb{R}^d} ||x||^2 \mu(\mathrm{d}x) < \infty\}$.

We can then recast the sampling problem as an **optimisation problem**:

$$\pi = \underset{\mu \in \mathcal{P}_2(\mathbb{R}^d)}{\arg\min}\ \mathcal{F}(\mu),$$

where $\mathcal{F} : \mathcal{P}(\mathbb{R}^d) \to \mathbb{R}$ is a **dissimilarity functional** which is uniquely minimised at π.

A general strategy for solving this problem is to simulate a **time-discretisation** of the **gradient flow** of $\mathcal{F}$ over $\mathcal{P}_2(\mathbb{R}^d)$.

Sampling as Optimisation

Let's assume that $\pi \in \mathcal{P}_2(\mathbb{R}^d) := \{\mu : \int_{\mathbb{R}^d} ||x||^2 \mu(\mathrm{d}x) < \infty\}$.

We can then recast the sampling problem as an **optimisation problem**:

$$\pi = \underset{\mu \in \mathcal{P}_2(\mathbb{R}^d)}{\arg\min}\ \mathcal{F}(\mu),$$

where $\mathcal{F} : \mathcal{P}(\mathbb{R}^d) \to \mathbb{R}$ is a **dissimilarity functional** which is uniquely minimised at π.

A general strategy for solving this problem is to simulate a **time-discretisation** of the **gradient flow** of $\mathcal{F}$ over $\mathcal{P}_2(\mathbb{R}^d)$.

Sampling as Optimisation

Let's assume that $\pi \in \mathcal{P}_2(\mathbb{R}^d) := \{\mu : \int_{\mathbb{R}^d} ||x||^2 \mu(\mathrm{d}x) < \infty\}$.

We can then recast the sampling problem as an **optimisation problem**:

$$\pi = \underset{\mu \in \mathcal{P}_2(\mathbb{R}^d)}{\arg\min}\ \mathcal{F}(\mu),$$

where $\mathcal{F} : \mathcal{P}(\mathbb{R}^d) \to \mathbb{R}$ is a **dissimilarity functional** which is uniquely minimised at π.

A general strategy for solving this problem is to simulate a **time-discretisation** of the **gradient flow** of $\mathcal{F}$ over $\mathcal{P}_2(\mathbb{R}^d)$.

Sampling as Optimisation

This framework has been fruitful in the design and analysis of many existing sampling algorithms. For example:

- **Langevin Monte Carlo** (LMC): forward-flow discretisation of the gradient flow of the KL divergence with respect to the quadratic Wasserstein metric [Wib18, DMM19].
- **Stein Variational Gradient Descent** (SVGD): explicit Euler discretisation of the gradient flow of the KL divergence with respect to a kernelised Wasserstein metric [LW16, DNS23].
- Lots of other examples, e.g., **kernel Stein discrepancy descent** [KPCAF+21], **Laplacian adjusted Wasserstein gradient descent** [CLL+20], etc.

Sampling as Optimisation

This framework has been fruitful in the design and analysis of many existing sampling algorithms. For example:

- **Langevin Monte Carlo** (LMC): forward-flow discretisation of the gradient flow of the KL divergence with respect to the quadratic Wasserstein metric [Wib18, DMM19].
- **Stein Variational Gradient Descent** (SVGD): explicit Euler discretisation of the gradient flow of the KL divergence with respect to a kernelised Wasserstein metric [LW16, DNS23].
- Lots of other examples, e.g., **kernel Stein discrepancy descent** [KPCAF+21], **Laplacian adjusted Wasserstein gradient descent** [CLL+20], etc.

Sampling as Optimisation

This framework has been fruitful in the design and analysis of many existing sampling algorithms. For example:

- **Langevin Monte Carlo** (LMC): forward-flow discretisation of the gradient flow of the KL divergence with respect to the quadratic Wasserstein metric [Wib18, DMM19].
- **Stein Variational Gradient Descent** (SVGD): explicit Euler discretisation of the gradient flow of the KL divergence with respect to a kernelised Wasserstein metric [LW16, DNS23].
- Lots of other examples, e.g., **kernel Stein discrepancy descent** [KPCAF+21], **Laplacian adjusted Wasserstein gradient descent** [CLL+20], etc.

Sampling as Optimisation

This framework has been fruitful in the design and analysis of many existing sampling algorithms. For example:

- **Langevin Monte Carlo** (LMC): forward-flow discretisation of the gradient flow of the KL divergence with respect to the quadratic Wasserstein metric [Wib18, DMM19].
- **Stein Variational Gradient Descent** (SVGD): explicit Euler discretisation of the gradient flow of the KL divergence with respect to a kernelised Wasserstein metric [LW16, DNS23].
- Lots of other examples, e.g., **kernel Stein discrepancy descent** [KPCAF$^+$21], **Laplacian adjusted Wasserstein gradient descent** [CLL$^+$20], etc.

Learning Rates

One feature common to all of these approaches is the need to specify an appropriate **learning rate** or **step size** γ.

- It should be **sufficiently small** to ensure **convergence to the target measure**, or a close approximation thereof.
- It should also be **large enough** to ensure convergence within a **reasonable time period**.

Learning Rates

One feature common to all of these approaches is the need to specify an appropriate **learning rate** or **step size** γ.

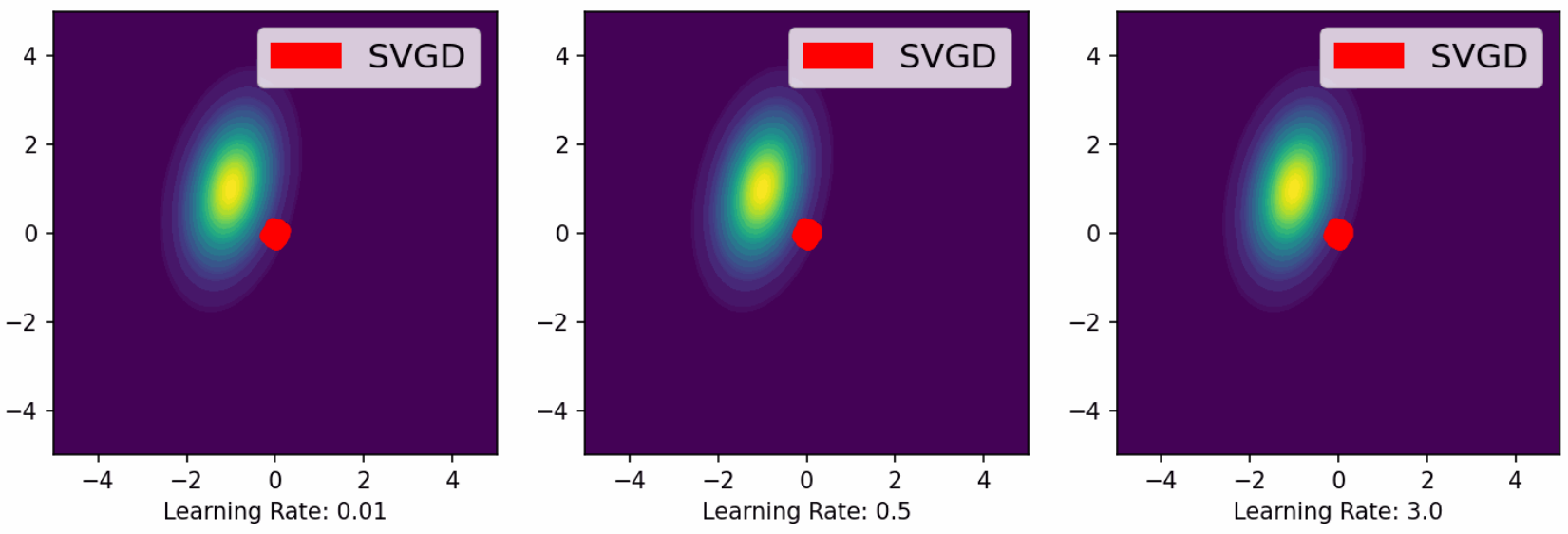
- It should be **sufficiently small** to ensure **convergence to the target measure**, or a close approximation thereof.
- It should also be **large enough** to ensure convergence within a **reasonable time period**.

Learning Rates

One feature common to all of these approaches is the need to specify an appropriate **learning rate** or **step size** γ.

- It should be **sufficiently small** to ensure **convergence to the target measure**, or a close approximation thereof.
- It should also be **large enough** to ensure convergence within a **reasonable time period**.

Learning Rates



Learning Rates

In theory, existing **non-asymptotic convergence rates** for LMC [Dal17, DM17, DM19] and SVGD [KSA$^+$20, SSP22, SR22] allow us to derive **optimal learning rates** for these algorithms.

- In practice, the optimal learning rate is a **function of the unknown target measure**, and so it can't actually be computed.

No More Learning Rates?

There are various existing approaches to selecting learning rates, none of which are entirely satisfactory:

- Run the algorithm of choice for **multiple learning rates**, and select the learning rate which **minimises an appropriately chosen metric**.
- Use ideas from the optimisation literature to design effective **step size schedules** [CCG+16, LCCC16, KSL22].

Can we obtain a gradient-based sampling algorithm which is entirely learning-rate free?

No More Learning Rates?

There are various existing approaches to selecting learning rates, none of which are entirely satisfactory:

- Run the algorithm of choice for **multiple learning rates**, and select the learning rate which **minimises an appropriately chosen metric**.
- Use ideas from the optimisation literature to design effective **step size schedules** [CCG+16, LCCC16, KSL22].

Can we obtain a gradient-based sampling algorithm which is entirely learning-rate free?

No More Learning Rates?

There are various existing approaches to selecting learning rates, none of which are entirely satisfactory:

- Run the algorithm of choice for **multiple learning rates**, and select the learning rate which **minimises an appropriately chosen metric**.
- Use ideas from the optimisation literature to design effective **step size schedules** [CCG+16, LCCC16, KSL22].

Can we obtain a gradient-based sampling algorithm which is entirely learning-rate free?

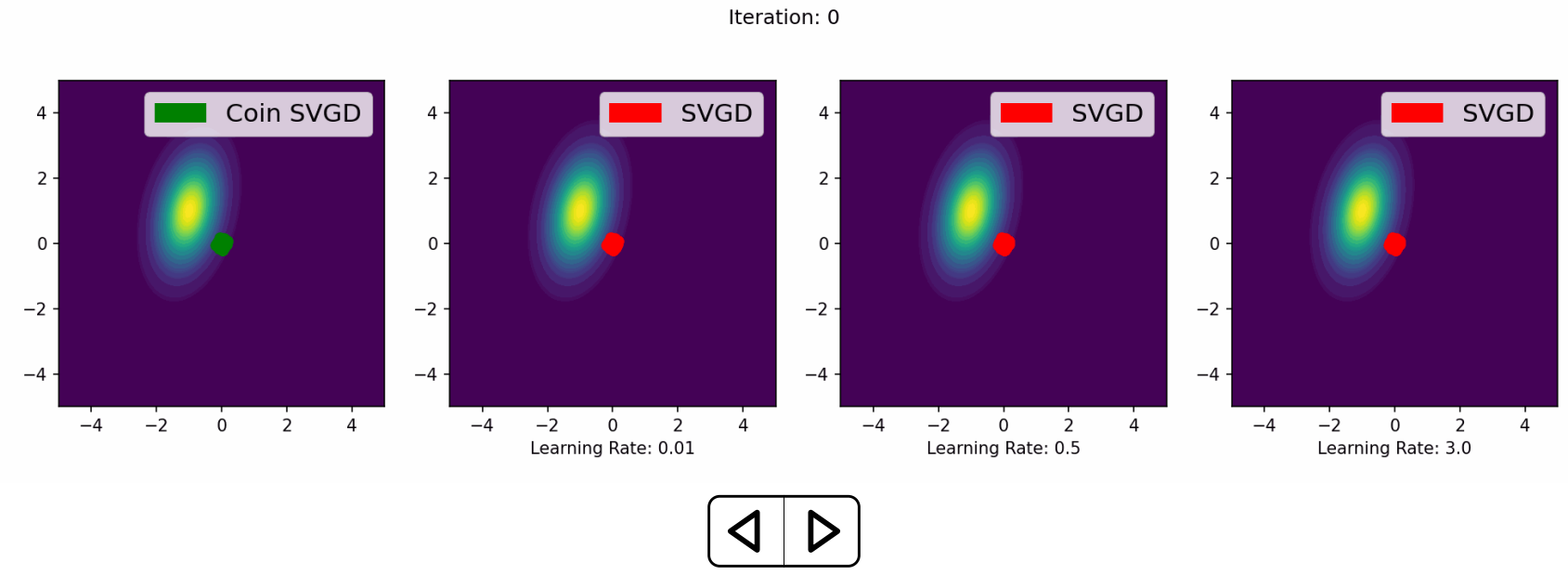
No More Learning Rates?

There are various existing approaches to selecting learning rates, none of which are entirely satisfactory:

- Run the algorithm of choice for **multiple learning rates**, and select the learning rate which **minimises an appropriately chosen metric**.
- Use ideas from the optimisation literature to design effective **step size schedules** [CCG+16, LCCC16, KSL22].

Can we obtain a gradient-based sampling algorithm which is entirely learning-rate free?

No More Learning Rates?



Optimisation in Euclidean Space

Preliminaries

Suppose that f is **convex**. We say that $g \in \mathbb{R}^d$ is a **subgradient** of f at x, and write $g \in \partial f(x)$ if, for any $z \in \mathbb{R}^d$,

$$f(z) - f(x) \geq \langle g, z - x \rangle$$

If f is **differentiable** at x, then the **differential set** $\partial f(x)$ contains only $\nabla f(x)$, the **gradient** of f at x.

We define the **convex conjugate** of f as the function $f^* : \mathbb{R}^d \to \mathbb{R}$ such that

$$f^*(u) = \sup_{x \in \mathcal{X}} [\langle u, x \rangle - f(x)]$$

Preliminaries

Suppose that f is **convex**. We say that $g \in \mathbb{R}^d$ is a **subgradient** of f at x, and write $g \in \partial f(x)$ if, for any $z \in \mathbb{R}^d$,

$$f(z) - f(x) \geq \langle g, z - x \rangle$$

If f is **differentiable** at x, then the **differential set** $\partial f(x)$ contains only $\nabla f(x)$, the **gradient** of f at x.

We define the **convex conjugate** of f as the function $f^* : \mathbb{R}^d \to \mathbb{R}$ such that

$$f^*(u) = \sup_{x \in \mathcal{X}} [\langle u, x \rangle - f(x)]$$

Preliminaries

Suppose that f is **convex**. We say that $g \in \mathbb{R}^d$ is a **subgradient** of f at x, and write $g \in \partial f(x)$ if, for any $z \in \mathbb{R}^d$,

$$f(z) - f(x) \geq \langle g, z - x \rangle$$

If f is **differentiable** at x, then the **differential set** $\partial f(x)$ contains only $\nabla f(x)$, the **gradient** of f at x.

We define the **convex conjugate** of f as the function $f^* : \mathbb{R}^d \to \mathbb{R}$ such that

$$f^*(u) = \sup_{x \in \mathcal{X}} [\langle u, x \rangle - f(x)]$$

Gradient Flows in Euclidean Space

Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex. Suppose we are interested in the following optimisation problem

$$x^* = \arg\min_{x \in \mathcal{X}} f(x).$$

We can solve this problem using the **gradient flow** of f, defined as the solution $x : [0, \infty) \to \mathbb{R}^d$ of the **differential inclusion**

$$\dot{x}_t \in -\partial f(x_t).$$

This inclusion admits a unique, absolutely continuous solution for almost all $t \geq 0$ [Bré73]. Moreover, the **function** $t \mapsto f(x_t)$ **is decreasing**, with $\lim_{t\to\infty} f(x_t) = \inf_{x\in\mathbb{R}^d} f(x)$ [PS10].

Gradient Flows in Euclidean Space

Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex. Suppose we are interested in the following optimisation problem

$$x^* = \arg\min_{x \in \mathcal{X}} f(x).$$

We can solve this problem using the **gradient flow** of f, defined as the solution $x : [0, \infty) \to \mathbb{R}^d$ of the **differential inclusion**

$$\dot{x}_t \in -\partial f(x_t).$$

This inclusion admits a unique, absolutely continuous solution for almost all $t \geq 0$ [Bré73]. Moreover, the **function** $t \mapsto f(x_t)$ **is decreasing**, with $\lim_{t\to\infty} f(x_t) = \inf_{x\in\mathbb{R}^d} f(x)$ [PS10].

Gradient Flows in Euclidean Space

Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex. Suppose we are interested in the following optimisation problem

$$x^* = \arg\min_{x \in \mathcal{X}} f(x).$$

We can solve this problem using the **gradient flow** of f, defined as the solution $x : [0, \infty) \to \mathbb{R}^d$ of the **differential inclusion**

$$\dot{x}_t \in -\partial f(x_t).$$

This inclusion admits a unique, absolutely continuous solution for almost all $t \geq 0$ [Bré73]. Moreover, the **function** $t \mapsto f(x_t)$ **is decreasing**, with $\lim_{t\to\infty} f(x_t) = \inf_{x\in\mathbb{R}^d} f(x)$ [PS10].

Gradient Descent in Euclidean Space

In practice, it is necessary to use a time-discretisation of this gradient flow.

- The **backward Euler discretisation**, which results in the proximal point algorithm [Gül91, De 93]

$$x_{t+1} \in \arg\min_{x \in \mathbb{R}^d} \left[f(x) + \frac{1}{2\gamma} ||x - x_t||^2 \right].$$

- The **forward Euler discretisation**, which results in the subgradient descent algorithm [Sho85]

$$x_{t+1} = x_t - \gamma g_t \ , \quad g_t \in \partial f(x_t).$$

The properties of these algorithms depend, of course, on the choice of learning rate $\gamma > 0$.

Gradient Descent in Euclidean Space

In practice, it is necessary to use a time-discretisation of this gradient flow.

- The **backward Euler discretisation**, which results in the proximal point algorithm [Gül91, De 93]

$$x_{t+1} \in \arg\min_{x \in \mathbb{R}^d} \left[f(x) + \frac{1}{2\gamma} ||x - x_t||^2 \right].$$

- The **forward Euler discretisation**, which results in the subgradient descent algorithm [Sho85]

$$x_{t+1} = x_t - \gamma g_t \ , \quad g_t \in \partial f(x_t).$$

The properties of these algorithms depend, of course, on the choice of learning rate $\gamma > 0$.

Gradient Descent in Euclidean Space

In practice, it is necessary to use a time-discretisation of this gradient flow.

- The **backward Euler discretisation**, which results in the proximal point algorithm [Gül91, De 93]

$$x_{t+1} \in \arg\min_{x \in \mathbb{R}^d} \left[f(x) + \frac{1}{2\gamma} ||x - x_t||^2 \right].$$

- The **forward Euler discretisation**, which results in the subgradient descent algorithm [Sho85]

$$x_{t+1} = x_t - \gamma g_t \ , \quad g_t \in \partial f(x_t).$$

The properties of these algorithms depend, of course, on the choice of learning rate $\gamma > 0$.

Gradient Descent in Euclidean Space

In practice, it is necessary to use a time-discretisation of this gradient flow.

- The **backward Euler discretisation**, which results in the proximal point algorithm [Gül91, De 93]

$$x_{t+1} \in \arg\min_{x \in \mathbb{R}^d} \left[f(x) + \frac{1}{2\gamma} ||x - x_t||^2 \right].$$

- The **forward Euler discretisation**, which results in the subgradient descent algorithm [Sho85]

$$x_{t+1} = x_t - \gamma g_t \ , \quad g_t \in \partial f(x_t).$$

The properties of these algorithms depend, of course, on the choice of learning rate $\gamma > 0$.

Optimal Learning Rates

Given an L-Lipschitz function, one can show that the iterates of the subgradient descent algorithm satisfy [Zin03]

$$f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - f(x^*) \leq \frac{1}{T}\left[\frac{||x_1 - x^*||^2}{2\gamma} + \frac{L^2 T\gamma}{2}\right].$$

This implies that the optimal learning rate is $\gamma_{\text{opt}} = \frac{||x_1 - x^*||}{L\sqrt{T}}$, which yields the optimal error bound

$$f(\bar{x}_T) - f(x^*) \leq \frac{L||x_1 - x^*||}{\sqrt{T}}.$$

In practice, however, it is **not possible to achieve this bound**. Even in hindsight, one **cannot compute** the optimal learning rate γ_{ideal}, since it depends on the unknown $||x_1 - x^*||$.

Optimal Learning Rates

Given an L-Lipschitz function, one can show that the iterates of the subgradient descent algorithm satisfy [Zin03]

$$f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - f(x^*) \leq \frac{1}{T}\left[\frac{||x_1 - x^*||^2}{2\gamma} + \frac{L^2 T\gamma}{2}\right].$$

This implies that the optimal learning rate is $\gamma_{\text{opt}} = \frac{||x_1 - x^*||}{L\sqrt{T}}$, which yields the optimal error bound

$$f(\bar{x}_T) - f(x^*) \leq \frac{L||x_1 - x^*||}{\sqrt{T}}.$$

In practice, however, it is **not possible to achieve this bound**. Even in hindsight, one **cannot compute** the optimal learning rate γ_{ideal}, since it depends on the unknown $||x_1 - x^*||$.

Optimal Learning Rates

Given an L-Lipschitz function, one can show that the iterates of the subgradient descent algorithm satisfy [Zin03]

$$f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - f(x^*) \leq \frac{1}{T}\left[\frac{||x_1 - x^*||^2}{2\gamma} + \frac{L^2 T\gamma}{2}\right].$$

This implies that the optimal learning rate is $\gamma_{\text{opt}} = \frac{||x_1 - x^*||}{L\sqrt{T}}$, which yields the optimal error bound

$$f(\bar{x}_T) - f(x^*) \leq \frac{L||x_1 - x^*||}{\sqrt{T}}.$$

In practice, however, it is **not possible to achieve this bound**. Even in hindsight, one **cannot compute** the optimal learning rate γ_{ideal}, since it depends on the unknown $||x_1 - x^*||$.

Coin Betting

Consider a gambler who bets on the outcomes of a series of adversarial coin flips.

The gambler starts with an initial wealth $w_0 = \varepsilon > 0$. In the t^{th} round, the gambler **bets on the outcome of a coin flip** $c_t \in \{-1, 1\}$, where $+1$ denotes heads and -1 denotes tails.

We will encode the gambler's bets in the t^{th} round by $x_t \in \mathbb{R}$. In particular, $\operatorname{sign}(x_t) \in \{-1, 1\}$ will denote **whether the bet is on heads or tails**, and $|x_t| \in \mathbb{R}$ will denote the **size of the bet**.

Thus, in the t^{th} round, the gambler **wins** $|x_t c_t| = x_t c_t$ if $\operatorname{sign}(c_t) = \operatorname{sign}(x_t)$, and **loses** $|x_t c_t| = -x_t c_t$ otherwise.

Coin Betting

Consider a gambler who bets on the outcomes of a series of adversarial coin flips.

The gambler starts with an initial wealth $w_0 = \varepsilon > 0$. In the t^{th} round, the gambler **bets on the outcome of a coin flip** $c_t \in \{-1, 1\}$, where $+1$ denotes heads and -1 denotes tails.

We will encode the gambler's bets in the t^{th} round by $x_t \in \mathbb{R}$. In particular, $\operatorname{sign}(x_t) \in \{-1, 1\}$ will denote **whether the bet is on heads or tails**, and $|x_t| \in \mathbb{R}$ will denote the **size of the bet**.

Thus, in the t^{th} round, the gambler **wins** $|x_t c_t| = x_t c_t$ if $\operatorname{sign}(c_t) = \operatorname{sign}(x_t)$, and **loses** $|x_t c_t| = -x_t c_t$ otherwise.

Coin Betting

Consider a gambler who bets on the outcomes of a series of adversarial coin flips.

The gambler starts with an initial wealth $w_0 = \varepsilon > 0$. In the t^{th} round, the gambler **bets on the outcome of a coin flip** $c_t \in \{-1, 1\}$, where $+1$ denotes heads and -1 denotes tails.

We will encode the gambler's bets in the t^{th} round by $x_t \in \mathbb{R}$. In particular, $\operatorname{sign}(x_t) \in \{-1, 1\}$ will denote **whether the bet is on heads or tails**, and $|x_t| \in \mathbb{R}$ will denote the **size of the bet**.

Thus, in the t^{th} round, the gambler **wins** $|x_t c_t| = x_t c_t$ if $\operatorname{sign}(c_t) = \operatorname{sign}(x_t)$, and **loses** $|x_t c_t| = -x_t c_t$ otherwise.

Coin Betting

Consider a gambler who bets on the outcomes of a series of adversarial coin flips.

The gambler starts with an initial wealth $w_0 = \varepsilon > 0$. In the t^{th} round, the gambler **bets on the outcome of a coin flip** $c_t \in \{-1, 1\}$, where $+1$ denotes heads and -1 denotes tails.

We will encode the gambler's bets in the t^{th} round by $x_t \in \mathbb{R}$. In particular, $\operatorname{sign}(x_t) \in \{-1, 1\}$ will denote **whether the bet is on heads or tails**, and $|x_t| \in \mathbb{R}$ will denote the **size of the bet**.

Thus, in the t^{th} round, the gambler **wins** $|x_t c_t| = x_t c_t$ if $\operatorname{sign}(c_t) = \operatorname{sign}(x_t)$, and **loses** $|x_t c_t| = -x_t c_t$ otherwise.

Coin Betting

Suppose that we write w_t for the wealth of the gambler at the end of the t^{th} round. Clearly, we then have that

$$w_t = \varepsilon + \sum_{i=1}^{t} c_i x_i.$$

We will impose the additional restriction that the gambler's bets satisfy $x_t = \beta_t w_{t-1}$, for some **betting fraction** $\beta_t \in [-1, 1]$. This is equivalent to the assumption that the gambler **cannot borrow any money**.

Coin Betting

Suppose that we write w_t for the wealth of the gambler at the end of the t^{th} round. Clearly, we then have that

$$w_t = \varepsilon + \sum_{i=1}^{t} c_i x_i.$$

We will impose the additional restriction that the gambler's bets satisfy $x_t = \beta_t w_{t-1}$, for some **betting fraction** $\beta_t \in [-1, 1]$. This is equivalent to the assumption that the gambler **cannot borrow any money**.

Coin Betting

How should we choose the betting fraction $\beta_t \in [-1, 1]$?

- If we **knew the probability of heads** p, then the optimal strategy is given by $\beta_t = \mathbb{E}[c_t] = 2p - 1$ [Kel56].
- If we **knew the future outcomes**, then the optimal **fixed fraction** is given by $\beta_t = T^{-1} \sum_{i=1}^{T} c_i$. This guarantees the wealth increases exponentially.
- If we **don't know the future outcomes**, then the **KT estimator** yields the betting strategy $\beta_t = t^{-1} \sum_{i=1}^{t-1} c_i$. This guarantees almost the same wealth that one would obtain knowing in advance the fraction of heads [KT81].

Coin Betting

How should we choose the betting fraction $\beta_t \in [-1, 1]$?

- If we **knew the probability of heads** p, then the optimal strategy is given by $\beta_t = \mathbb{E}[c_t] = 2p - 1$ [Kel56].
- If we **knew the future outcomes**, then the optimal **fixed fraction** is given by $\beta_t = T^{-1} \sum_{i=1}^{T} c_i$. This guarantees the wealth increases exponentially.
- If we **don't know the future outcomes**, then the **KT estimator** yields the betting strategy $\beta_t = t^{-1} \sum_{i=1}^{t-1} c_i$. This guarantees almost the same wealth that one would obtain knowing in advance the fraction of heads [KT81].

Coin Betting

How should we choose the betting fraction $\beta_t \in [-1, 1]$?

- If we **knew the probability of heads** p, then the optimal strategy is given by $\beta_t = \mathbb{E}[c_t] = 2p - 1$ [Kel56].
- If we **knew the future outcomes**, then the optimal **fixed fraction** is given by $\beta_t = T^{-1} \sum_{i=1}^{T} c_i$. This guarantees the wealth increases exponentially.
- If we **don't know the future outcomes**, then the **KT estimator** yields the betting strategy $\beta_t = t^{-1} \sum_{i=1}^{t-1} c_i$. This guarantees almost the same wealth that one would obtain knowing in advance the fraction of heads [KT81].

Coin Betting

How should we choose the betting fraction $\beta_t \in [-1, 1]$?

- If we **knew the probability of heads** p, then the optimal strategy is given by $\beta_t = \mathbb{E}[c_t] = 2p - 1$ [Kel56].
- If we **knew the future outcomes**, then the optimal **fixed fraction** is given by $\beta_t = T^{-1} \sum_{i=1}^{T} c_i$. This guarantees the wealth increases exponentially.
- If we **don't know the future outcomes**, then the **KT estimator** yields the betting strategy $\beta_t = t^{-1} \sum_{i=1}^{t-1} c_i$. This guarantees almost the same wealth that one would obtain knowing in advance the fraction of heads [KT81].

Coin Betting

Recalling our previous formula for the wealth, we see that this betting fraction yields the **sequence of bets**

$$x_t = \beta_t w_{t-1} = \frac{\sum_{i=1}^{t-1} c_i}{t} \left(\varepsilon + \sum_{i=1}^{t-1} c_i x_i \right).$$

One can show that this betting strategy guarantees that the wealth **grows exponentially** [OP16]

$$w_T \geq h\left(\sum_{t=1}^{T} c_t \right) := \frac{\varepsilon}{K\sqrt{T}} \exp\left(\frac{\left(\sum_{t=1}^{T} c_t \right)^2}{2T} \right).$$

In fact, this holds for **continuous outcomes** $c_t \in [-1, 1]$, and not just for $c_t \in \{-1, 1\}$.

Coin Betting

Recalling our previous formula for the wealth, we see that this betting fraction yields the **sequence of bets**

$$x_t = \beta_t w_{t-1} = \frac{\sum_{i=1}^{t-1} c_i}{t} \left(\varepsilon + \sum_{i=1}^{t-1} c_i x_i \right).$$

One can show that this betting strategy guarantees that the wealth **grows exponentially** [OP16]

$$w_T \geq h\left(\sum_{t=1}^{T} c_t \right) := \frac{\varepsilon}{K\sqrt{T}} \exp\left(\frac{\left(\sum_{t=1}^{T} c_t \right)^2}{2T} \right).$$

In fact, this holds for **continuous outcomes** $c_t \in [-1, 1]$, and not just for $c_t \in \{-1, 1\}$.

Coin Betting

Recalling our previous formula for the wealth, we see that this betting fraction yields the **sequence of bets**

$$x_t = \beta_t w_{t-1} = \frac{\sum_{i=1}^{t-1} c_i}{t} \left(\varepsilon + \sum_{i=1}^{t-1} c_i x_i \right).$$

One can show that this betting strategy guarantees that the wealth **grows exponentially** [OP16]

$$w_T \geq h\left(\sum_{t=1}^{T} c_t \right) := \frac{\varepsilon}{K\sqrt{T}} \exp\left(\frac{\left(\sum_{t=1}^{T} c_t \right)^2}{2T} \right).$$

In fact, this holds for **continuous outcomes** $c_t \in [-1, 1]$, and not just for $c_t \in \{-1, 1\}$.

Coin Betting and Parameter Free Optimisation

What is the connection between **convex optimisation** and **coin betting**?

- Let $f : \mathbb{R} \to \mathbb{R}$ be a convex function, with $\nabla f(x) \in [-1, 1]$ for all $x \in \mathbb{R}$.
- Consider a betting game where, in the t^{th} round, we bet x_t on $c_t = -\nabla f(x_t) \in [-1, 1]$ according to our strategy from the last slide. Thus

$$x_t = -\frac{\sum_{i=1}^{t-1} \nabla f(x_i)}{t} \left(\varepsilon - \sum_{i=1}^{t-1} \nabla f(x_i) x_i \right).$$

- Remarkably, we can show that the **average of the bets converges to** $x^* = \arg\min_{x \in \mathbb{R}} f(x)$ at a rate governed by our betting strategy.

Coin Betting and Parameter Free Optimisation

What is the connection between **convex optimisation** and **coin betting**?

- Let $f : \mathbb{R} \to \mathbb{R}$ be a convex function, with $\nabla f(x) \in [-1, 1]$ for all $x \in \mathbb{R}$.
- Consider a betting game where, in the t^{th} round, we bet x_t on $c_t = -\nabla f(x_t) \in [-1, 1]$ according to our strategy from the last slide. Thus

$$x_t = -\frac{\sum_{i=1}^{t-1} \nabla f(x_i)}{t} \left(\varepsilon - \sum_{i=1}^{t-1} \nabla f(x_i) x_i \right).$$

- Remarkably, we can show that the **average of the bets converges to** $x^* = \arg\min_{x \in \mathbb{R}} f(x)$ at a rate governed by our betting strategy.

Coin Betting and Parameter Free Optimisation

What is the connection between **convex optimisation** and **coin betting**?

- Let $f : \mathbb{R} \to \mathbb{R}$ be a convex function, with $\nabla f(x) \in [-1, 1]$ for all $x \in \mathbb{R}$.
- Consider a betting game where, in the t^{th} round, we bet x_t on $c_t = -\nabla f(x_t) \in [-1, 1]$ according to our strategy from the last slide. Thus

$$x_t = -\frac{\sum_{i=1}^{t-1} \nabla f(x_i)}{t} \left(\varepsilon - \sum_{i=1}^{t-1} \nabla f(x_i) x_i \right).$$

- Remarkably, we can show that the **average of the bets converges to** $x^* = \arg\min_{x \in \mathbb{R}} f(x)$ at a rate governed by our betting strategy.

Coin Betting and Parameter Free Optimisation

What is the connection between **convex optimisation** and **coin betting**?

- Let $f : \mathbb{R} \to \mathbb{R}$ be a convex function, with $\nabla f(x) \in [-1, 1]$ for all $x \in \mathbb{R}$.
- Consider a betting game where, in the t^{th} round, we bet x_t on $c_t = -\nabla f(x_t) \in [-1, 1]$ according to our strategy from the last slide. Thus

$$x_t = -\frac{\sum_{i=1}^{t-1} \nabla f(x_i)}{t} \left(\varepsilon - \sum_{i=1}^{t-1} \nabla f(x_i) x_i \right).$$

- Remarkably, we can show that the **average of the bets converges to** $x^* = \arg\min_{x \in \mathbb{R}} f(x)$ at a rate governed by our betting strategy.

Coin Betting and Parameter Free Optimisation

The essential steps are as follows [OP16]

$$f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - f(x^*) \leq \frac{1}{T}\sum_{t=1}^{T}\left(f(x_t) - f(x^*)\right) \qquad \text{(Jensen's inequality)}$$

$$\leq \frac{1}{T}\left(\sum_{t=1}^{T} c_t x^* - \sum_{t=1}^{T} c_t x_t\right) \qquad \text{(convexity)}$$

$$\leq \frac{1}{T}\left(\left(\sum_{t=1}^{T} c_t\right) x^* - h\left(\sum_{t=1}^{T} c_t\right) + \varepsilon\right) \qquad \text{(bound on } w_T\text{)}$$

$$\leq \frac{1}{T}\left(\max_{v}\left[v x^* - h(v)\right] + \varepsilon\right) \qquad \text{(max over } v = \textstyle\sum_{t=1}^{T} c_t\text{)}$$

$$= \frac{h^*(x^*) + \varepsilon}{T}. \qquad \text{(definition of convex conjugate)}$$

Coin Betting and Parameter Free Optimisation

The essential steps are as follows [OP16]

$$f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - f(x^*) \leq \frac{1}{T}\sum_{t=1}^{T}\left(f(x_t) - f(x^*)\right) \qquad \text{(Jensen's inequality)}$$

$$\leq \frac{1}{T}\left(\sum_{t=1}^{T} c_t x^* - \sum_{t=1}^{T} c_t x_t\right) \qquad \text{(convexity)}$$

$$\leq \frac{1}{T}\left(\left(\sum_{t=1}^{T} c_t\right) x^* - h\left(\sum_{t=1}^{T} c_t\right) + \varepsilon\right) \qquad \text{(bound on } w_T\text{)}$$

$$\leq \frac{1}{T}\left(\max_{v}\left[v x^* - h(v)\right] + \varepsilon\right) \qquad \text{(max over } v = \textstyle\sum_{t=1}^{T} c_t\text{)}$$

$$= \frac{h^*(x^*) + \varepsilon}{T}. \qquad \text{(definition of convex conjugate)}$$

Coin Betting and Parameter Free Optimisation

The essential steps are as follows [OP16]

$$
\begin{aligned}
f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - f(x^*) &\leq \frac{1}{T}\sum_{t=1}^{T}\left(f(x_t) - f(x^*)\right) && \text{(Jensen's inequality)} \\
&\leq \frac{1}{T}\left(\sum_{t=1}^{T} c_t x^* - \sum_{t=1}^{T} c_t x_t\right) && \text{(convexity)} \\
&\leq \frac{1}{T}\left(\left(\sum_{t=1}^{T} c_t\right) x^* - h\left(\sum_{t=1}^{T} c_t\right) + \varepsilon\right) && \text{(bound on } w_T\text{)} \\
&\leq \frac{1}{T}\left(\max_{v}\left[v x^* - h(v)\right] + \varepsilon\right) && \text{(max over } v = \textstyle\sum_{t=1}^{T} c_t\text{)} \\
&= \frac{h^*(x^*) + \varepsilon}{T}. && \text{(definition of convex conjugate)}
\end{aligned}
$$

Coin Betting and Parameter Free Optimisation

The essential steps are as follows [OP16]

$$
\begin{aligned}
f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - f(x^*) &\leq \frac{1}{T}\sum_{t=1}^{T}\left(f(x_t) - f(x^*)\right) && \text{(Jensen's inequality)} \\
&\leq \frac{1}{T}\left(\sum_{t=1}^{T} c_t x^* - \sum_{t=1}^{T} c_t x_t\right) && \text{(convexity)} \\
&\leq \frac{1}{T}\left(\left(\sum_{t=1}^{T} c_t\right) x^* - h\left(\sum_{t=1}^{T} c_t\right) + \varepsilon\right) && \text{(bound on } w_T\text{)} \\
&\leq \frac{1}{T}\left(\max_{v}\left[v x^* - h(v)\right] + \varepsilon\right) && \text{(max over } v = \textstyle\sum_{t=1}^{T} c_t\text{)} \\
&= \frac{h^*(x^*) + \varepsilon}{T}. && \text{(definition of convex conjugate)}
\end{aligned}
$$

Coin Betting and Parameter Free Optimisation

The essential steps are as follows [OP16]

$$
\begin{aligned}
f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - f(x^*) &\leq \frac{1}{T}\sum_{t=1}^{T}\left(f(x_t) - f(x^*)\right) && \text{(Jensen's inequality)} \\
&\leq \frac{1}{T}\left(\sum_{t=1}^{T} c_t x^* - \sum_{t=1}^{T} c_t x_t\right) && \text{(convexity)} \\
&\leq \frac{1}{T}\left(\left(\sum_{t=1}^{T} c_t\right) x^* - h\left(\sum_{t=1}^{T} c_t\right) + \varepsilon\right) && \text{(bound on } w_T\text{)} \\
&\leq \frac{1}{T}\left(\max_{v}\left[v x^* - h(v)\right] + \varepsilon\right) && \text{(max over } v = \textstyle\sum_{t=1}^{T} c_t\text{)} \\
&= \frac{h^*(x^*) + \varepsilon}{T}. && \text{(definition of convex conjugate)}
\end{aligned}
$$

Coin Betting and Parameter Free Optimisation

By substituting an appropriate bound for the convex conjugate h^*, this result yields the following **convergence rate** [OP16]

$$f(\bar{x}_T) - f(x^*) \leq K \frac{||x^*|| \sqrt{\log(1 + \frac{24T^2||x^*||^2}{\varepsilon^2}) + \varepsilon}}{\sqrt{T}}.$$

We can compare this with the convergence rate of subgradient descent with the **unknown optimal learning rate**:

$$f(\bar{x}_T) - f(x^*) \leq \frac{||x_1 - x^*||}{\sqrt{T}}.$$

Coin Betting and Parameter Free Optimisation

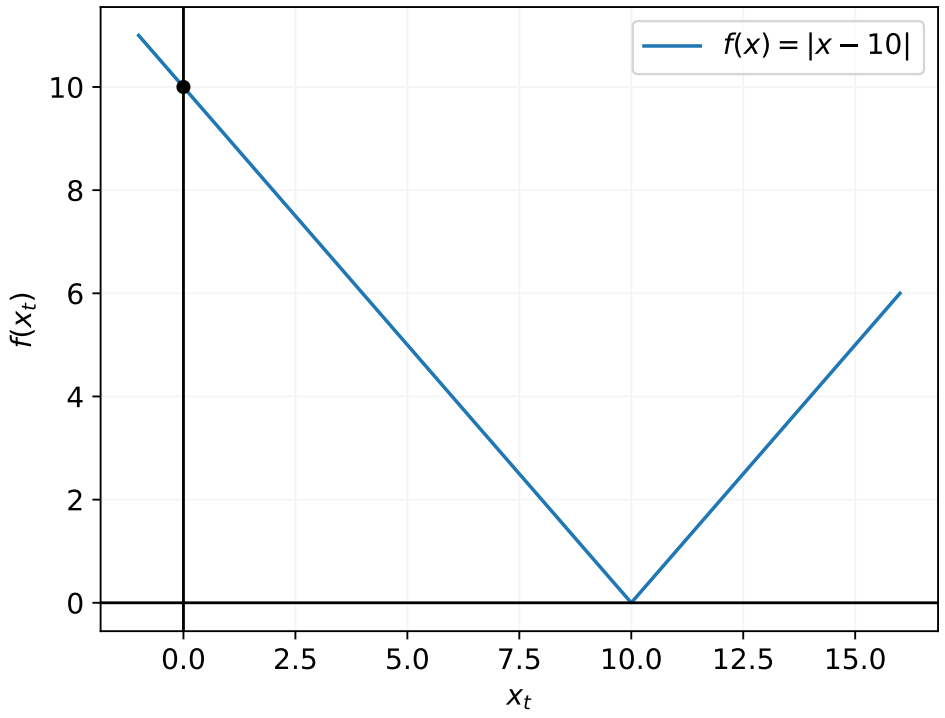
By substituting an appropriate bound for the convex conjugate h^*, this result yields the following **convergence rate** [OP16]

$$f(\bar{x}_T) - f(x^*) \leq K \frac{||x^*|| \sqrt{\log(1 + \frac{24T^2||x^*||^2}{\varepsilon^2}) + \varepsilon}}{\sqrt{T}}.$$

We can compare this with the convergence rate of subgradient descent with the **unknown optimal learning rate**:

$$f(\bar{x}_T) - f(x^*) \leq \frac{||x_1 - x^*||}{\sqrt{T}}.$$

Coin Betting and Parameter Free Optimisation



Initial wealth.

$w_0 = 1$

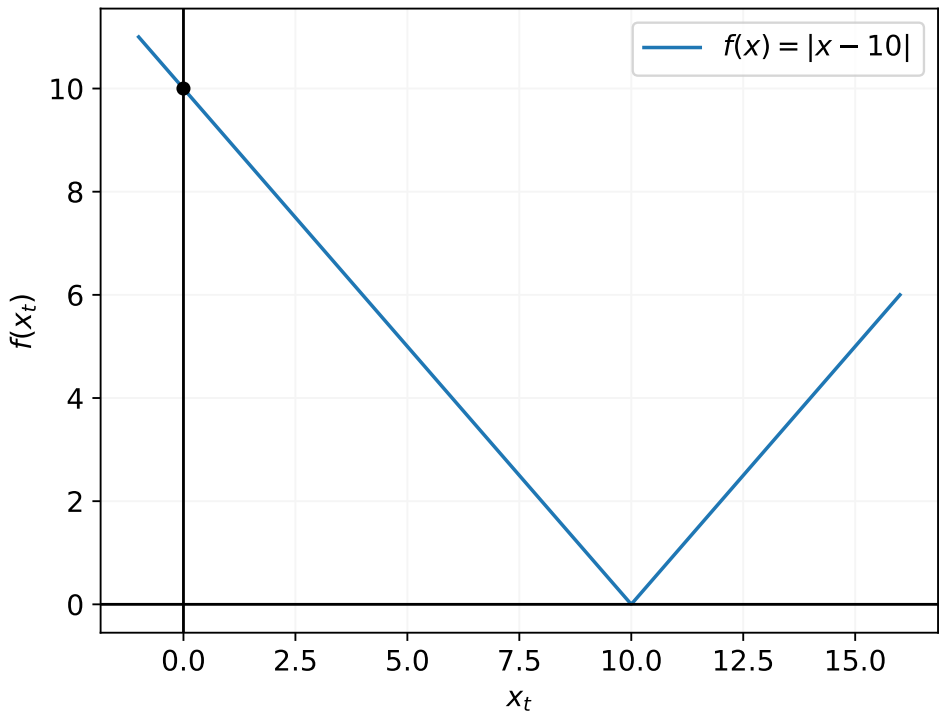
Betting fraction.

$$\beta_1 = \frac{\sum_{i=1}^{0} c_i}{1} = 0$$

Initial bet.

$x_1 = \beta_1 w_0 = 0.$

Coin Betting and Parameter Free Optimisation



Current wealth.

$w_0 = 1$

Current bet.

$x_1 = 0$

Outcome.

$c_1 = -\nabla f(x_1) = 1$

New wealth.

$w_1 = w_0 + c_1 x_1 = 1$

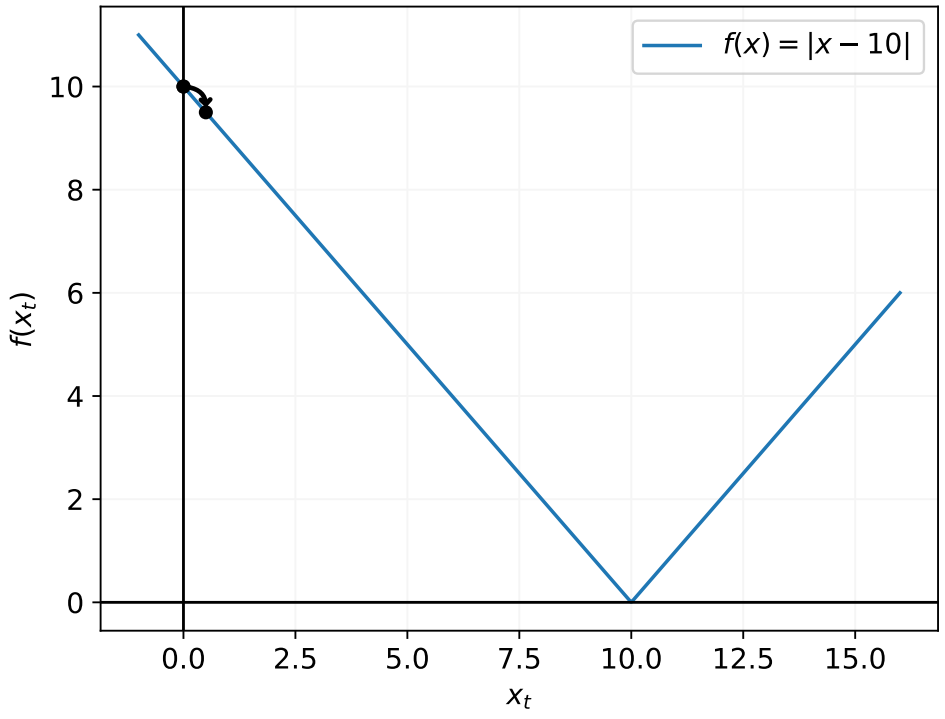
New betting fraction.

$$\beta_2 = \frac{\sum_{i=1}^{1} c_i}{2} = 0.5$$

New bet.

$x_2 = \beta_2 w_1 = 0.5.$

Coin Betting and Parameter Free Optimisation



Current wealth.

$w_0 = 1$

Current bet.

$x_1 = 0$

Outcome.

$c_1 = -\nabla f(x_1) = 1$

New wealth.

$w_1 = w_0 + c_1 x_1 = 1$

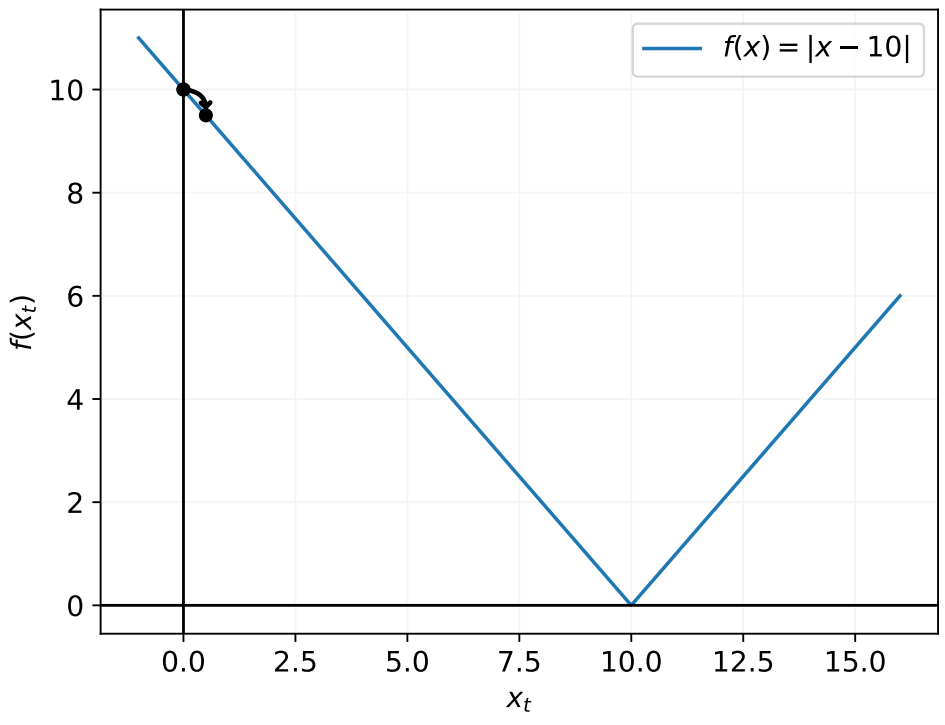
New betting fraction.

$$\beta_2 = \frac{\sum_{i=1}^{1} c_i}{2} = 0.5$$

New bet.

$x_2 = \beta_2 w_1 = 0.5.$

Coin Betting and Parameter Free Optimisation



Current wealth.

$w_1 = 1$

Current bet.

$x_2 = 0.5$

Outcome.

$c_2 = -\nabla f(x_2) = 1$

New wealth.

$w_2 = w_1 + c_2 x_2 = 1.5$

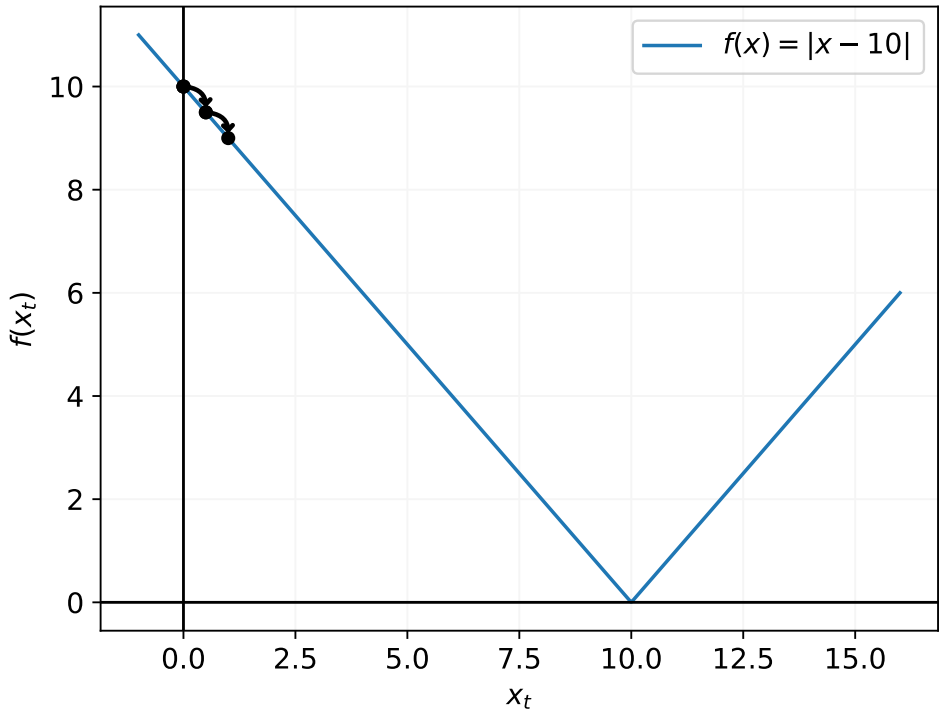
New betting fraction.

$$\beta_3 = \frac{\sum_{i=1}^{2} c_i}{3} = 0.\dot{6}$$

New bet.

$x_3 = \beta_3 w_2 = 1.$

Coin Betting and Parameter Free Optimisation



Current wealth.

$w_1 = 1$

Current bet.

$x_2 = 0.5$

Outcome.

$c_2 = -\nabla f(x_2) = 1$

New wealth.

$w_2 = w_1 + c_2 x_2 = 1.5$

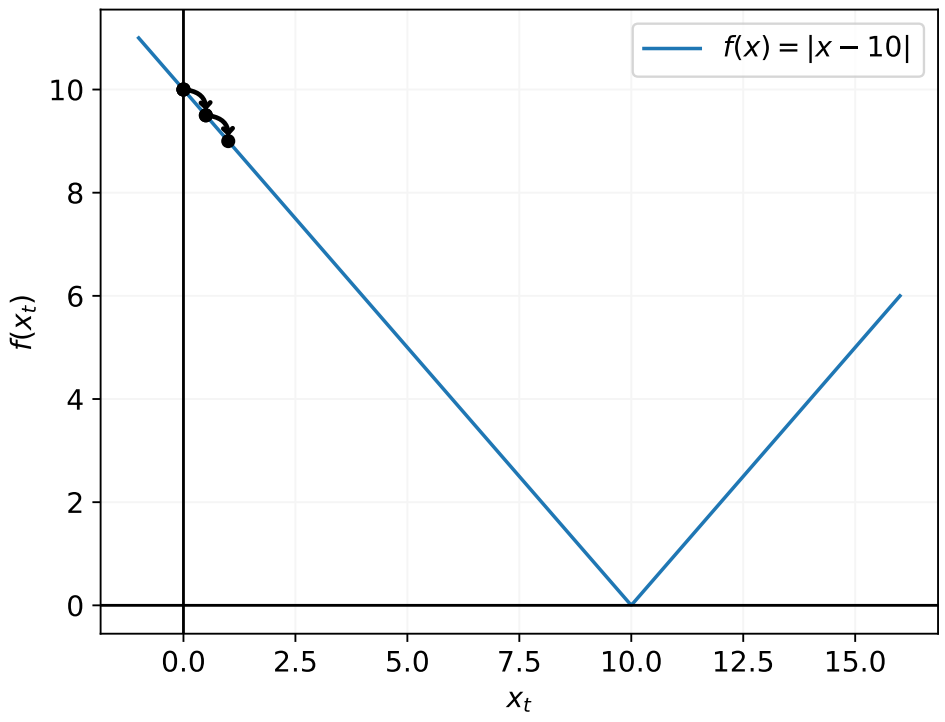
New betting fraction.

$$\beta_3 = \frac{\sum_{i=1}^{2} c_i}{3} = 0.\dot{6}$$

New bet.

$x_3 = \beta_3 w_2 = 1.$

Coin Betting and Parameter Free Optimisation



Current wealth.

$w_2 = 1.5$

Current bet.

$x_3 = 1$

Outcome.

$c_3 = -\nabla f(x_3) = 1$

New wealth.

$w_3 = w_2 + c_3 x_3 = 2.5$

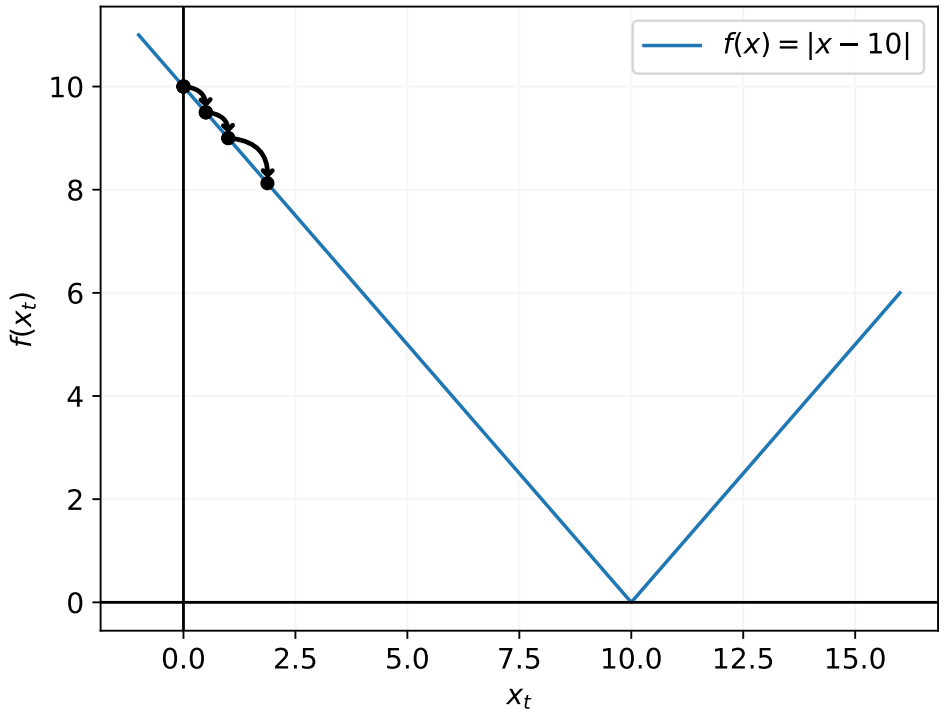
New betting fraction.

$$\beta_4 = \frac{\sum_{i=1}^{3} c_i}{4} = 0.75$$

New bet.

$x_4 = \beta_4 w_3 = 1.875$

Coin Betting and Parameter Free Optimisation



Current wealth.

$w_2 = 1.5$

Current bet.

$x_3 = 1$

Outcome.

$c_3 = -\nabla f(x_3) = 1$

New wealth.

$w_3 = w_2 + c_3 x_3 = 2.5$

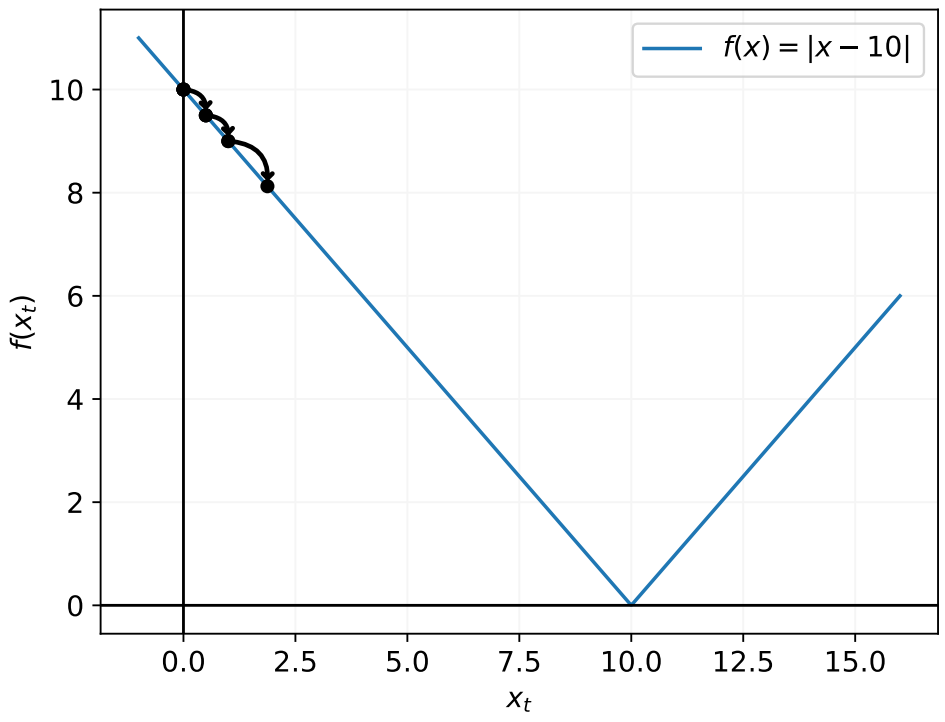
New betting fraction.

$$\beta_4 = \frac{\sum_{i=1}^{3} c_i}{4} = 0.75$$

New bet.

$x_4 = \beta_4 w_3 = 1.875$

Coin Betting and Parameter Free Optimisation



Current wealth.

$w_3 = 2.5$

Current bet.

$x_4 = 1.875$

Outcome.

$c_4 = -\nabla f(x_4) = 1$

New wealth.

$w_4 = w_3 + c_4 x_4 = 4.375$

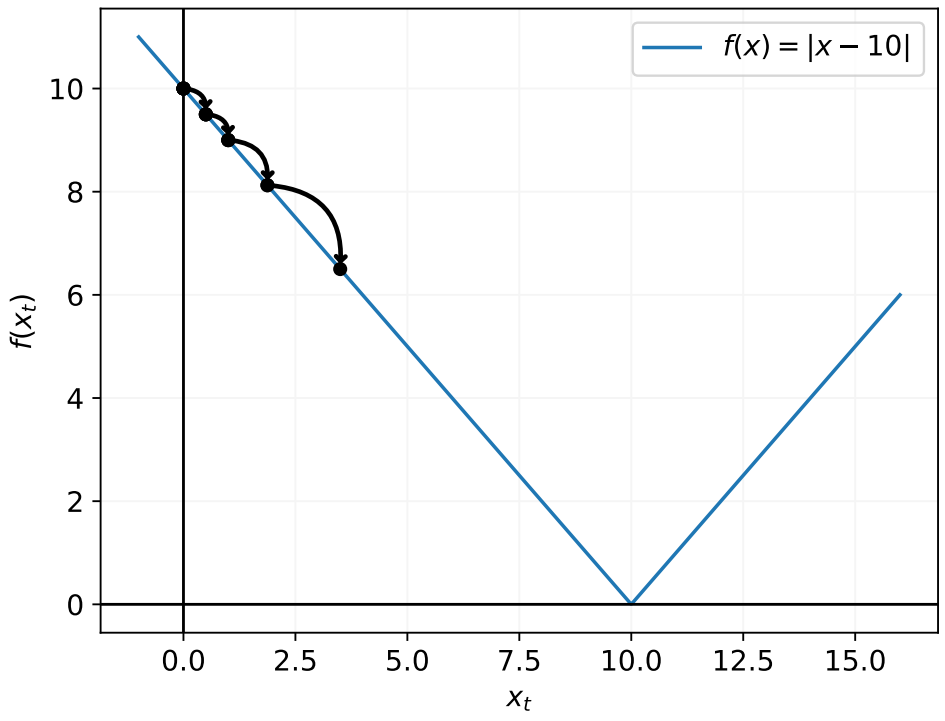
New betting fraction.

$$\beta_5 = \frac{\sum_{i=1}^{4} c_i}{5} = 0.8$$

New bet.

$x_5 = \beta_5 w_4 = 3.5$

Coin Betting and Parameter Free Optimisation



Current wealth.

$w_3 = 2.5$

Current bet.

$x_4 = 1.875$

Outcome.

$c_4 = -\nabla f(x_4) = 1$

New wealth.

$w_4 = w_3 + c_4 x_4 = 4.375$

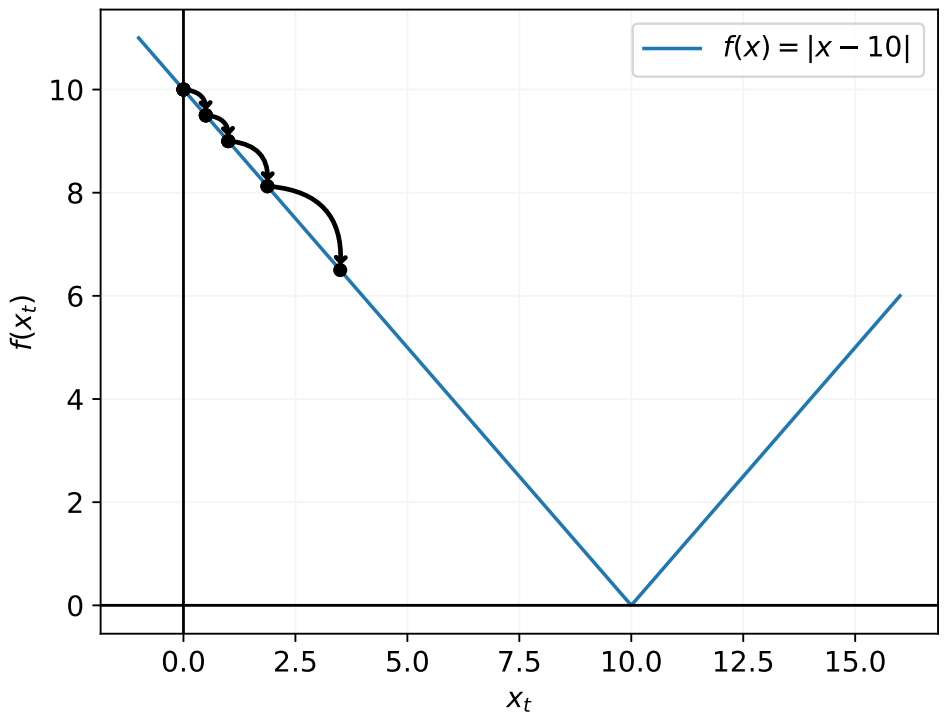
New betting fraction.

$$\beta_5 = \frac{\sum_{i=1}^{4} c_i}{5} = 0.8$$

New bet.

$x_5 = \beta_5 w_4 = 3.5$

Coin Betting and Parameter Free Optimisation



Current wealth.

$w_4 = 4.375$

Current bet.

$x_5 = 3.5$

Outcome.

$c_5 = -\nabla f(x_5) = 1$

New wealth.

$w_5 = w_4 + c_5 x_5 = 7.875$

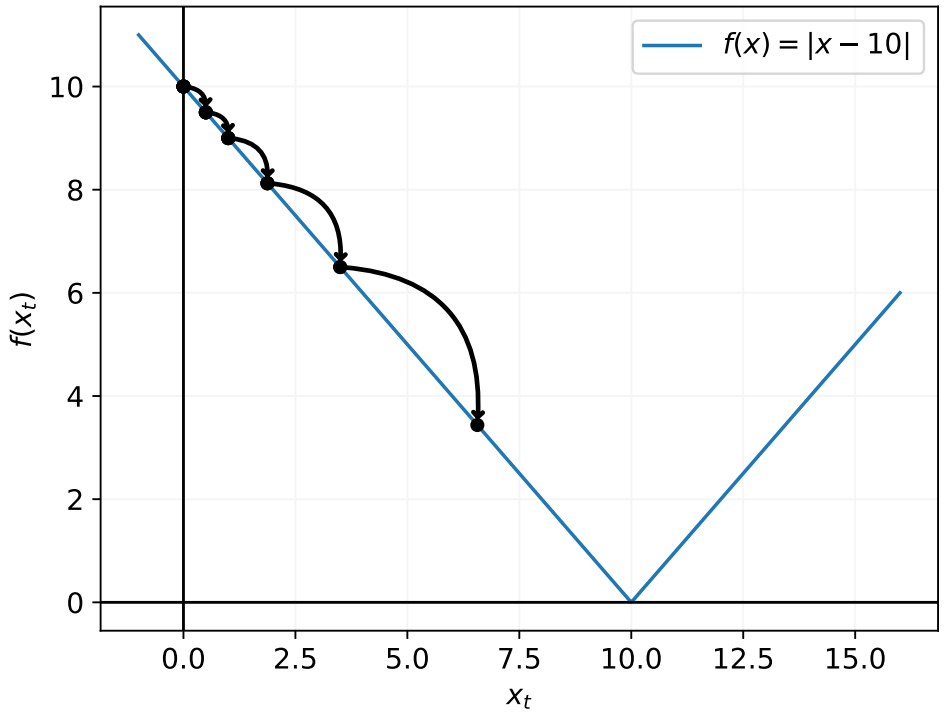
New betting fraction.

$$\beta_6 = \frac{\sum_{i=1}^{5} c_i}{6} = 0.8\dot{3}$$

New bet.

$x_6 = \beta_6 w_5 = 6.5625$

Coin Betting and Parameter Free Optimisation



Current wealth.

$w_4 = 4.375$

Current bet.

$x_5 = 3.5$

Outcome.

$c_5 = -\nabla f(x_5) = 1$

New wealth.

$w_5 = w_4 + c_5 x_5 = 7.875$

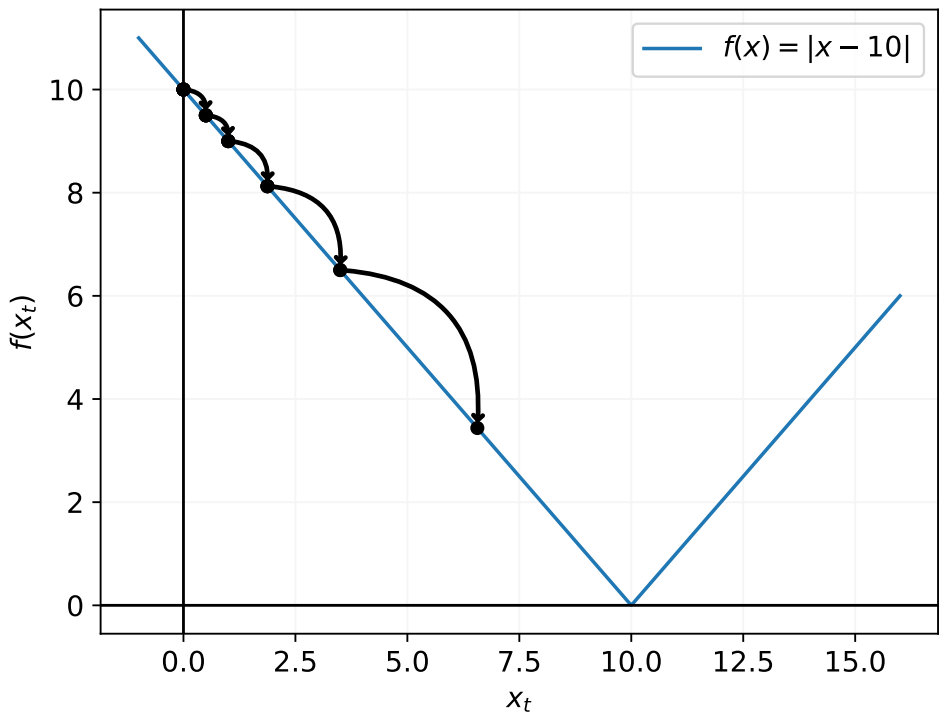
New betting fraction.

$$\beta_6 = \frac{\sum_{i=1}^{5} c_i}{6} = 0.8\dot{3}$$

New bet.

$x_6 = \beta_6 w_5 = 6.5625$

Coin Betting and Parameter Free Optimisation



Current wealth.

$w_5 = 7.875$

Current bet.

$x_6 = 6.5625$

Outcome.

$c_6 = -\nabla f(x_6) = 1$

New wealth.

$w_6 = w_5 + c_6 x_6 = 14.4375$

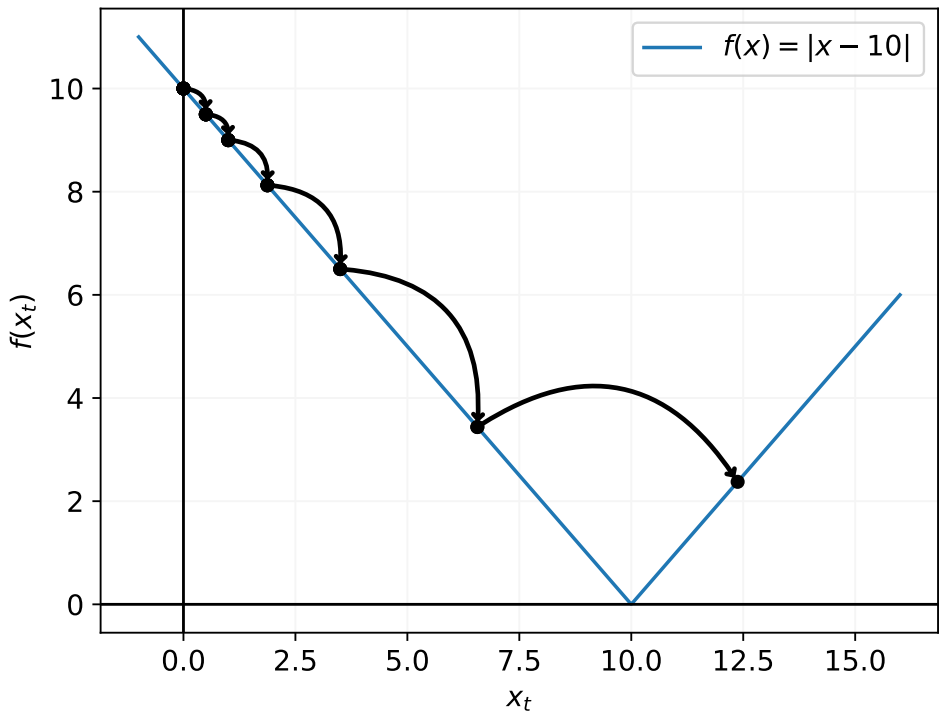
New betting fraction.

$$\beta_7 = \frac{\sum_{i=1}^{6} c_i}{7} = 0.\dot{8}57142\dot{}$$

New bet.

$x_7 = \beta_7 w_6 = 12.375$

Coin Betting and Parameter Free Optimisation



Current wealth.

$w_5 = 7.875$

Current bet.

$x_6 = 6.5625$

Outcome.

$c_6 = -\nabla f(x_6) = 1$

New wealth.

$w_6 = w_5 + c_6 x_6 = 14.4375$

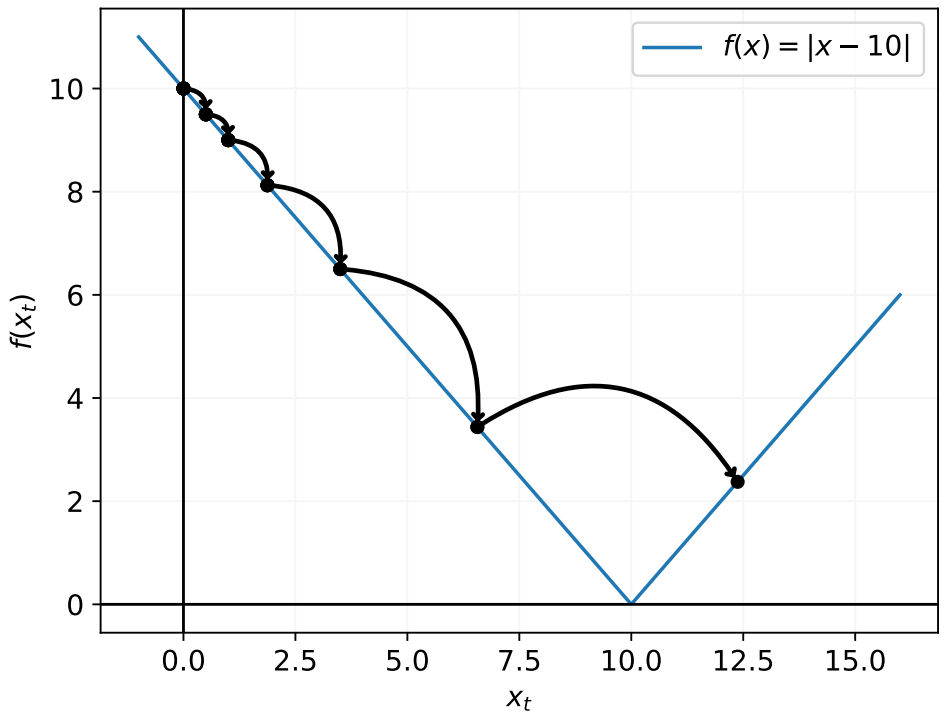
New betting fraction.

$$\beta_7 = \frac{\sum_{i=1}^{6} c_i}{7} = 0.\dot{8}57142\dot{}$$

New bet.

$x_7 = \beta_7 w_6 = 12.375$

Coin Betting and Parameter Free Optimisation



Current wealth.

$w_6 = 14.4375$

Current bet.

$x_7 = 12.375$

Outcome.

$c_7 = -\nabla f(x_7) = -1$

New wealth.

$w_7 = w_6 + c_7 x_7 = 2.0625$

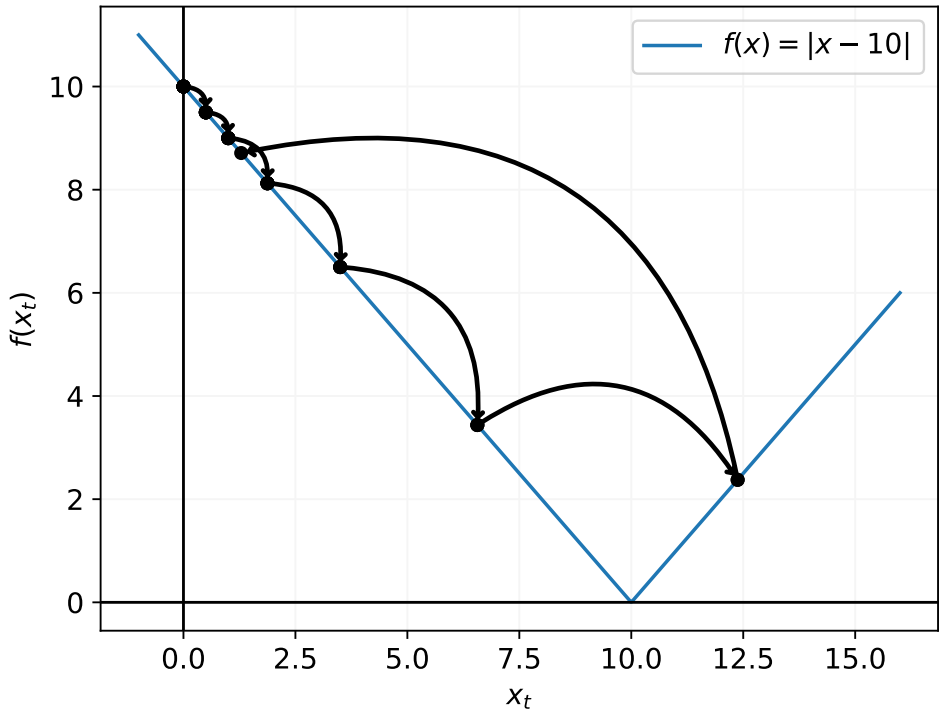
New betting fraction.

$$\beta_8 = \frac{\sum_{i=1}^{7} c_i}{8} = 0.625$$

New bet.

$x_8 = \beta_8 w_7 = 1.2890625$

Coin Betting and Parameter Free Optimisation



Current wealth.

$w_6 = 14.4375$

Current bet.

$x_7 = 12.375$

Outcome.

$c_7 = -\nabla f(x_7) = -1$

New wealth.

$w_7 = w_6 + c_7 x_7 = 2.0625$

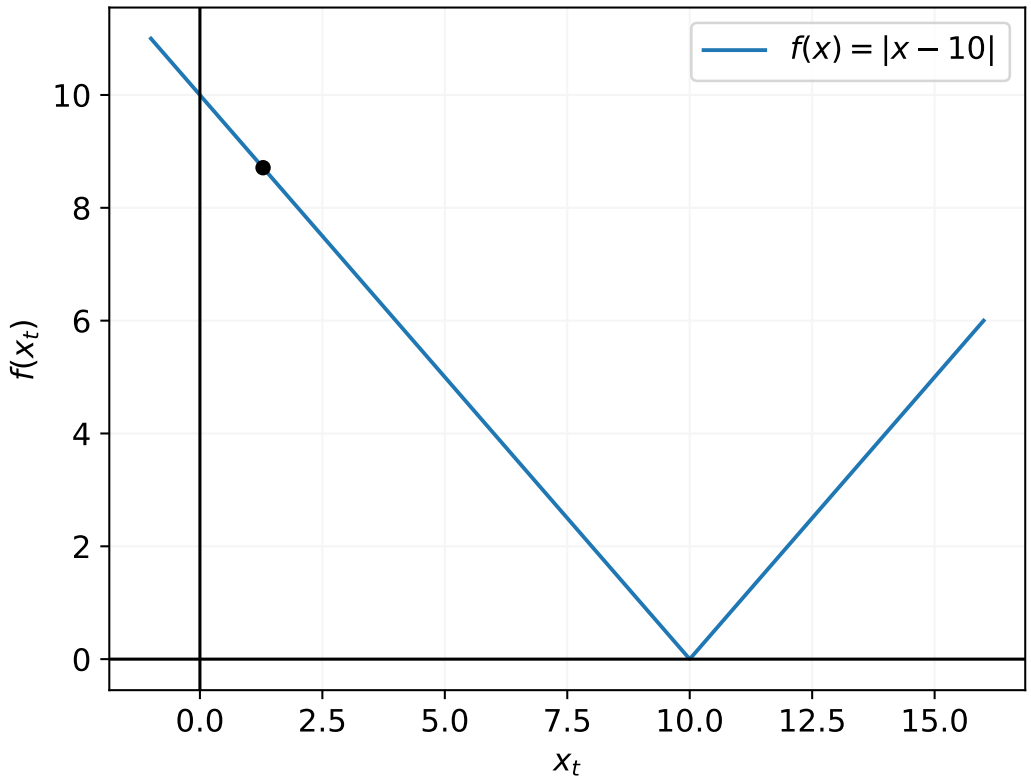
New betting fraction.

$$\beta_8 = \frac{\sum_{i=1}^{7} c_i}{8} = 0.625$$

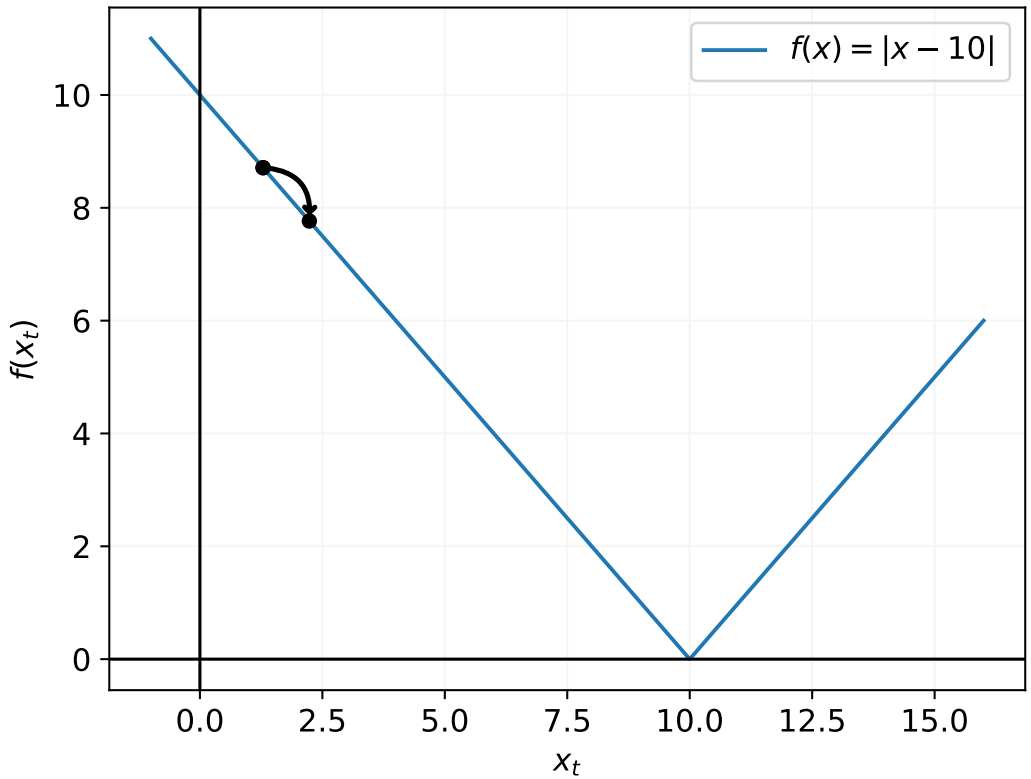
New bet.

$x_8 = \beta_8 w_7 = 1.2890625$

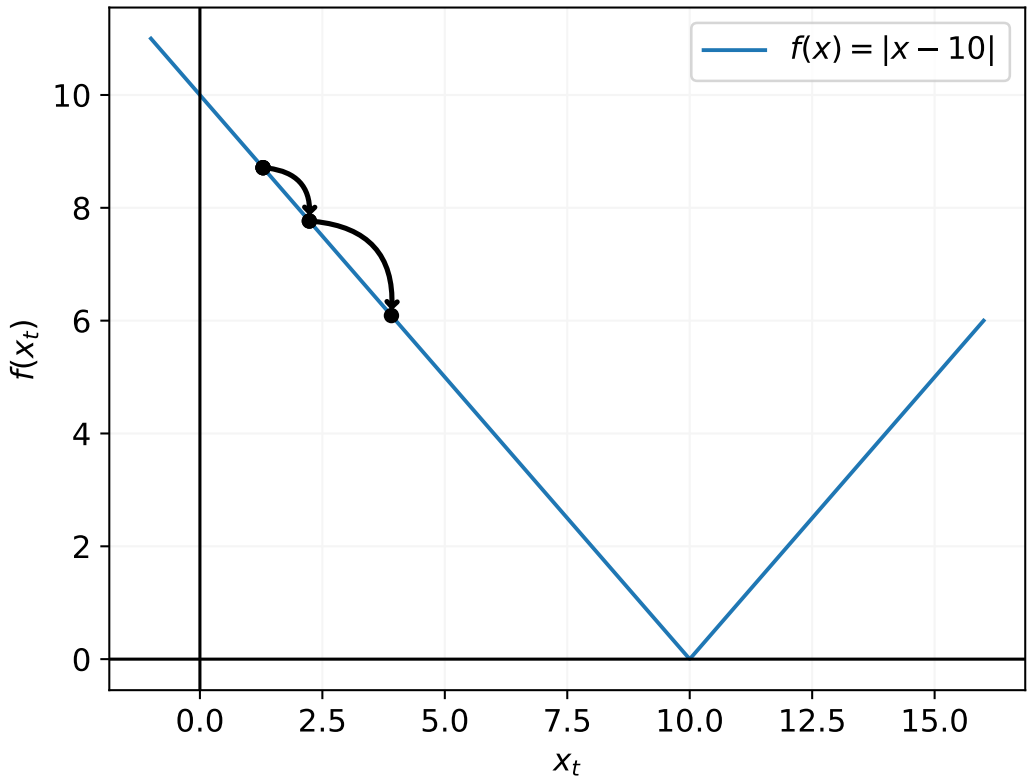
Coin Betting and Parameter Free Optimisation



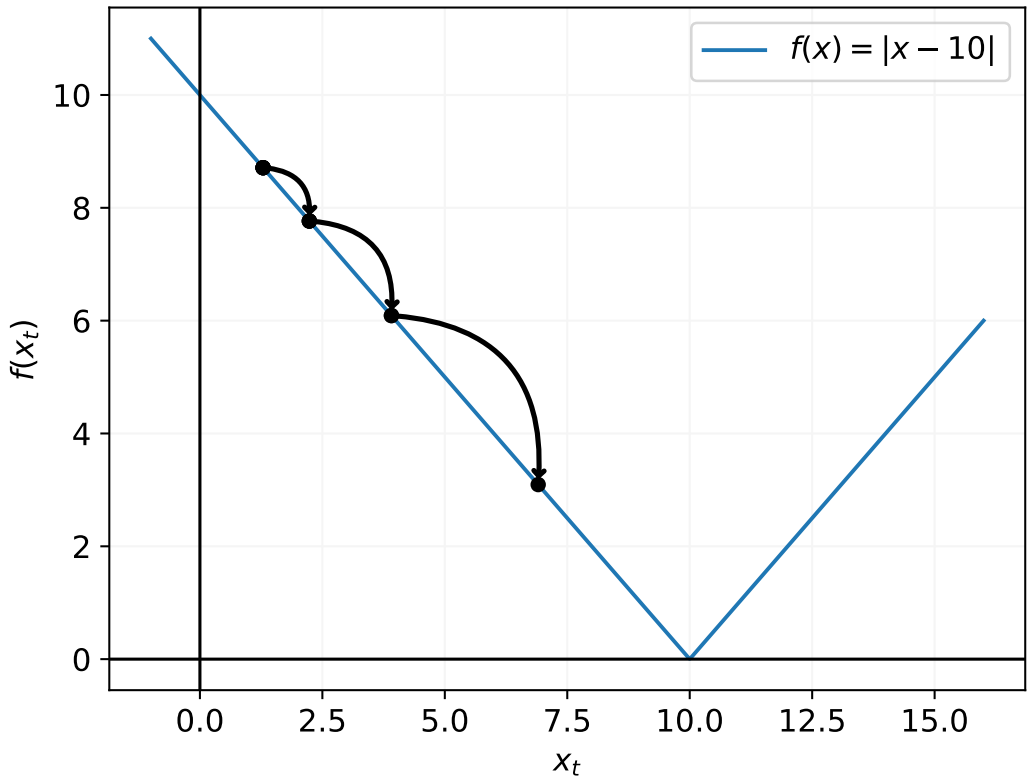
Coin Betting and Parameter Free Optimisation



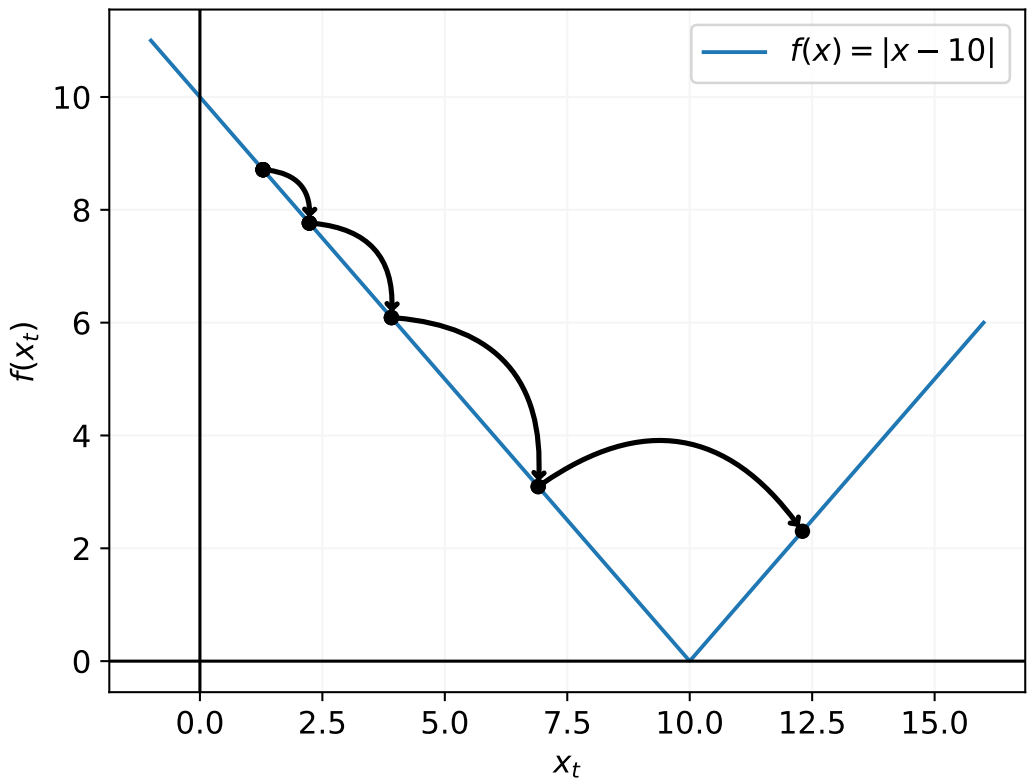
Coin Betting and Parameter Free Optimisation



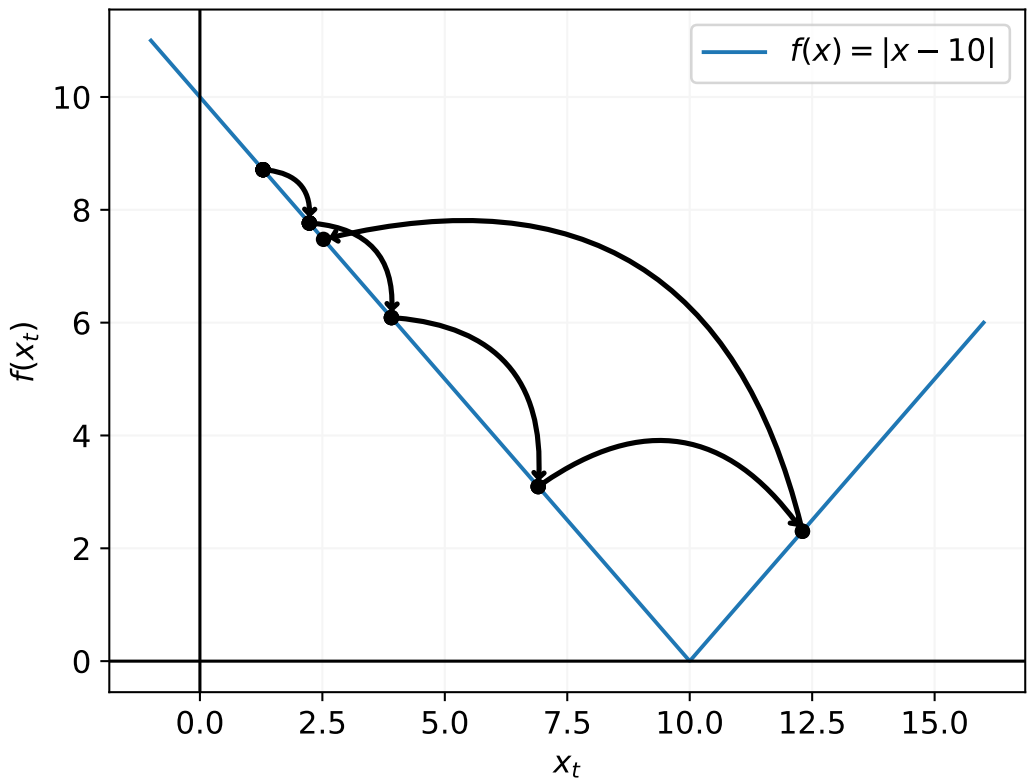
Coin Betting and Parameter Free Optimisation



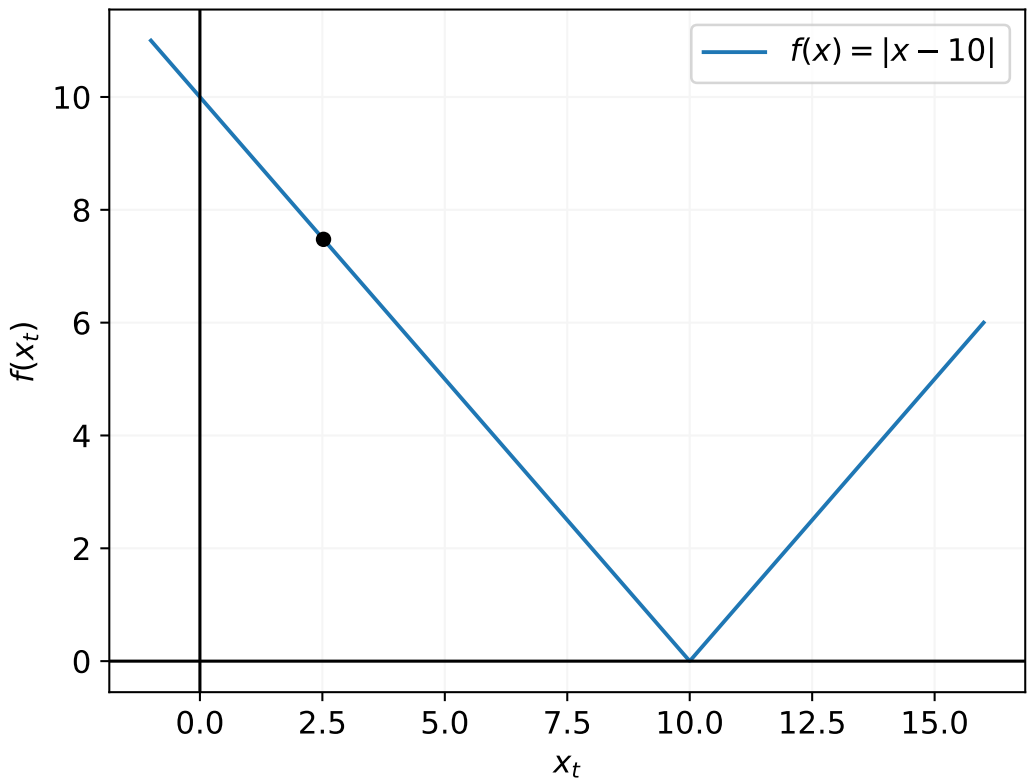
Coin Betting and Parameter Free Optimisation



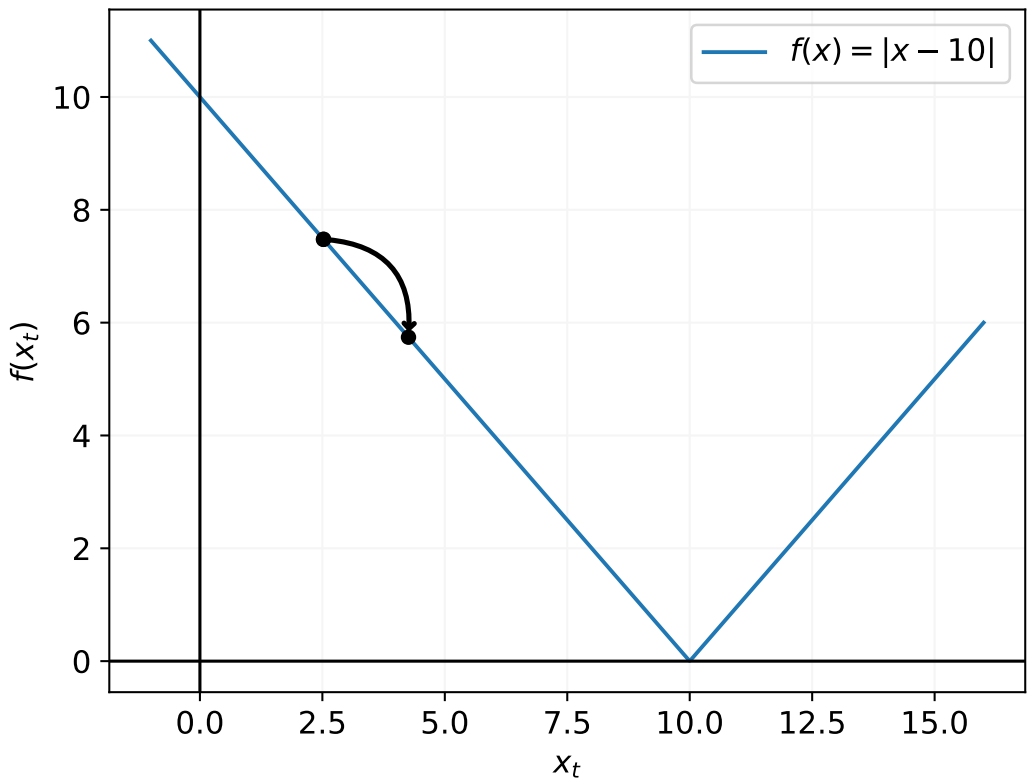
Coin Betting and Parameter Free Optimisation



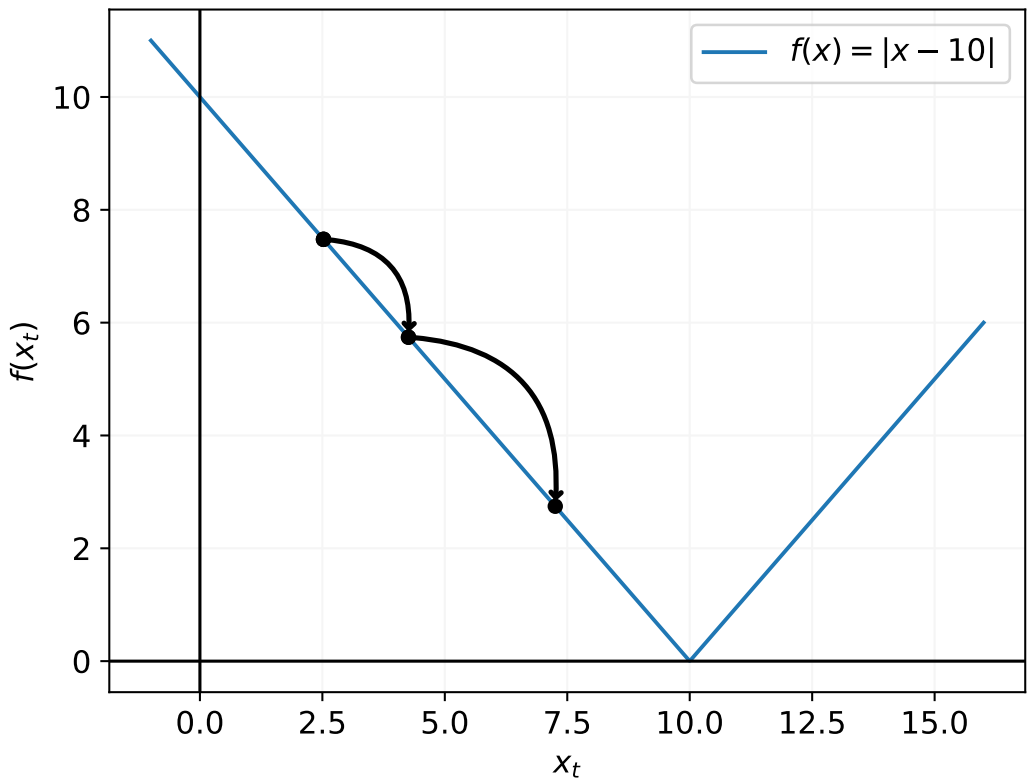
Coin Betting and Parameter Free Optimisation



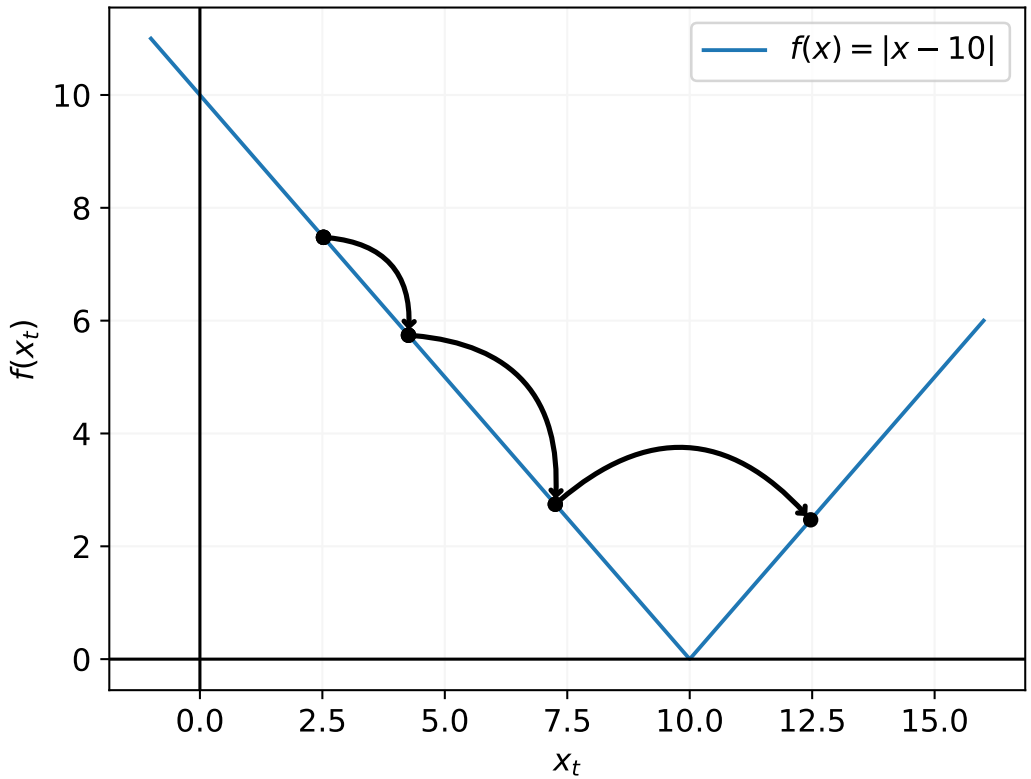
Coin Betting and Parameter Free Optimisation



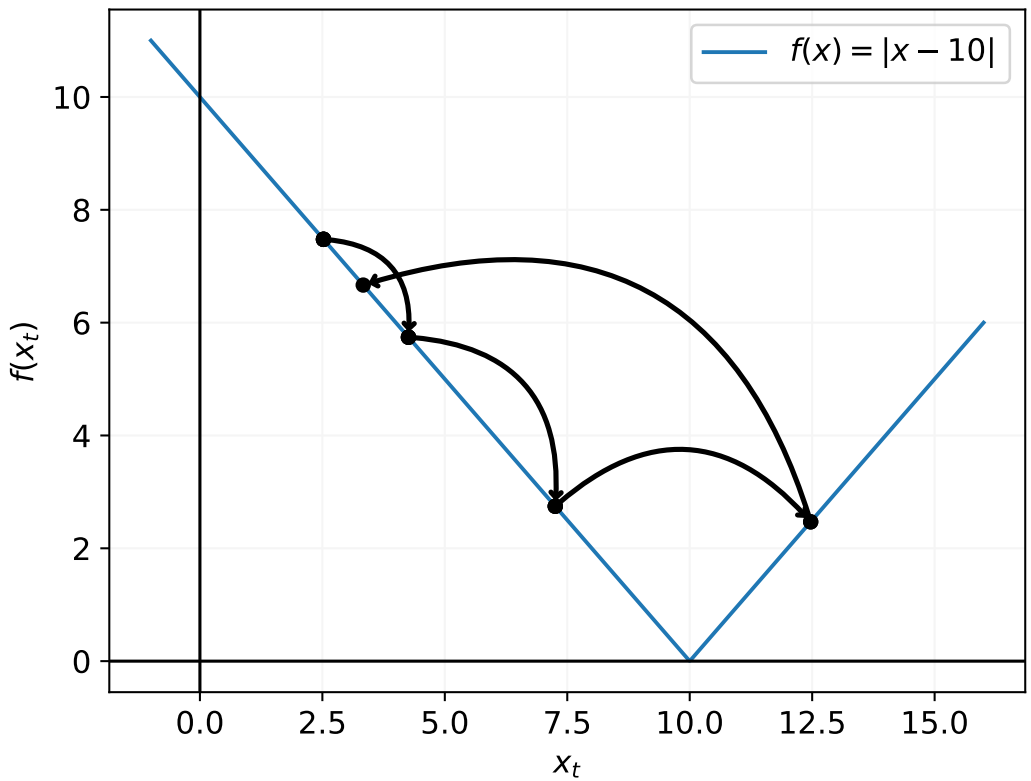
Coin Betting and Parameter Free Optimisation



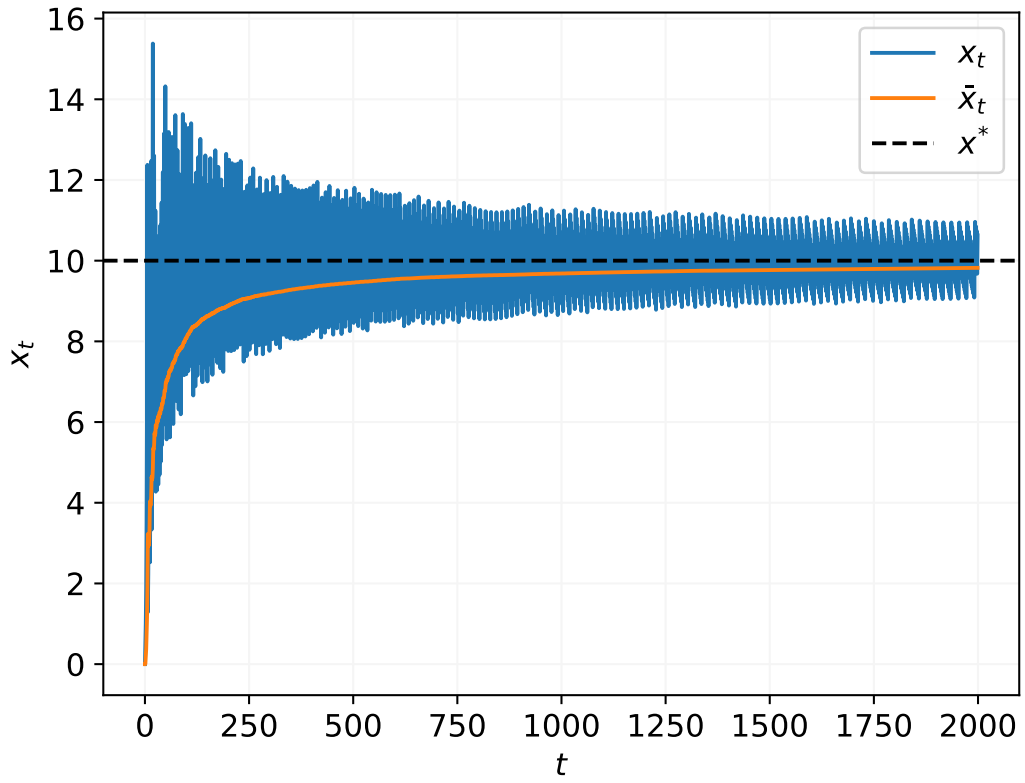
Coin Betting and Parameter Free Optimisation



Coin Betting and Parameter Free Optimisation



Coin Betting and Parameter Free Optimisation



Going from $\mathbb{R}$ to $\mathbb{R}^d$

Thus far we have just just considered $f : \mathbb{R} \to \mathbb{R}$. What about the case where $f : \mathbb{R}^d \to \mathbb{R}$?

There are a few different approaches. The most straightforward are:

- **Inner products**.

$$x_t = -\frac{\sum_{i=1}^{t-1} \nabla f(x_i)}{t} \left(\varepsilon - \sum_{i=1}^{t-1} \langle \nabla f(x_i), x_i \rangle \right).$$

- **Coordinate-wise updates**. For $j = 1, \ldots, d$,

$$x_{t,j} = -\frac{\sum_{i=1}^{t-1} \partial_j f(x_i)}{t} \left(\varepsilon - \sum_{i=1}^{t-1} \partial_j f(x_i) x_{i,j} \right).$$

Going from $\mathbb{R}$ to $\mathbb{R}^d$

Thus far we have just just considered $f : \mathbb{R} \to \mathbb{R}$. What about the case where $f : \mathbb{R}^d \to \mathbb{R}$?

There are a few different approaches. The most straightforward are:

- **Inner products**.

$$x_t = -\frac{\sum_{i=1}^{t-1} \nabla f(x_i)}{t} \left(\varepsilon - \sum_{i=1}^{t-1} \langle \nabla f(x_i), x_i \rangle \right).$$

- **Coordinate-wise updates**. For $j = 1, \ldots, d$,

$$x_{t,j} = -\frac{\sum_{i=1}^{t-1} \partial_j f(x_i)}{t} \left(\varepsilon - \sum_{i=1}^{t-1} \partial_j f(x_i) x_{i,j} \right).$$

Going from $\mathbb{R}$ to $\mathbb{R}^d$

Thus far we have just just considered $f : \mathbb{R} \to \mathbb{R}$. What about the case where $f : \mathbb{R}^d \to \mathbb{R}$?

There are a few different approaches. The most straightforward are:

- **Inner products**.

$$x_t = -\frac{\sum_{i=1}^{t-1} \nabla f(x_i)}{t} \left(\varepsilon - \sum_{i=1}^{t-1} \langle \nabla f(x_i), x_i \rangle \right).$$

- **Coordinate-wise updates**. For $j = 1, \ldots, d$,

$$x_{t,j} = -\frac{\sum_{i=1}^{t-1} \partial_j f(x_i)}{t} \left(\varepsilon - \sum_{i=1}^{t-1} \partial_j f(x_i) x_{i,j} \right).$$

Going from $\mathbb{R}$ to $\mathbb{R}^d$

Thus far we have just just considered $f : \mathbb{R} \to \mathbb{R}$. What about the case where $f : \mathbb{R}^d \to \mathbb{R}$?

There are a few different approaches. The most straightforward are:

- **Inner products**.

$$x_t = -\frac{\sum_{i=1}^{t-1} \nabla f(x_i)}{t} \left(\varepsilon - \sum_{i=1}^{t-1} \langle \nabla f(x_i), x_i \rangle \right).$$

- **Coordinate-wise updates**. For $j = 1, \ldots, d$,

$$x_{t,j} = -\frac{\sum_{i=1}^{t-1} \partial_j f(x_i)}{t} \left(\varepsilon - \sum_{i=1}^{t-1} \partial_j f(x_i) x_{i,j} \right).$$

Optimisation in Wasserstein Space

The Wasserstein Space

Let $\mathcal{P}_2(\mathbb{R}^d)$ denote the space of probability distributions on $\mathbb{R}^d$ with finite second moments:

$$\mathcal{P}_2(\mathbb{R}^d) = \left\{ \mu \in \mathcal{P}(\mathbb{R}^d) : \int_{\mathbb{R}^d} ||x||^2 \mu(\mathrm{d}x) < \infty \right\}.$$

Let W_2 denote the Wasserstein 2-distance between μ and ν, defined according to

$$W_2^2(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu,\nu)} \int_{\mathbb{R}^d \times \mathbb{R}^d} ||x - y||^2 \gamma(\mathrm{d}x, \mathrm{d}y).$$

where $\Gamma(\mu, \nu)$ denotes the set of **couplings** between μ and ν (i.e., joint distributions on $\mathbb{R}^d \times \mathbb{R}^d$ with marginals μ and ν).

$(\mathcal{P}_2(\mathbb{R}^d), W_2)$ is a metric space of probability measures, known as the **Wasserstein space**.

The Wasserstein Space

Let $\mathcal{P}_2(\mathbb{R}^d)$ denote the space of probability distributions on $\mathbb{R}^d$ with finite second moments:

$$\mathcal{P}_2(\mathbb{R}^d) = \left\{ \mu \in \mathcal{P}(\mathbb{R}^d) : \int_{\mathbb{R}^d} ||x||^2 \mu(\mathrm{d}x) < \infty \right\}.$$

Let W_2 denote the Wasserstein 2-distance between μ and ν, defined according to

$$W_2^2(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu,\nu)} \int_{\mathbb{R}^d \times \mathbb{R}^d} ||x - y||^2 \gamma(\mathrm{d}x, \mathrm{d}y).$$

where $\Gamma(\mu, \nu)$ denotes the set of **couplings** between μ and ν (i.e., joint distributions on $\mathbb{R}^d \times \mathbb{R}^d$ with marginals μ and ν).

$(\mathcal{P}_2(\mathbb{R}^d), W_2)$ is a metric space of probability measures, known as the **Wasserstein space**.

The Wasserstein Space

Let $\mathcal{P}_2(\mathbb{R}^d)$ denote the space of probability distributions on $\mathbb{R}^d$ with finite second moments:

$$\mathcal{P}_2(\mathbb{R}^d) = \left\{ \mu \in \mathcal{P}(\mathbb{R}^d) : \int_{\mathbb{R}^d} ||x||^2 \mu(\mathrm{d}x) < \infty \right\}.$$

Let W_2 denote the Wasserstein 2-distance between μ and ν, defined according to

$$W_2^2(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu,\nu)} \int_{\mathbb{R}^d \times \mathbb{R}^d} ||x - y||^2 \gamma(\mathrm{d}x, \mathrm{d}y).$$

where $\Gamma(\mu, \nu)$ denotes the set of **couplings** between μ and ν (i.e., joint distributions on $\mathbb{R}^d \times \mathbb{R}^d$ with marginals μ and ν).

$(\mathcal{P}_2(\mathbb{R}^d), W_2)$ is a metric space of probability measures, known as the **Wasserstein space**.

Optimisation in Wasserstein Space

Recall that we are interested in solving the following **optimisation problem**:

$$\pi = \underset{\mu \in \mathcal{P}_2(\mathbb{R}^d)}{\arg\min}\ \mathcal{F}(\mu)$$

where $\mathcal{F} : \mathcal{P}(\mathbb{R}^d) \to \mathbb{R}$ is a **dissimilarity functional** which is uniquely minimised at π.

There are various possible choices for $\mathcal{F}$. A typical choice is the **Kullback-Leibler (KL) divergence**

$$\mathrm{KL}(\mu|\pi) = \begin{cases} \displaystyle\int_{\mathbb{R}^d} \log\left(\frac{\mathrm{d}\mu}{\mathrm{d}\pi}(x)\right)\mu(\mathrm{d}x) & , \quad \mu \ll \pi, \\ +\infty & , \quad \text{else.} \end{cases}$$

Optimisation in Wasserstein Space

Recall that we are interested in solving the following **optimisation problem**:

$$\pi = \underset{\mu \in \mathcal{P}_2(\mathbb{R}^d)}{\arg\min}\ \mathcal{F}(\mu)$$

where $\mathcal{F} : \mathcal{P}(\mathbb{R}^d) \to \mathbb{R}$ is a **dissimilarity functional** which is uniquely minimised at π.

There are various possible choices for $\mathcal{F}$. A typical choice is the **Kullback-Leibler (KL) divergence**

$$\mathrm{KL}(\mu|\pi) = \begin{cases} \displaystyle\int_{\mathbb{R}^d} \log\left(\frac{\mathrm{d}\mu}{\mathrm{d}\pi}(x)\right)\mu(\mathrm{d}x) & , \quad \mu \ll \pi, \\ +\infty & , \quad \text{else.} \end{cases}$$

Wasserstein Gradient Flows

To solve this optimisation problem, a natural approach would be to consider the **gradient flow** of $\mathcal{F}$ over the Wasserstein space.

First, we will need to define an appropriate notion of the gradient on $(\mathcal{P}_2(\mathbb{R}^d), W_2)$.

Definition

Let $\mu \in \mathcal{P}_2(\mathbb{R}^d)$. Consider a **perturbation on the Wasserstein space** $(\mathrm{id} + \varepsilon h)_{\#}\mu$, for $h \in L^2(\mu)$. Then, if

$$\mathcal{F}\left((\mathrm{id} + \varepsilon h)_{\#}\mu\right) = \mathcal{F}(\mu) + \varepsilon \langle \nabla_{W_2}\mathcal{F}(\mu), h \rangle_{\mu} + o(\varepsilon)$$

we say that $\nabla_{W_2}\mathcal{F}(\mu) \in L^2(\mu)$ is the **Wasserstein gradient** of $\mathcal{F}$ at μ.

Wasserstein Gradient Flows

To solve this optimisation problem, a natural approach would be to consider the **gradient flow** of $\mathcal{F}$ over the Wasserstein space.

First, we will need to define an appropriate notion of the gradient on $(\mathcal{P}_2(\mathbb{R}^d), W_2)$.

Definition

Let $\mu \in \mathcal{P}_2(\mathbb{R}^d)$. Consider a **perturbation on the Wasserstein space** $(\mathrm{id} + \varepsilon h)_{\#}\mu$, for $h \in L^2(\mu)$. Then, if

$$\mathcal{F}\left((\mathrm{id} + \varepsilon h)_{\#}\mu\right) = \mathcal{F}(\mu) + \varepsilon \langle \nabla_{W_2}\mathcal{F}(\mu), h \rangle_{\mu} + o(\varepsilon)$$

we say that $\nabla_{W_2}\mathcal{F}(\mu) \in L^2(\mu)$ is the **Wasserstein gradient** of $\mathcal{F}$ at μ.

Wasserstein Gradient Flows

To solve this optimisation problem, a natural approach would be to consider the **gradient flow** of $\mathcal{F}$ over the Wasserstein space.

First, we will need to define an appropriate notion of the gradient on $(\mathcal{P}_2(\mathbb{R}^d), W_2)$.

Definition

Let $\mu \in \mathcal{P}_2(\mathbb{R}^d)$. Consider a **perturbation on the Wasserstein space** $(\mathrm{id} + \varepsilon h)_{\#}\mu$, for $h \in L^2(\mu)$. Then, if

$$\mathcal{F}\left((\mathrm{id} + \varepsilon h)_{\#}\mu\right) = \mathcal{F}(\mu) + \varepsilon \langle \nabla_{W_2}\mathcal{F}(\mu), h \rangle_{\mu} + o(\varepsilon)$$

we say that $\nabla_{W_2}\mathcal{F}(\mu) \in L^2(\mu)$ is the **Wasserstein gradient** of $\mathcal{F}$ at μ.

Wasserstein Gradient Flows

The Wasserstein gradient can be related to the **first variation** of $\mathcal{F}$ at μ, defined as follows.

Definition

Let $\mu \in \mathcal{P}_2(\mathbb{R}^d)$. Consider a **linear perturbation** $\mu + \varepsilon\xi \in \mathcal{P}_2(\mathbb{R}^d)$, for some $\xi \in \mathcal{P}_2(\mathbb{R}^d)$. Then, if

$$\mathcal{F}(\mu + \varepsilon\xi) = \mathcal{F}(\mu) + \varepsilon \int \mathcal{F}'(\mu)(x)\xi(\mathrm{d}x) + o(\varepsilon),$$

we say that $\mathcal{F}'(\mu) : \mathbb{R}^d \to \mathbb{R}$ is the **first variation** of $\mathcal{F}$ at μ.

In particular, in most cases of interest [AGG08, Theorem 10.4.13], we have that

$$\nabla_{W_2}\mathcal{F}(\mu) = \nabla\mathcal{F}'(\mu).$$

Wasserstein Gradient Flows

The Wasserstein gradient can be related to the **first variation** of $\mathcal{F}$ at μ, defined as follows.

Definition

Let $\mu \in \mathcal{P}_2(\mathbb{R}^d)$. Consider a **linear perturbation** $\mu + \varepsilon\xi \in \mathcal{P}_2(\mathbb{R}^d)$, for some $\xi \in \mathcal{P}_2(\mathbb{R}^d)$. Then, if

$$\mathcal{F}(\mu + \varepsilon\xi) = \mathcal{F}(\mu) + \varepsilon \int \mathcal{F}'(\mu)(x)\xi(\mathrm{d}x) + o(\varepsilon),$$

we say that $\mathcal{F}'(\mu) : \mathbb{R}^d \to \mathbb{R}$ is the **first variation** of $\mathcal{F}$ at μ.

In particular, in most cases of interest [AGG08, Theorem 10.4.13], we have that

$$\nabla_{W_2}\mathcal{F}(\mu) = \nabla\mathcal{F}'(\mu).$$

Wasserstein Gradient Flows

The Wasserstein gradient can be related to the **first variation** of $\mathcal{F}$ at μ, defined as follows.

Definition

Let $\mu \in \mathcal{P}_2(\mathbb{R}^d)$. Consider a **linear perturbation** $\mu + \varepsilon\xi \in \mathcal{P}_2(\mathbb{R}^d)$, for some $\xi \in \mathcal{P}_2(\mathbb{R}^d)$. Then, if

$$\mathcal{F}(\mu + \varepsilon\xi) = \mathcal{F}(\mu) + \varepsilon \int \mathcal{F}'(\mu)(x)\xi(\mathrm{d}x) + o(\varepsilon),$$

we say that $\mathcal{F}'(\mu) : \mathbb{R}^d \to \mathbb{R}$ is the **first variation** of $\mathcal{F}$ at μ.

In particular, in most cases of interest [AGG08, Theorem 10.4.13], we have that

$$\boxed{\nabla_{W_2}\mathcal{F}(\mu) = \nabla\mathcal{F}'(\mu)}.$$

Wasserstein Gradient Flows

The **Wasserstein gradient flow** (WGF) of $\mathcal{F}$ is defined as the weak solution $\mu : [0, \infty) \to \mathcal{P}_2(\mathbb{R}^d)$ of the continuity equation [AGG08]

$$\partial_t \mu_t + \nabla \cdot (v_t \mu_t) = 0 \ , \quad v_t = -\nabla_{W_2} \mathcal{F}(\mu_t).$$

How can we interpret this? Let $x = (x_t)_{t \geq 0}$ be the integral curve of the vector fields $(v_t)_{t \geq 0}$, viz

$$\dot{x}_t = -\nabla_{W_2} \mathcal{F}(\mu_t)(x_t) \ , \quad v_t = -\nabla_{W_2} \mathcal{F}(\mu_t),$$

with initial condition $x_0 \sim \mu_0$. Then the WGF describes the evolution of the marginal law $(\mu_t)_{t \geq 0}$ of $(x_t)_{t \geq 0}$.

Wasserstein Gradient Flows

The **Wasserstein gradient flow** (WGF) of $\mathcal{F}$ is defined as the weak solution $\mu : [0, \infty) \to \mathcal{P}_2(\mathbb{R}^d)$ of the continuity equation [AGG08]

$$\partial_t \mu_t + \nabla \cdot (v_t \mu_t) = 0 \ , \quad v_t = -\nabla_{W_2} \mathcal{F}(\mu_t).$$

How can we interpret this? Let $x = (x_t)_{t \geq 0}$ be the integral curve of the vector fields $(v_t)_{t \geq 0}$, viz

$$\dot{x}_t = -\nabla_{W_2} \mathcal{F}(\mu_t)(x_t) \ , \quad v_t = -\nabla_{W_2} \mathcal{F}(\mu_t),$$

with initial condition $x_0 \sim \mu_0$. Then the WGF describes the evolution of the marginal law $(\mu_t)_{t \geq 0}$ of $(x_t)_{t \geq 0}$.

Wasserstein Gradient Flows: A Quick Example

Suppose we consider $\mathcal{F}(\mu) = \mathrm{KL}(\mu||\pi)$. In this case, the first variation is $\mathcal{F}'(\mu) = \log\left(\frac{\mu}{\pi}\right)$, and thus $\nabla_{W_2}\mathcal{F}(\mu) = \nabla \log\left(\frac{\mu}{\pi}\right)$.

The Wasserstein gradient flow can thus be written as

$$\frac{\partial \mu_t}{\partial t} = \nabla \cdot \left[\mu_t \nabla \log\left(\frac{\mu_t}{\pi}\right)\right] = \nabla \cdot (\mu_t \nabla U) + \Delta \mu_t.$$

This is nothing more than the Fokker-Planck equation of the **overdamped Langevin dynamics**

$$\mathrm{d}x_t = -\nabla U(x_t)\mathrm{d}t + \sqrt{2}\mathrm{d}w_t$$

where $w = (w_t)_{t\geq 0}$ is a standard $\mathbb{R}^d$-valued Brownian motion.

Wasserstein Gradient Flows: A Quick Example

Suppose we consider $\mathcal{F}(\mu) = \mathrm{KL}(\mu||\pi)$. In this case, the first variation is $\mathcal{F}'(\mu) = \log\left(\frac{\mu}{\pi}\right)$, and thus $\nabla_{W_2}\mathcal{F}(\mu) = \nabla \log\left(\frac{\mu}{\pi}\right)$.

The Wasserstein gradient flow can thus be written as

$$\frac{\partial \mu_t}{\partial t} = \nabla \cdot \left[\mu_t \nabla \log\left(\frac{\mu_t}{\pi}\right)\right] = \nabla \cdot (\mu_t \nabla U) + \Delta \mu_t.$$

This is nothing more than the Fokker-Planck equation of the **overdamped Langevin dynamics**

$$\mathrm{d}x_t = -\nabla U(x_t)\mathrm{d}t + \sqrt{2}\mathrm{d}w_t$$

where $w = (w_t)_{t\geq 0}$ is a standard $\mathbb{R}^d$-valued Brownian motion.

Wasserstein Gradient Flows: A Quick Example

Suppose we consider $\mathcal{F}(\mu) = \mathrm{KL}(\mu||\pi)$. In this case, the first variation is $\mathcal{F}'(\mu) = \log\left(\frac{\mu}{\pi}\right)$, and thus $\nabla_{W_2}\mathcal{F}(\mu) = \nabla \log\left(\frac{\mu}{\pi}\right)$.

The Wasserstein gradient flow can thus be written as

$$\frac{\partial \mu_t}{\partial t} = \nabla \cdot \left[\mu_t \nabla \log\left(\frac{\mu_t}{\pi}\right)\right] = \nabla \cdot (\mu_t \nabla U) + \Delta \mu_t.$$

This is nothing more than the Fokker-Planck equation of the **overdamped Langevin dynamics**

$$\mathrm{d}x_t = -\nabla U(x_t)\mathrm{d}t + \sqrt{2}\mathrm{d}w_t$$

where $w = (w_t)_{t\geq 0}$ is a standard $\mathbb{R}^d$-valued Brownian motion.

Wasserstein Gradient Descent

In practice, we we will need to **discretise** the Wasserstein gradient flow. Similar to the Euclidean case, we have several options:

- The **backward Euler discretisation**, which results in the JKO scheme [JKO98]

$$\mu_{t+1} \in \underset{\mu \in \mathcal{P}_2(\mathbb{R}^d)}{\arg\min} \left[\mathcal{F}(\mu) + \frac{1}{2\gamma} \mathbb{W}_2^2(\mu, \mu_t) \right],$$

- The **forward Euler discretisation**, which results in the **Wasserstein gradient descent** algorithm [GHLR22]

$$\mu_{t+1} = (\mathbf{id} - \gamma \xi_t)_{\#} \mu_t \ , \quad \xi_t = \nabla_{W_2} \mathcal{F}(\mu_t),$$

In $\mathbb{R}^d$, writing $\mu_t = \mathrm{Law}(x_t)$, this corresponds to

$$x_{t+1} = x_t - \gamma \nabla_{W_2} \mathcal{F}(\mu_t)(x_t) \ , \quad x_0 \sim \mu_0.$$

Wasserstein Gradient Descent

In practice, we we will need to **discretise** the Wasserstein gradient flow. Similar to the Euclidean case, we have several options:

- The **backward Euler discretisation**, which results in the JKO scheme [JKO98]

$$\mu_{t+1} \in \operatorname*{arg\,min}_{\mu \in \mathcal{P}_2(\mathbb{R}^d)} \left[\mathcal{F}(\mu) + \frac{1}{2\gamma} \mathbb{W}_2^2(\mu, \mu_t) \right],$$

- The **forward Euler discretisation**, which results in the **Wasserstein gradient descent** algorithm [GHLR22]

$$\mu_{t+1} = (\mathbf{id} - \gamma \xi_t)_{\#} \mu_t \ , \quad \xi_t = \nabla_{W_2} \mathcal{F}(\mu_t),$$

In $\mathbb{R}^d$, writing $\mu_t = \mathrm{Law}(x_t)$, this corresponds to

$$x_{t+1} = x_t - \gamma \nabla_{W_2} \mathcal{F}(\mu_t)(x_t) \ , \quad x_0 \sim \mu_0.$$

Wasserstein Gradient Descent

In practice, we we will need to **discretise** the Wasserstein gradient flow. Similar to the Euclidean case, we have several options:

- The **backward Euler discretisation**, which results in the JKO scheme [JKO98]

$$\mu_{t+1} \in \operatorname*{arg\,min}_{\mu \in \mathcal{P}_2(\mathbb{R}^d)} \left[\mathcal{F}(\mu) + \frac{1}{2\gamma} \mathbb{W}_2^2(\mu, \mu_t) \right],$$

- The **forward Euler discretisation**, which results in the **Wasserstein gradient descent** algorithm [GHLR22]

$$\mu_{t+1} = (\mathbf{id} - \gamma \xi_t)_{\#} \mu_t \ , \quad \xi_t = \nabla_{W_2} \mathcal{F}(\mu_t),$$

In $\mathbb{R}^d$, writing $\mu_t = \mathrm{Law}(x_t)$, this corresponds to

$$x_{t+1} = x_t - \gamma \nabla_{W_2} \mathcal{F}(\mu_t)(x_t) \ , \quad x_0 \sim \mu_0.$$

Wasserstein Gradient Descent

For different choices of $\mathcal{F}$, the forward Euler discretisation yields several existing sampling algorithms.

- **Maximum mean discrepancy descent** (MMDD) [AKSG19].
- **Kernel Stein discrepancy descent** (KSDD) [KPCAF+21].

If, in addition, we replace the Wasserstein gradient $\nabla\mathcal{F}(\mu)$ by a kernelised approximation (more on this later), we also have

- **Stein variational gradient descent** (SVGD) [LW16].
- **Laplacian adjusted Wasserstein gradient descent** (LAWGD)[CLL+20].

However, similar to before, the convergence properties of all of these methods depend on the choice of **step size** γ.

Wasserstein Gradient Descent

For different choices of $\mathcal{F}$, the forward Euler discretisation yields several existing sampling algorithms.

- **Maximum mean discrepancy descent** (MMDD) [AKSG19].
- **Kernel Stein discrepancy descent** (KSDD) [KPCAF$^+$21].

If, in addition, we replace the Wasserstein gradient $\nabla\mathcal{F}(\mu)$ by a kernelised approximation (more on this later), we also have

- **Stein variational gradient descent** (SVGD) [LW16].
- **Laplacian adjusted Wasserstein gradient descent** (LAWGD)[CLL$^+$20].

However, similar to before, the convergence properties of all of these methods depend on the choice of **step size** γ.

Wasserstein Gradient Descent

For different choices of $\mathcal{F}$, the forward Euler discretisation yields several existing sampling algorithms.

- **Maximum mean discrepancy descent** (MMDD) [AKSG19].
- **Kernel Stein discrepancy descent** (KSDD) [KPCAF$^+$21].

If, in addition, we replace the Wasserstein gradient $\nabla\mathcal{F}(\mu)$ by a kernelised approximation (more on this later), we also have

- **Stein variational gradient descent** (SVGD) [LW16].
- **Laplacian adjusted Wasserstein gradient descent** (LAWGD)[CLL$^+$20].

However, similar to before, the convergence properties of all of these methods depend on the choice of **step size** γ.

Wasserstein Gradient Descent

For different choices of $\mathcal{F}$, the forward Euler discretisation yields several existing sampling algorithms.

- **Maximum mean discrepancy descent** (MMDD) [AKSG19].
- **Kernel Stein discrepancy descent** (KSDD) [KPCAF$^+$21].

If, in addition, we replace the Wasserstein gradient $\nabla\mathcal{F}(\mu)$ by a kernelised approximation (more on this later), we also have

- **Stein variational gradient descent** (SVGD) [LW16].
- **Laplacian adjusted Wasserstein gradient descent** (LAWGD)[CLL$^+$20].

However, similar to before, the convergence properties of all of these methods depend on the choice of **step size** γ.

Wasserstein Gradient Descent

For different choices of $\mathcal{F}$, the forward Euler discretisation yields several existing sampling algorithms.

- **Maximum mean discrepancy descent** (MMDD) [AKSG19].
- **Kernel Stein discrepancy descent** (KSDD) [KPCAF$^+$21].

If, in addition, we replace the Wasserstein gradient $\nabla\mathcal{F}(\mu)$ by a kernelised approximation (more on this later), we also have

- **Stein variational gradient descent** (SVGD) [LW16].
- **Laplacian adjusted Wasserstein gradient descent** (LAWGD)[CLL$^+$20].

However, similar to before, the convergence properties of all of these methods depend on the choice of **step size** γ.

Wasserstein Gradient Descent

For example, assuming the Wasserstein (sub) gradients satisfy $||\xi_t(x)|| \leq L$ for all $x \in \mathbb{R}^d$, Wasserstein (sub) gradient descent satisfies [GHLR22]

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{1}{T}\left[\frac{W_2^2(\mu_1,\pi)}{2\gamma} + \frac{L^2 T\gamma}{2}\right].$$

This implies that the **optimal** worst case learning rate is given by $\gamma_{\text{opt}} = \frac{W_2(\mu_1,\pi)}{L\sqrt{T}}$, and thus the **optimal error bound** is

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{LW_2(\mu_1,\pi)}{\sqrt{T}}.$$

Once again, this rate cannot be achieved in practice, since γ_{opt} depends on the **unknown** Wasserstein distance $W_2(\mu_1,\pi)$.

Wasserstein Gradient Descent

For example, assuming the Wasserstein (sub) gradients satisfy $||\xi_t(x)|| \leq L$ for all $x \in \mathbb{R}^d$, Wasserstein (sub) gradient descent satisfies [GHLR22]

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{1}{T}\left[\frac{W_2^2(\mu_1,\pi)}{2\gamma} + \frac{L^2 T\gamma}{2}\right].$$

This implies that the **optimal** worst case learning rate is given by $\gamma_{\text{opt}} = \frac{W_2(\mu_1,\pi)}{L\sqrt{T}}$, and thus the **optimal error bound** is

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{LW_2(\mu_1,\pi)}{\sqrt{T}}.$$

Once again, this rate cannot be achieved in practice, since γ_{opt} depends on the **unknown** Wasserstein distance $W_2(\mu_1,\pi)$.

Wasserstein Gradient Descent

For example, assuming the Wasserstein (sub) gradients satisfy $||\xi_t(x)|| \leq L$ for all $x \in \mathbb{R}^d$, Wasserstein (sub) gradient descent satisfies [GHLR22]

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{1}{T}\left[\frac{W_2^2(\mu_1,\pi)}{2\gamma} + \frac{L^2 T\gamma}{2}\right].$$

This implies that the **optimal** worst case learning rate is given by $\gamma_{\text{opt}} = \frac{W_2(\mu_1,\pi)}{L\sqrt{T}}$, and thus the **optimal error bound** is

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{LW_2(\mu_1,\pi)}{\sqrt{T}}.$$

Once again, this rate cannot be achieved in practice, since γ_{opt} depends on the **unknown** Wasserstein distance $W_2(\mu_1,\pi)$.

Coin Wasserstein Gradient Descent

We will obtain a **learning-rate free** sampling algorithm by extending the coin-betting approach to $\mathcal{P}_2(\mathbb{R}^d)$. The setup is similar to before.

- Consider a gambler with **initial wealth** w_0, for some $w_0 > 0$.
- We suppose that the **gambler** bets $(x_t - x_0)_{t=1}^T$ on a series of outcomes $(c_t)_{t=1}^T$, for some baseline $x_0 \in \mathbb{R}^d$.
- The **wealth** of the gambler thus accumulates as

$$w_t = w_0 + \sum_{s=1}^{t} \langle c_s, x_s - x_0 \rangle.$$

- We once again assume the bets satisfy $x_t - x_0 = \beta_t w_{t-1}$, and that $\beta_t = \frac{1}{t} \sum_{s=1}^{t-1} c_s$.
- The **bets** made by the gambler are thus given by

$$x_t - x_0 = \frac{\sum_{s=1}^{t-1} c_s}{t} \left(w_0 + \sum_{s=1}^{t-1} \langle c_s, x_s - x_0 \rangle \right).$$

Coin Wasserstein Gradient Descent

We will obtain a **learning-rate free** sampling algorithm by extending the coin-betting approach to $\mathcal{P}_2(\mathbb{R}^d)$. The setup is similar to before.

- Consider a gambler with **initial wealth** w_0, for some $w_0 > 0$.
- We suppose that the **gambler** bets $(x_t - x_0)_{t=1}^T$ on a series of outcomes $(c_t)_{t=1}^T$, for some baseline $x_0 \in \mathbb{R}^d$.
- The **wealth** of the gambler thus accumulates as

$$w_t = w_0 + \sum_{s=1}^{t} \langle c_s, x_s - x_0 \rangle.$$

- We once again assume the bets satisfy $x_t - x_0 = \beta_t w_{t-1}$, and that $\beta_t = \frac{1}{t} \sum_{s=1}^{t-1} c_s$.
- The **bets** made by the gambler are thus given by

$$x_t - x_0 = \frac{\sum_{s=1}^{t-1} c_s}{t} \left(w_0 + \sum_{s=1}^{t-1} \langle c_s, x_s - x_0 \rangle \right).$$

Coin Wasserstein Gradient Descent

We will obtain a **learning-rate free** sampling algorithm by extending the coin-betting approach to $\mathcal{P}_2(\mathbb{R}^d)$. The setup is similar to before.

- Consider a gambler with **initial wealth** w_0, for some $w_0 > 0$.
- We suppose that the **gambler** bets $(x_t - x_0)_{t=1}^T$ on a series of outcomes $(c_t)_{t=1}^T$, for some baseline $x_0 \in \mathbb{R}^d$.
- The **wealth** of the gambler thus accumulates as

$$w_t = w_0 + \sum_{s=1}^{t} \langle c_s, x_s - x_0 \rangle.$$

- We once again assume the bets satisfy $x_t - x_0 = \beta_t w_{t-1}$, and that $\beta_t = \frac{1}{t} \sum_{s=1}^{t-1} c_s$.
- The **bets** made by the gambler are thus given by

$$x_t - x_0 = \frac{\sum_{s=1}^{t-1} c_s}{t} \left(w_0 + \sum_{s=1}^{t-1} \langle c_s, x_s - x_0 \rangle \right).$$

Coin Wasserstein Gradient Descent

We will obtain a **learning-rate free** sampling algorithm by extending the coin-betting approach to $\mathcal{P}_2(\mathbb{R}^d)$. The setup is similar to before.

- Consider a gambler with **initial wealth** w_0, for some $w_0 > 0$.
- We suppose that the **gambler** bets $(x_t - x_0)_{t=1}^T$ on a series of outcomes $(c_t)_{t=1}^T$, for some baseline $x_0 \in \mathbb{R}^d$.
- The **wealth** of the gambler thus accumulates as

$$w_t = w_0 + \sum_{s=1}^{t} \langle c_s, x_s - x_0 \rangle.$$

- We once again assume the bets satisfy $x_t - x_0 = \beta_t w_{t-1}$, and that $\beta_t = \frac{1}{t} \sum_{s=1}^{t-1} c_s$.
- The **bets** made by the gambler are thus given by

$$x_t - x_0 = \frac{\sum_{s=1}^{t-1} c_s}{t} \left(w_0 + \sum_{s=1}^{t-1} \langle c_s, x_s - x_0 \rangle \right).$$

Coin Wasserstein Gradient Descent

We will obtain a **learning-rate free** sampling algorithm by extending the coin-betting approach to $\mathcal{P}_2(\mathbb{R}^d)$. The setup is similar to before.

- Consider a gambler with **initial wealth** w_0, for some $w_0 > 0$.
- We suppose that the **gambler** bets $(x_t - x_0)_{t=1}^T$ on a series of outcomes $(c_t)_{t=1}^T$, for some baseline $x_0 \in \mathbb{R}^d$.
- The **wealth** of the gambler thus accumulates as

$$w_t = w_0 + \sum_{s=1}^{t} \langle c_s, x_s - x_0 \rangle.$$

- We once again assume the bets satisfy $x_t - x_0 = \beta_t w_{t-1}$, and that $\beta_t = \frac{1}{t} \sum_{s=1}^{t-1} c_s$.
- The **bets** made by the gambler are thus given by

$$x_t - x_0 = \frac{\sum_{s=1}^{t-1} c_s}{t} \left(w_0 + \sum_{s=1}^{t-1} \langle c_s, x_s - x_0 \rangle \right).$$

Coin Wasserstein Gradient Descent

We will obtain a **learning-rate free** sampling algorithm by extending the coin-betting approach to $\mathcal{P}_2(\mathbb{R}^d)$. The setup is similar to before.

- Consider a gambler with **initial wealth** w_0, for some $w_0 > 0$.
- We suppose that the **gambler** bets $(x_t - x_0)_{t=1}^T$ on a series of outcomes $(c_t)_{t=1}^T$, for some baseline $x_0 \in \mathbb{R}^d$.
- The **wealth** of the gambler thus accumulates as

$$w_t = w_0 + \sum_{s=1}^{t} \langle c_s, x_s - x_0 \rangle.$$

- We once again assume the bets satisfy $x_t - x_0 = \beta_t w_{t-1}$, and that $\beta_t = \frac{1}{t} \sum_{s=1}^{t-1} c_s$.
- The **bets** made by the gambler are thus given by

$$x_t - x_0 = \frac{\sum_{s=1}^{t-1} c_s}{t} \left(w_0 + \sum_{s=1}^{t-1} \langle c_s, x_s - x_0 \rangle \right).$$

Coin Wasserstein Gradient Descent

We will now need a few additional definitions in order to define our algorithm.

- Let $x_0 \sim \mu_0$, for some initial distribution $\mu_0 \in \mathcal{P}_2(\mathbb{R}^d)$
- Let $\varphi_t : \mathbb{R}^d \to \mathbb{R}^d$ denote the function which maps $x_0 \mapsto x_t$.
- Let $\mu_t \in \mathcal{P}_2(\mathbb{R}^d)$ be the **push-forward** of μ_0 under the function $\varphi_t : \mathbb{R}^d \to \mathbb{R}^d$:

$$\mu_t = (\varphi_t)_{\#}\mu_0.$$

This definition means that, given $x_0 \sim \mu_0$, the random variable $x_t := \varphi_t(x_0)$ is **distributed according to** μ_t.

Coin Wasserstein Gradient Descent

We will now need a few additional definitions in order to define our algorithm.

- Let $x_0 \sim \mu_0$, for some initial distribution $\mu_0 \in \mathcal{P}_2(\mathbb{R}^d)$
- Let $\varphi_t : \mathbb{R}^d \to \mathbb{R}^d$ denote the function which maps $x_0 \mapsto x_t$.
- Let $\mu_t \in \mathcal{P}_2(\mathbb{R}^d)$ be the **push-forward** of μ_0 under the function $\varphi_t : \mathbb{R}^d \to \mathbb{R}^d$:

$$\mu_t = (\varphi_t)_{\#}\mu_0.$$

This definition means that, given $x_0 \sim \mu_0$, the random variable $x_t := \varphi_t(x_0)$ is **distributed according to** μ_t.

Coin Wasserstein Gradient Descent

We will now need a few additional definitions in order to define our algorithm.

- Let $x_0 \sim \mu_0$, for some initial distribution $\mu_0 \in \mathcal{P}_2(\mathbb{R}^d)$
- Let $\varphi_t : \mathbb{R}^d \to \mathbb{R}^d$ denote the function which maps $x_0 \mapsto x_t$.
- Let $\mu_t \in \mathcal{P}_2(\mathbb{R}^d)$ be the **push-forward** of μ_0 under the function $\varphi_t : \mathbb{R}^d \to \mathbb{R}^d$:

$$\mu_t = (\varphi_t)_{\#}\mu_0.$$

This definition means that, given $x_0 \sim \mu_0$, the random variable $x_t := \varphi_t(x_0)$ is **distributed according to** μ_t.

Coin Wasserstein Gradient Descent

We will now need a few additional definitions in order to define our algorithm.

- Let $x_0 \sim \mu_0$, for some initial distribution $\mu_0 \in \mathcal{P}_2(\mathbb{R}^d)$
- Let $\varphi_t : \mathbb{R}^d \to \mathbb{R}^d$ denote the function which maps $x_0 \mapsto x_t$.
- Let $\mu_t \in \mathcal{P}_2(\mathbb{R}^d)$ be the **push-forward** of μ_0 under the function $\varphi_t : \mathbb{R}^d \to \mathbb{R}^d$:

$$\mu_t = (\varphi_t)_{\#}\mu_0.$$

This definition means that, given $x_0 \sim \mu_0$, the random variable $x_t := \varphi_t(x_0)$ is **distributed according to** μ_t.

Coin Wasserstein Gradient Descent

We will now need a few additional definitions in order to define our algorithm.

- Let $x_0 \sim \mu_0$, for some initial distribution $\mu_0 \in \mathcal{P}_2(\mathbb{R}^d)$
- Let $\varphi_t : \mathbb{R}^d \to \mathbb{R}^d$ denote the function which maps $x_0 \mapsto x_t$.
- Let $\mu_t \in \mathcal{P}_2(\mathbb{R}^d)$ be the **push-forward** of μ_0 under the function $\varphi_t : \mathbb{R}^d \to \mathbb{R}^d$:

$$\mu_t = (\varphi_t)_{\#}\mu_0.$$

This definition means that, given $x_0 \sim \mu_0$, the random variable $x_t := \varphi_t(x_0)$ is **distributed according to** μ_t.

Coin Wasserstein Gradient Descent

How do we go from this **betting game** to a **sampling algorithm**?

- Consider a **hypothetical betting game** where $x_0 \sim \mu_0$, and we bet $x_t - x_0$ on $c_t = -\nabla_{W_2}\mathcal{F}(\mu_t)(x_t)$.

- The **sequence of bets** and **betting measures** associated with this betting game are then given by

$$x_t = x_0 - \frac{\sum_{s=1}^{t-1} \nabla_{W_2}\mathcal{F}(\mu_s)(x_s)}{t}\left(w_0 - \sum_{s=1}^{t-1}\langle \nabla_{W_2}\mathcal{F}(\mu_s)(x_s), x_s - x_0\rangle\right)$$

$$\mu_t = (\varphi_t)_{\#}\mu_0.$$

- In the paper, we obtain a sufficient condition under which $\bar{\mu}_T$ **converges to** $\pi = \arg\min_{\mu\in\mathcal{P}_2(\mathbb{R}^d)}\mathcal{F}(\mu)$ at a rate determined by the betting strategy.

Coin Wasserstein Gradient Descent

How do we go from this **betting game** to a **sampling algorithm**?

- Consider a **hypothetical betting game** where $x_0 \sim \mu_0$, and we bet $x_t - x_0$ on $c_t = -\nabla_{W_2}\mathcal{F}(\mu_t)(x_t)$.

- The **sequence of bets** and **betting measures** associated with this betting game are then given by

$$x_t = x_0 - \frac{\sum_{s=1}^{t-1} \nabla_{W_2}\mathcal{F}(\mu_s)(x_s)}{t}\left(w_0 - \sum_{s=1}^{t-1}\langle \nabla_{W_2}\mathcal{F}(\mu_s)(x_s), x_s - x_0\rangle\right)$$

$$\mu_t = (\varphi_t)_{\#}\mu_0.$$

- In the paper, we obtain a sufficient condition under which $\bar{\mu}_T$ **converges to** $\pi = \arg\min_{\mu\in\mathcal{P}_2(\mathbb{R}^d)}\mathcal{F}(\mu)$ at a rate determined by the betting strategy.

Coin Wasserstein Gradient Descent

How do we go from this **betting game** to a **sampling algorithm**?

- Consider a **hypothetical betting game** where $x_0 \sim \mu_0$, and we bet $x_t - x_0$ on $c_t = -\nabla_{W_2}\mathcal{F}(\mu_t)(x_t)$.

- The **sequence of bets** and **betting measures** associated with this betting game are then given by

$$x_t = x_0 - \frac{\sum_{s=1}^{t-1} \nabla_{W_2}\mathcal{F}(\mu_s)(x_s)}{t}\left(w_0 - \sum_{s=1}^{t-1}\langle \nabla_{W_2}\mathcal{F}(\mu_s)(x_s), x_s - x_0\rangle\right)$$

$$\mu_t = (\varphi_t)_{\#}\mu_0.$$

- In the paper, we obtain a sufficient condition under which $\bar{\mu}_T$ **converges to** $\pi = \arg\min_{\mu\in\mathcal{P}_2(\mathbb{R}^d)}\mathcal{F}(\mu)$ at a rate determined by the betting strategy.

Coin Wasserstein Gradient Descent

How do we go from this **betting game** to a **sampling algorithm**?

- Consider a **hypothetical betting game** where $x_0 \sim \mu_0$, and we bet $x_t - x_0$ on $c_t = -\nabla_{W_2}\mathcal{F}(\mu_t)(x_t)$.

- The **sequence of bets** and **betting measures** associated with this betting game are then given by

$$x_t = x_0 - \frac{\sum_{s=1}^{t-1} \nabla_{W_2}\mathcal{F}(\mu_s)(x_s)}{t}\left(w_0 - \sum_{s=1}^{t-1}\langle \nabla_{W_2}\mathcal{F}(\mu_s)(x_s), x_s - x_0\rangle\right)$$

$$\mu_t = (\varphi_t)_{\#}\mu_0.$$

- In the paper, we obtain a sufficient condition under which $\bar{\mu}_T$ **converges to** $\pi = \arg\min_{\mu\in\mathcal{P}_2(\mathbb{R}^d)}\mathcal{F}(\mu)$ at a rate determined by the betting strategy.

Coin Wasserstein Gradient Descent

Suppose that $\mathcal{F} : \mathcal{P}_2(\mathbb{R}^d) \to \mathbb{R}$ is a **geodesically convex** functional, and that $||\nabla_{W_2}\mathcal{F}(\mu_t)(x_t)|| \leq L$ for $t \in [1, T]$.

Then, under this sufficient condition, we can show that

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{L}{T}\left[w_0 + \int_{\mathbb{R}^d} ||x|| \sqrt{T \ln\left(1 + \frac{96K^2T^2||x||^2}{w_0^2}\right)}\pi(\mathrm{d}x) + \int_{\mathbb{R}^d} ||x|| \sqrt{T \ln\left(1 + \frac{96T^2||x||^2}{w_0^2}\right)}\mu_0(\mathrm{d}x)\right].$$

Coin Wasserstein Gradient Descent

Suppose that $\mathcal{F} : \mathcal{P}_2(\mathbb{R}^d) \to \mathbb{R}$ is a **geodesically convex** functional, and that $||\nabla_{W_2}\mathcal{F}(\mu_t)(x_t)|| \leq L$ for $t \in [1, T]$.

Then, under this sufficient condition, we can show that

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{L}{T}\Bigg[w_0 + \int_{\mathbb{R}^d} ||x|| \sqrt{T \ln\left(1 + \frac{96K^2T^2\,||x||^2}{w_0^2}\right)}\,\pi(\mathrm{d}x) + \int_{\mathbb{R}^d} ||x|| \sqrt{T \ln\left(1 + \frac{96T^2\,||x||^2}{w_0^2}\right)}\,\mu_0(\mathrm{d}x)\Bigg].$$

Coin Wasserstein Gradient Descent

In a very simple case, where $\mu_0 \sim \mathcal{N}(m_0, \Sigma)$ and $\pi \sim \mathcal{N}(m_\pi, \Sigma)$, we can improve this bound to

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{L}{T}\left[w_0 + ||m_\pi - m_0||\sqrt{T \ln\left(1 + \frac{24\,T^2\,||m_\pi - m_0||^2}{w_0^2}\right)}\right].$$

Particle-Based Coin Wasserstein Gradient Descent

Recall our update equation from before:

$$x_t = x_0 - \frac{\sum_{s=1}^{t-1} \nabla_{W_2} \mathcal{F}(\mu_s)(x_s)}{t} \left(w_0 - \sum_{s=1}^{t-1} \langle \nabla_{W_2} \mathcal{F}(\mu_s)(x_s), x_s - x_0 \rangle \right).$$

Problem. The updates depend on the **unknown** μ_s, the density of x_s at time s.

Solution. Replace this quantity by the **empirical measure** of n **interacting particles**. Let $x_0^1, \ldots, x_0^N \sim \mu_0$. Then, for $i \in [N]$, writing $\hat{\mu}_t^N = \frac{1}{N} \sum_{i=1}^N \delta_{x_t^i}$, update

$$x_t^i = x_0^i - \frac{\sum_{s=1}^{t-1} \nabla_{W_2} \mathcal{F}(\hat{\mu}_s^N)(x_s^i)}{t} \left(w_0 - \sum_{s=1}^{t-1} \langle \nabla_{W_2} \mathcal{F}(\hat{\mu}_s^N)(x_s^i), x_s^i - x_0^i \rangle \right).$$

Particle-Based Coin Wasserstein Gradient Descent

Recall our update equation from before:

$$x_t = x_0 - \frac{\sum_{s=1}^{t-1} \nabla_{W_2} \mathcal{F}(\mu_s)(x_s)}{t} \left(w_0 - \sum_{s=1}^{t-1} \langle \nabla_{W_2} \mathcal{F}(\mu_s)(x_s), x_s - x_0 \rangle \right).$$

Problem. The updates depend on the **unknown** μ_s, the density of x_s at time s.

Solution. Replace this quantity by the **empirical measure** of n **interacting particles**. Let $x_0^1, \ldots, x_0^N \sim \mu_0$. Then, for $i \in [N]$, writing $\hat{\mu}_t^N = \frac{1}{N} \sum_{i=1}^N \delta_{x_t^i}$, update

$$x_t^i = x_0^i - \frac{\sum_{s=1}^{t-1} \nabla_{W_2} \mathcal{F}(\hat{\mu}_s^N)(x_s^i)}{t} \left(w_0 - \sum_{s=1}^{t-1} \langle \nabla_{W_2} \mathcal{F}(\hat{\mu}_s^N)(x_s^i), x_s^i - x_0^i \rangle \right).$$

Particle-Based Coin Wasserstein Gradient Descent

Recall our update equation from before:

$$x_t = x_0 - \frac{\sum_{s=1}^{t-1} \nabla_{W_2} \mathcal{F}(\mu_s)(x_s)}{t} \left(w_0 - \sum_{s=1}^{t-1} \langle \nabla_{W_2} \mathcal{F}(\mu_s)(x_s), x_s - x_0 \rangle \right).$$

Problem. The updates depend on the **unknown** μ_s, the density of x_s at time s.

Solution. Replace this quantity by the **empirical measure** of n **interacting particles**. Let $x_0^1, \ldots, x_0^N \sim \mu_0$. Then, for $i \in [N]$, writing $\hat{\mu}_t^N = \frac{1}{N} \sum_{i=1}^N \delta_{x_t^i}$, update

$$x_t^i = x_0^i - \frac{\sum_{s=1}^{t-1} \nabla_{W_2} \mathcal{F}(\hat{\mu}_s^N)(x_s^i)}{t} \left(w_0 - \sum_{s=1}^{t-1} \langle \nabla_{W_2} \mathcal{F}(\hat{\mu}_s^N)(x_s^i), x_s^i - x_0^i \rangle \right).$$

Coin Stein Variational Gradient Descent

Our update equation now reads

$$x_t^i = x_0^i - \frac{\sum_{s=1}^{t-1} \nabla_{W_2} \mathcal{F}(\hat{\mu}_s^N)(x_s^i)}{t} \left(w_0 - \sum_{s=1}^{t-1} \langle \nabla_{W_2} \mathcal{F}(\hat{\mu}_s^N)(x_s^i), x_s^i - x_0^i \rangle \right).$$

Problem. If $\mathcal{F}(\mu) = \mathrm{KL}(\mu|\pi)$, then $\nabla_{W_2}\mathcal{F}(\mu) = \nabla \ln \frac{\mu}{\pi}$ is ill defined when $\mu = \hat{\mu}^N$ is a discrete measure.

Solution. Replace $\nabla_{W_2}\mathcal{F}(\mu)$ by $P_\mu \nabla_{W_2}\mathcal{F}(\mu)$, its image under the integral operator $P_\mu f(x) = \int_{\mathbb{R}^d} k(x,y) f(y) \mu(y) \mathrm{d}y$. We thus update

$$x_t^i = x_0^i - \frac{\sum_{s=1}^{t-1} P_{\hat{\mu}_s^N} \nabla \log\left(\frac{\hat{\mu}_s^N}{\pi}\right)(x_s^i)}{t} \cdot \left(w_0^i - \sum_{s=1}^{t-1} \langle P_{\hat{\mu}_s^N} \nabla \log\left(\frac{\hat{\mu}_s^N}{\pi}\right)(x_s^i), x_s^i - x_0^i \rangle \right).$$

Coin Stein Variational Gradient Descent

Our update equation now reads

$$x_t^i = x_0^i - \frac{\sum_{s=1}^{t-1} \nabla_{W_2} \mathcal{F}(\hat{\mu}_s^N)(x_s^i)}{t} \left(w_0 - \sum_{s=1}^{t-1} \langle \nabla_{W_2} \mathcal{F}(\hat{\mu}_s^N)(x_s^i), x_s^i - x_0^i \rangle \right).$$

Problem. If $\mathcal{F}(\mu) = \mathrm{KL}(\mu|\pi)$, then $\nabla_{W_2} \mathcal{F}(\mu) = \nabla \ln \frac{\mu}{\pi}$ is ill defined when $\mu = \hat{\mu}^N$ is a discrete measure.

Solution. Replace $\nabla_{W_2} \mathcal{F}(\mu)$ by $P_\mu \nabla_{W_2} \mathcal{F}(\mu)$, its image under the integral operator $P_\mu f(x) = \int_{\mathbb{R}^d} k(x,y) f(y) \mu(y) \mathrm{d}y$. We thus update

$$x_t^i = x_0^i - \frac{\sum_{s=1}^{t-1} P_{\hat{\mu}_s^N} \nabla \log \left(\frac{\hat{\mu}_s^N}{\pi} \right) (x_s^i)}{t} \cdot \left(w_0^i - \sum_{s=1}^{t-1} \langle P_{\hat{\mu}_s^N} \nabla \log \left(\frac{\hat{\mu}_s^N}{\pi} \right) (x_s^i), x_s^i - x_0^i \rangle \right).$$

Coin Stein Variational Gradient Descent

Our update equation now reads

$$x_t^i = x_0^i - \frac{\sum_{s=1}^{t-1} \nabla_{W_2} \mathcal{F}(\hat{\mu}_s^N)(x_s^i)}{t} \left(w_0 - \sum_{s=1}^{t-1} \langle \nabla_{W_2} \mathcal{F}(\hat{\mu}_s^N)(x_s^i), x_s^i - x_0^i \rangle \right).$$

Problem. If $\mathcal{F}(\mu) = \mathrm{KL}(\mu|\pi)$, then $\nabla_{W_2} \mathcal{F}(\mu) = \nabla \ln \frac{\mu}{\pi}$ is ill defined when $\mu = \hat{\mu}^N$ is a discrete measure.

Solution. Replace $\nabla_{W_2} \mathcal{F}(\mu)$ by $P_\mu \nabla_{W_2} \mathcal{F}(\mu)$, its image under the integral operator $P_\mu f(x) = \int_{\mathbb{R}^d} k(x,y) f(y) \mu(y) \mathrm{d}y$. We thus update

$$x_t^i = x_0^i - \frac{\sum_{s=1}^{t-1} P_{\hat{\mu}_s^N} \nabla \log \left(\frac{\hat{\mu}_s^N}{\pi} \right) (x_s^i)}{t} \cdot \left(w_0^i - \sum_{s=1}^{t-1} \langle P_{\hat{\mu}_s^N} \nabla \log \left(\frac{\hat{\mu}_s^N}{\pi} \right) (x_s^i), x_s^i - x_0^i \rangle \right).$$

Coin Stein Variational Gradient Descent

Key Point. Using integration by parts, we have that

$$\begin{aligned} P_\mu \nabla \log \frac{\mu}{\pi}(x) &= \int k(x,y) \nabla \log \left(\frac{\mu}{\pi}\right)(y) \mu(\mathrm{d}y) \\ &= \int -k(x,y) \nabla \log \pi(y) \mu(\mathrm{d}y) + \int k(x,y) \nabla \mu(\mathrm{d}y) \\ &= \int_{\mathbb{R}^d} \left[k(x,y) \nabla U(y) - \nabla_y k(x,y)\right] \mu(\mathrm{d}y). \end{aligned}$$

Thus, in particular,

$$P_{\hat{\mu}_s^N} \nabla \log \left(\frac{\hat{\mu}_s^N}{\pi}\right)(x_s^i) = \frac{1}{N} \sum_{j=1}^{N} \left[k(x_s^i, x_s^j) \nabla U(x_s^j) - \nabla_2 k(x_s^i, x_s^j)\right].$$

Coin Stein Variational Gradient Descent

Key Point. Using integration by parts, we have that

$$\begin{aligned} P_\mu \nabla \log \frac{\mu}{\pi}(x) &= \int k(x,y) \nabla \log \left(\frac{\mu}{\pi}\right)(y) \mu(\mathrm{d}y) \\ &= \int -k(x,y) \nabla \log \pi(y) \mu(\mathrm{d}y) + \int k(x,y) \nabla \mu(\mathrm{d}y) \\ &= \int_{\mathbb{R}^d} \left[k(x,y) \nabla U(y) - \nabla_y k(x,y)\right] \mu(\mathrm{d}y). \end{aligned}$$

Thus, in particular,

$$P_{\hat{\mu}_s^N} \nabla \log \left(\frac{\hat{\mu}_s^N}{\pi}\right)(x_s^i) = \frac{1}{N} \sum_{j=1}^{N} \left[k(x_s^i, x_s^j) \nabla U(x_s^j) - \nabla_2 k(x_s^i, x_s^j)\right].$$

Coin SVGD vs SVGD

SVGD is a particle approximation of **Wasserstein gradient descent** for $\mathcal{F}(\mu) = \mathrm{KL}(\mu||\pi)$, with $\nabla_{W_2}\mathcal{F}(\mu)$ replaced by $P_{\hat{\mu}^N}\nabla_{W_2}\mathcal{F}(\hat{\mu}^N)$. This gives

$$x_t^i = x_{t-1}^i - \gamma P_{\hat{\mu}_{t-1}^N} \nabla \log\left(\frac{\hat{\mu}_{t-1}^N}{\pi}\right)(x_{t-1}^i).$$

Coin SVGD is a particle approximation of **coin Wasserstein gradient descent** for $\mathcal{F}(\mu) = \mathrm{KL}(\mu||\pi)$, with $\nabla_{W_2}\mathcal{F}(\mu)$ replaced by $P_{\hat{\mu}^N}\nabla_{W_2}\mathcal{F}(\hat{\mu}^N)$. This gives

$$x_t^i = x_0^i - \frac{\sum_{s=1}^{t-1} P_{\hat{\mu}_s^N} \nabla \log\left(\frac{\hat{\mu}_s^N}{\pi}\right)(x_s^i)}{t} \cdot \left(w_0^i - \sum_{s=1}^{t-1} \langle P_{\hat{\mu}_s^N} \nabla \log\left(\frac{\hat{\mu}_s^N}{\pi}\right)(x_s^i), x_s^i - x_0^i \rangle\right).$$

Coin SVGD vs SVGD

SVGD is a particle approximation of **Wasserstein gradient descent** for $\mathcal{F}(\mu) = \mathrm{KL}(\mu||\pi)$, with $\nabla_{W_2}\mathcal{F}(\mu)$ replaced by $P_{\hat{\mu}^N}\nabla_{W_2}\mathcal{F}(\hat{\mu}^N)$. This gives

$$x_t^i = x_{t-1}^i - \gamma P_{\hat{\mu}_{t-1}^N} \nabla \log\left(\frac{\hat{\mu}_{t-1}^N}{\pi}\right)(x_{t-1}^i).$$

Coin SVGD is a particle approximation of **coin Wasserstein gradient descent** for $\mathcal{F}(\mu) = \mathrm{KL}(\mu||\pi)$, with $\nabla_{W_2}\mathcal{F}(\mu)$ replaced by $P_{\hat{\mu}^N}\nabla_{W_2}\mathcal{F}(\hat{\mu}^N)$. This gives

$$x_t^i = x_0^i - \frac{\sum_{s=1}^{t-1} P_{\hat{\mu}_s^N} \nabla \log\left(\frac{\hat{\mu}_s^N}{\pi}\right)(x_s^i)}{t} \cdot \left(w_0^i - \sum_{s=1}^{t-1} \langle P_{\hat{\mu}_s^N} \nabla \log\left(\frac{\hat{\mu}_s^N}{\pi}\right)(x_s^i), x_s^i - x_0^i \rangle\right).$$

Coin KSDD and Coin LAWGD

By varying the **choice of** $\mathcal{F}$, or using different **approximations for** $\nabla_{W_2}\mathcal{F}(\mu)$, we can obtain coin analogues of other particle-based sampling algorithms.

In the paper [SN23], we also consider

- Coin kernel Stein discrepancy descent (KSDD) [KPCAF+21].
- Coin Laplacian adjusted Wasserstein gradient descent (LAWGD) [CLL+20].

Coin KSDD and Coin LAWGD

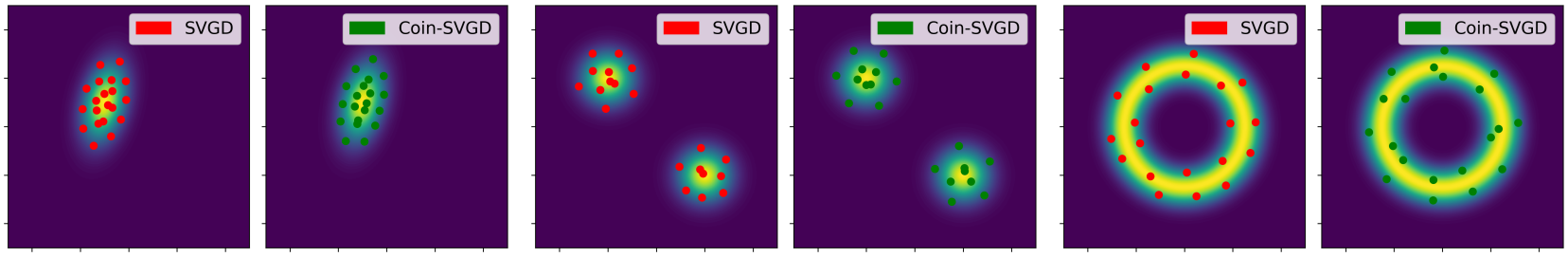
By varying the **choice of** $\mathcal{F}$, or using different **approximations for** $\nabla_{W_2}\mathcal{F}(\mu)$, we can obtain coin analogues of other particle-based sampling algorithms.

In the paper [SN23], we also consider

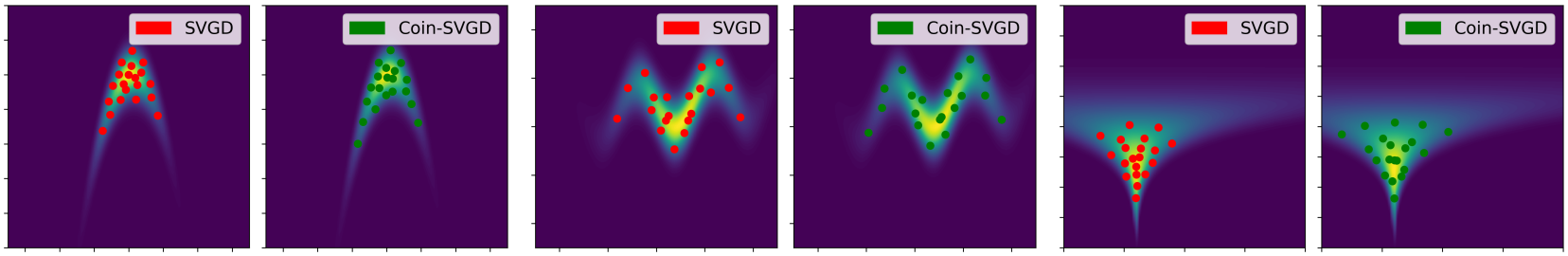
- Coin kernel Stein discrepancy descent (KSDD) [KPCAF$^+$21].
- Coin Laplacian adjusted Wasserstein gradient descent (LAWGD) [CLL$^+$20].

Numerical Results

Toy Examples



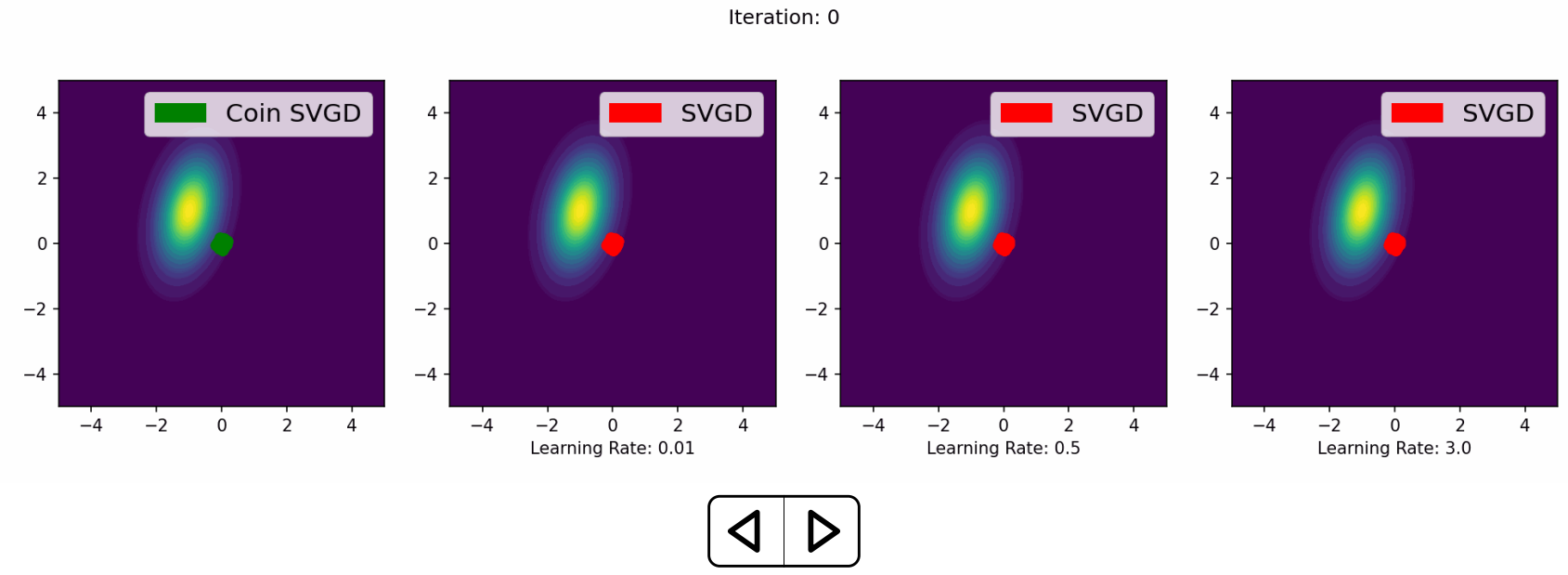
(a) Gaussian. (b) Mixture of Gaussians. (c) 'Donut'.



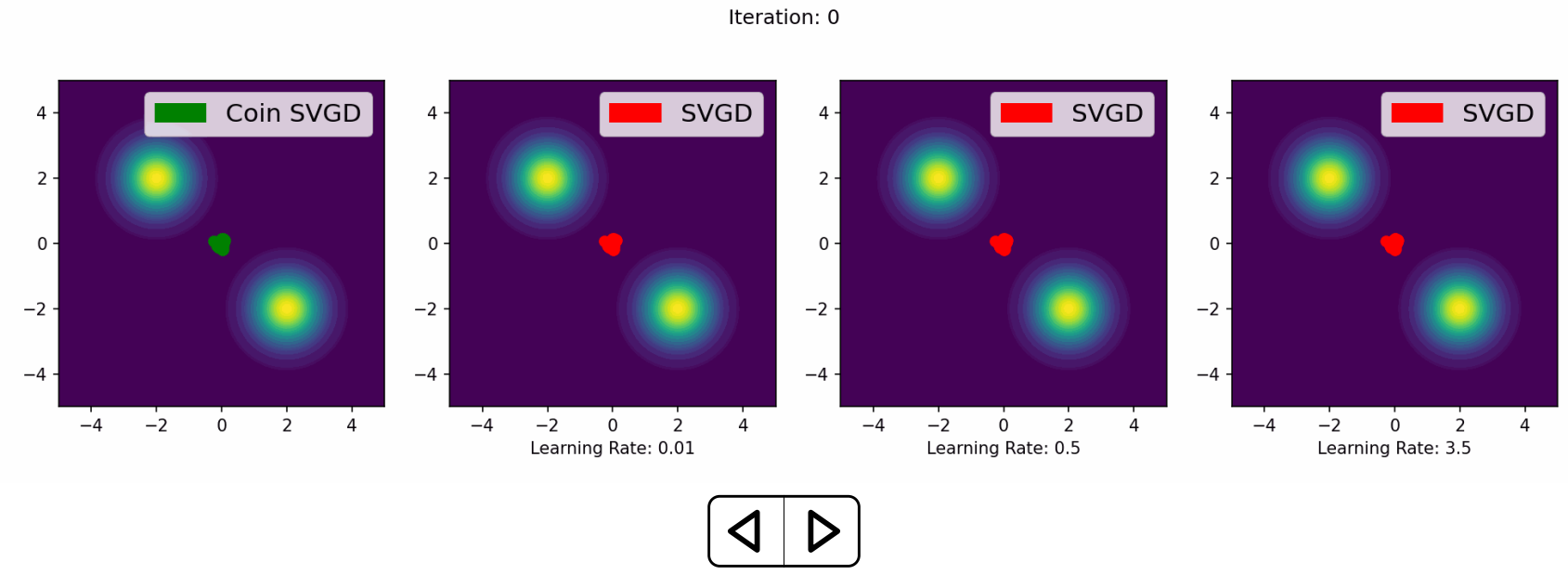
(d) Rosenbrock. (e) 'Squiggle'. (f) 'Funnel'.

Figure: **A comparison between SVGD [LW16] and its learning-rate free analogue, Coin SVGD**.

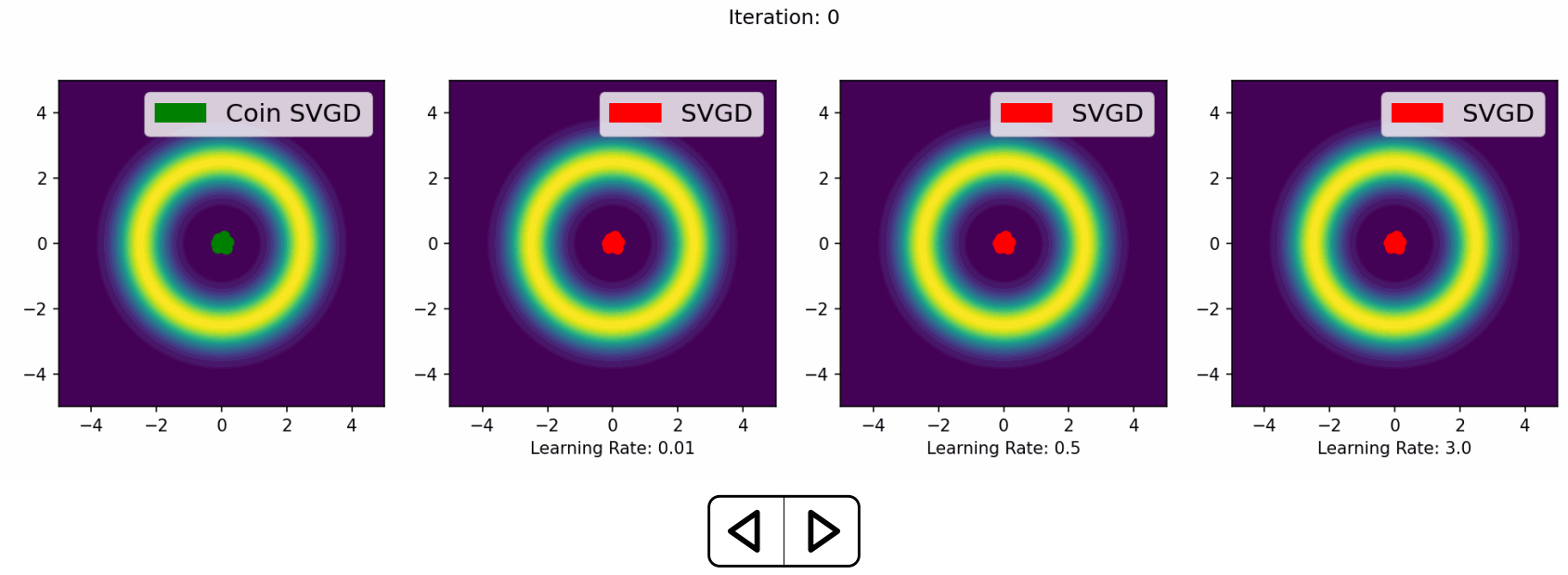
Toy Examples



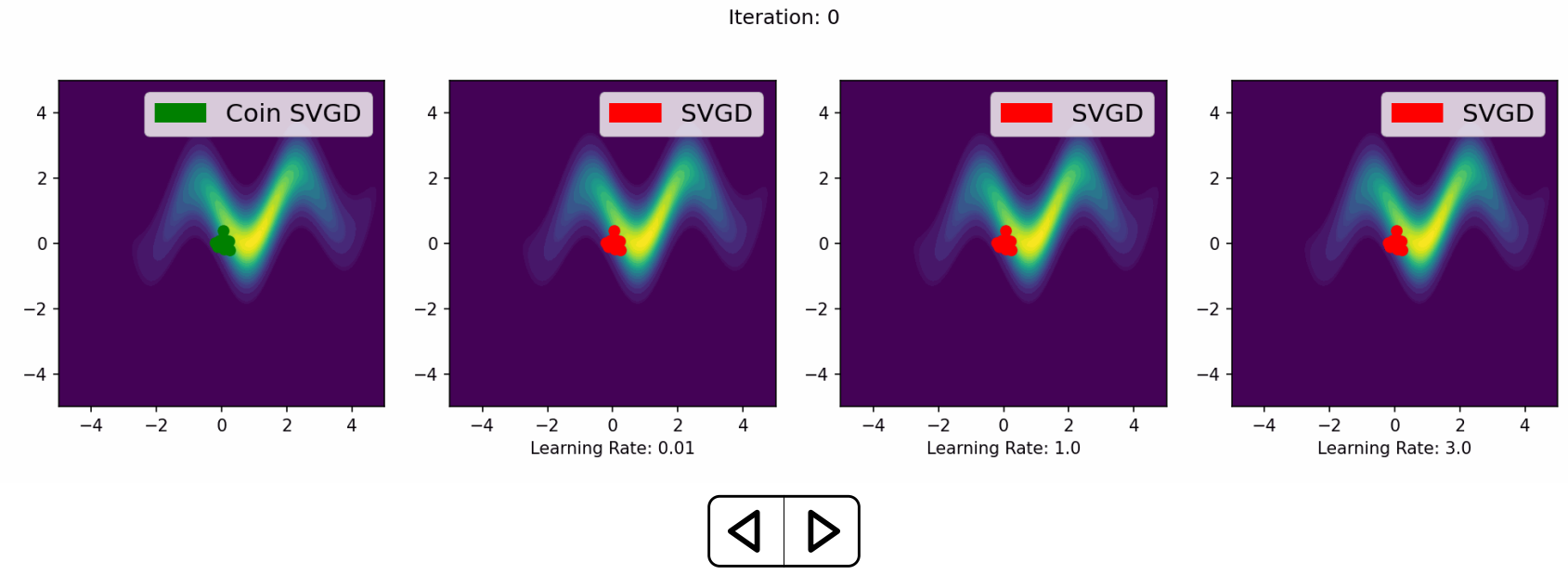
Toy Examples



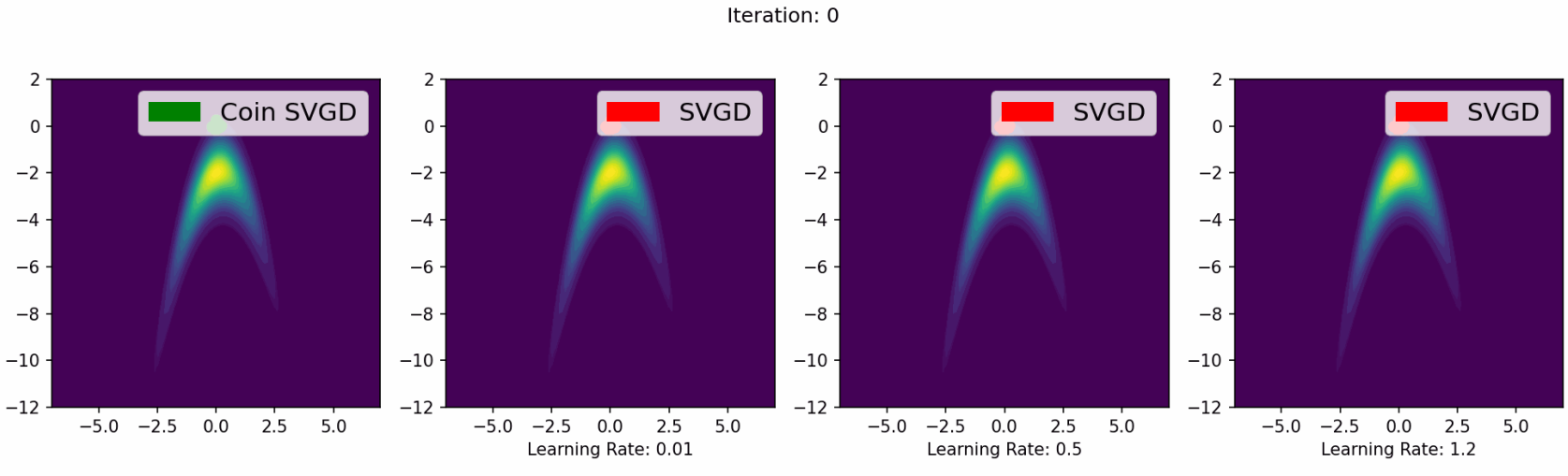
Toy Examples



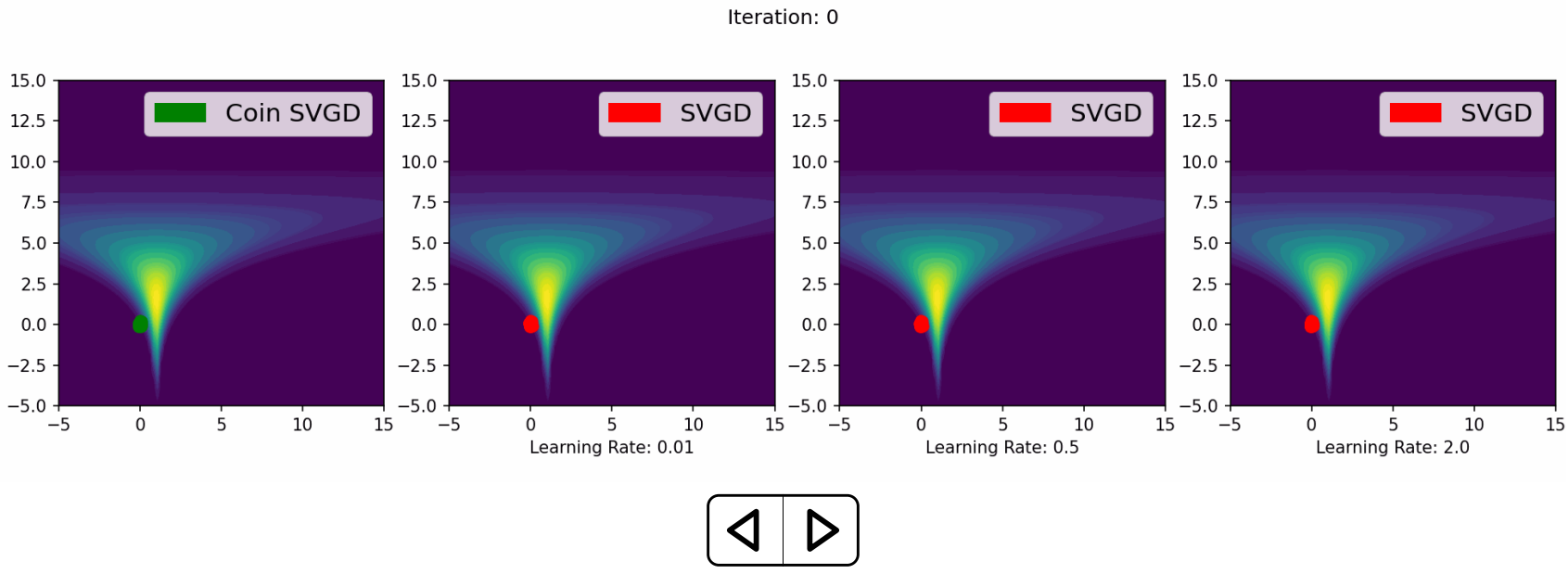
Toy Examples



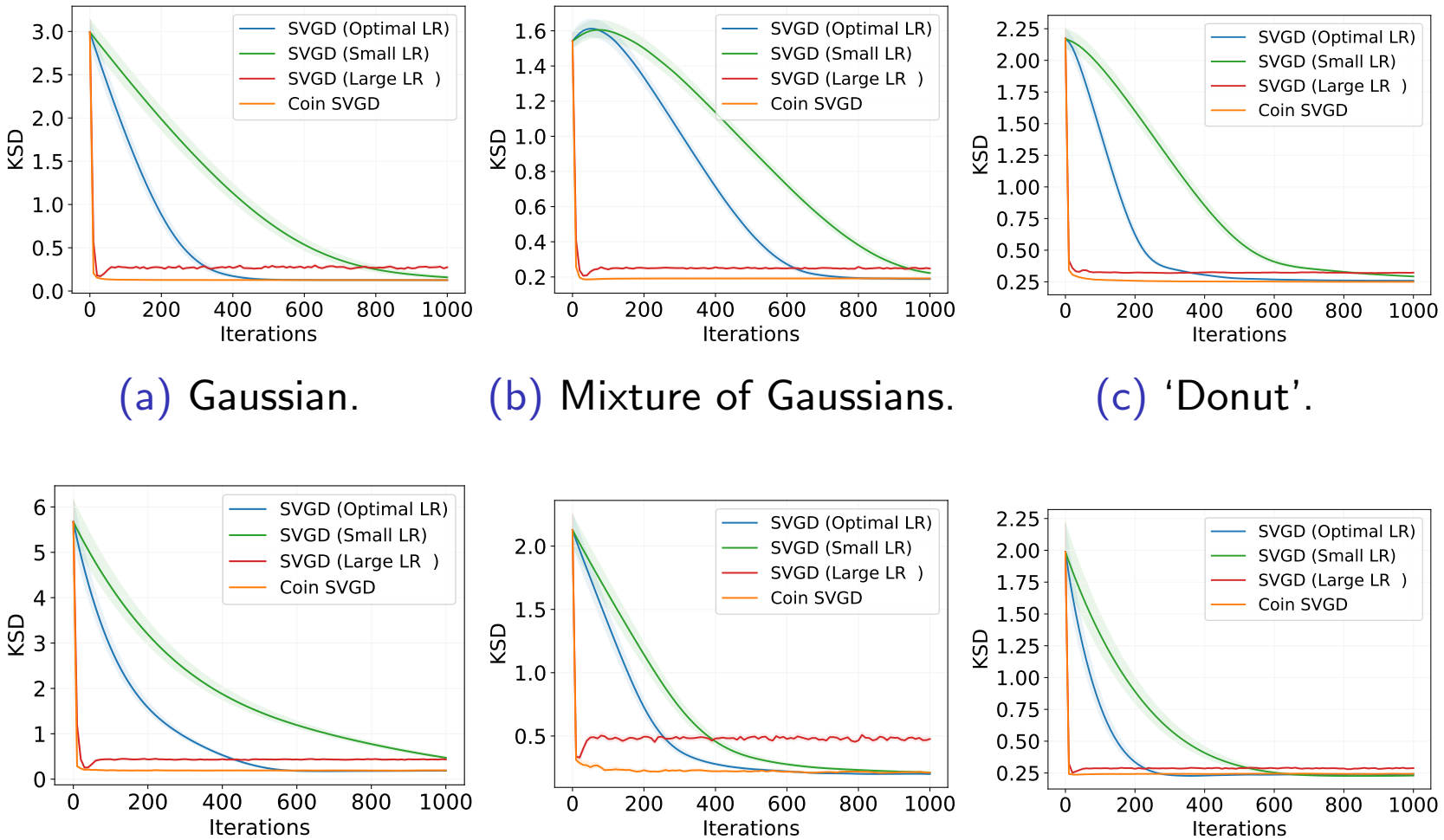
Toy Examples



Toy Examples



Toy Examples

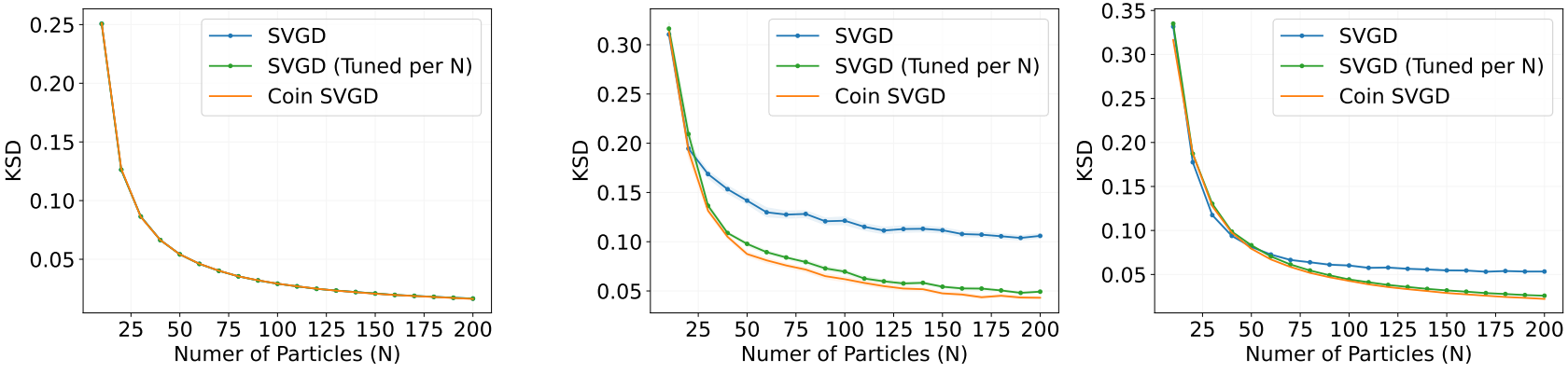


(a) Gaussian. (b) Mixture of Gaussians. (c) 'Donut'.

(d) Rosenbrock. (e) 'Squiggle'. (f) 'Funnel'.

Figure: **Kernel Stein Discrepancy (KSD) vs Iterations (T).**

Toy Examples



(a) Gaussian. (b) Mixture of Gaussians. (c) Rosenbrock.

Figure: **Kernel Stein Discrepancy (KSD) vs Number of Particles (N).**

Independent Component Analysis

Suppose we observe $\boldsymbol{x} \in \mathbb{R}^p$. The task of ICA is to infer the **unmixing matrix** $\mathbf{W} \in \mathbb{R}^{p \times p}$ such that $\boldsymbol{x} = \mathbf{W}^{-1}\boldsymbol{s}$, where $\boldsymbol{s} \in \mathbb{R}^p$ denote latent independent sources.

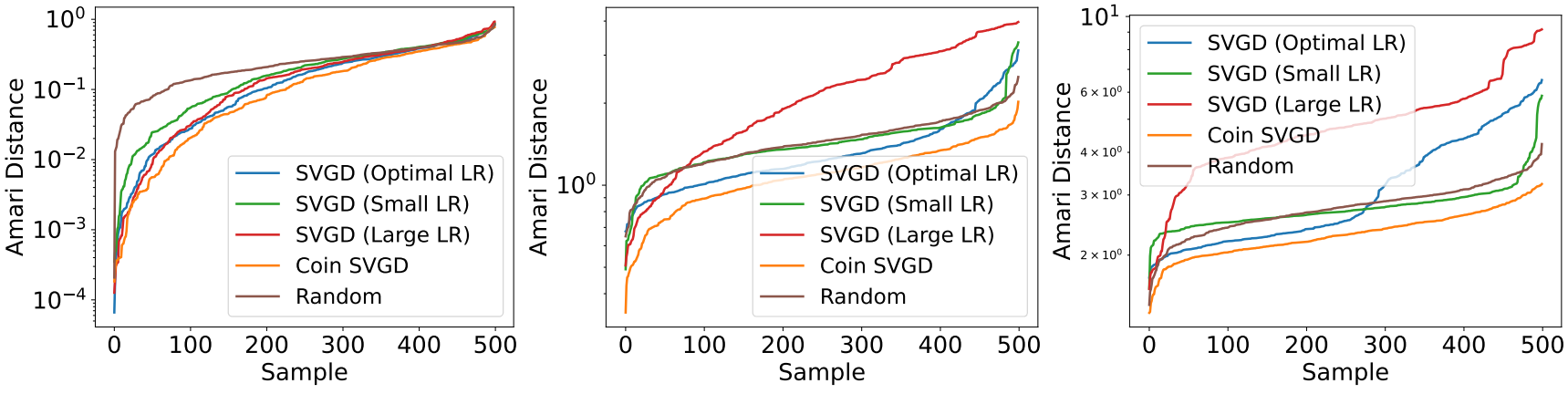
We assume each component s_i has the same density: $s_i \sim p_s$. The log-likelihood is then $\log p(\boldsymbol{x}|\mathbf{W}) = \log|\mathbf{W}| + \sum_{i=1}^{p} p_s([\mathbf{W}\boldsymbol{x}]_i)$.

For the prior, we assume that the entries of $\mathbf{W}$ are i.i.d., with law $\mathcal{N}(0,1)$. The posterior is then $p(\mathbf{W}|\boldsymbol{x}) \propto p(\boldsymbol{x}|\mathbf{W})p(\mathbf{W})$, with

$$\nabla_{\mathbf{W}} \log p(\mathbf{W}|\boldsymbol{x}) = (\mathbf{W}^{-1})^T - \frac{p_s'(\mathbf{W}\boldsymbol{x})}{p_s(\mathbf{W}\boldsymbol{x})}\boldsymbol{x}^T - \mathbf{W}$$

where p_s is chosen such that $\frac{p_s'(\cdot)}{p_s(\cdot)} = \tanh(\cdot)$ (see [KPCAF+21]).

Independent Component Analysis



(a) $p = 2$ (b) $p = 4$ (c) $p = 8$

Figure: **Results for the Bayesian ICA model**: Amari distances between the true unmixing matrix and the estimated unmixing matrices output by SVGD and Coin SVGD (lower is better).

Bayesian Logistic Regression

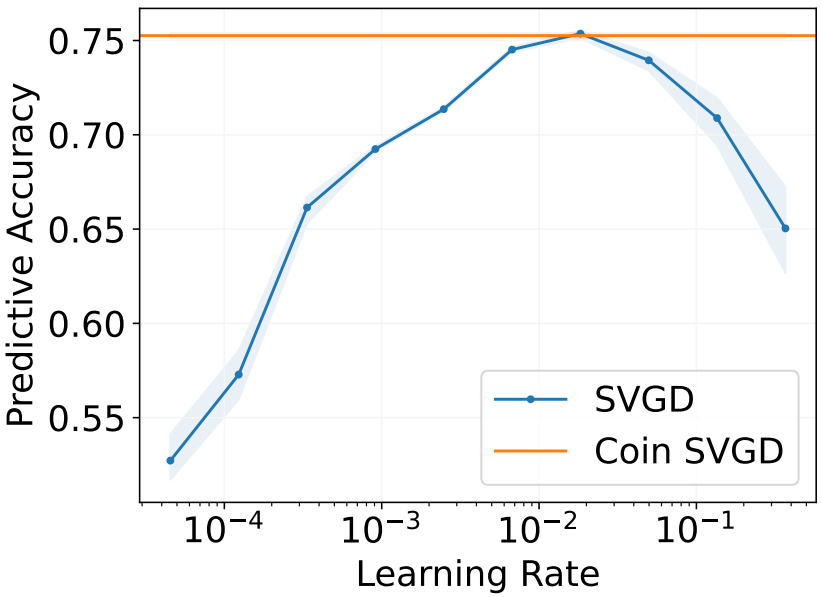
Let $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$ be a dataset with feature vectors $\boldsymbol{x}_i \in \mathbb{R}^p$, and binary labels $y_i \in \{-1, 1\}$.

We assume that $p(y_i = 1|\boldsymbol{x}_i, \boldsymbol{w}) = (1 + \exp(-\boldsymbol{w}^T \boldsymbol{x}_i))^{-1}$, for some $\boldsymbol{w} \in \mathbb{R}^p$.

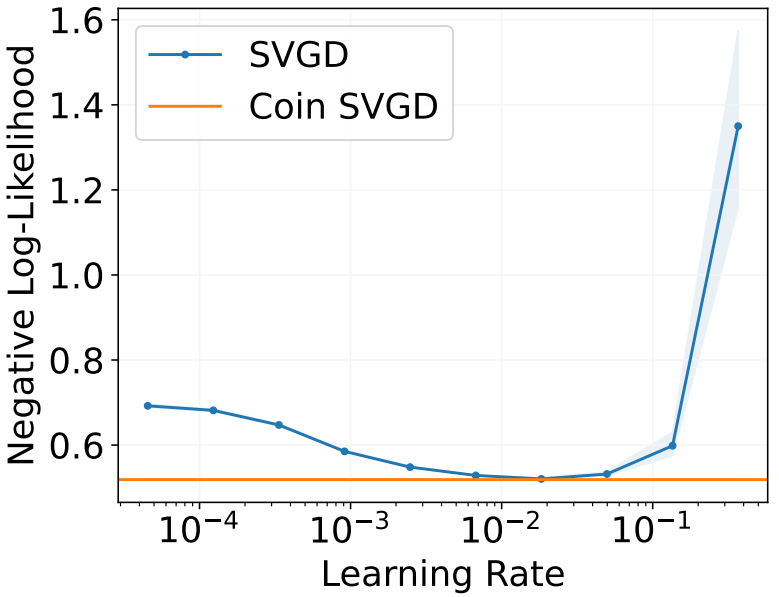
We put a **Gaussian prior** $p(\boldsymbol{w}|\alpha) = \mathcal{N}(\boldsymbol{w}|0, \alpha^{-1})$ on the weights $\boldsymbol{w}$, and a **Gamma prior** $p(\alpha) = \mathrm{Gamma}(\alpha|1, 0.01)$ on the precision $\alpha \in \mathbb{R}_+$.

We test our algorithm using the **Covertype dataset**, which consists of 581,012 data points and 54 features.

Bayesian Logistic Regression



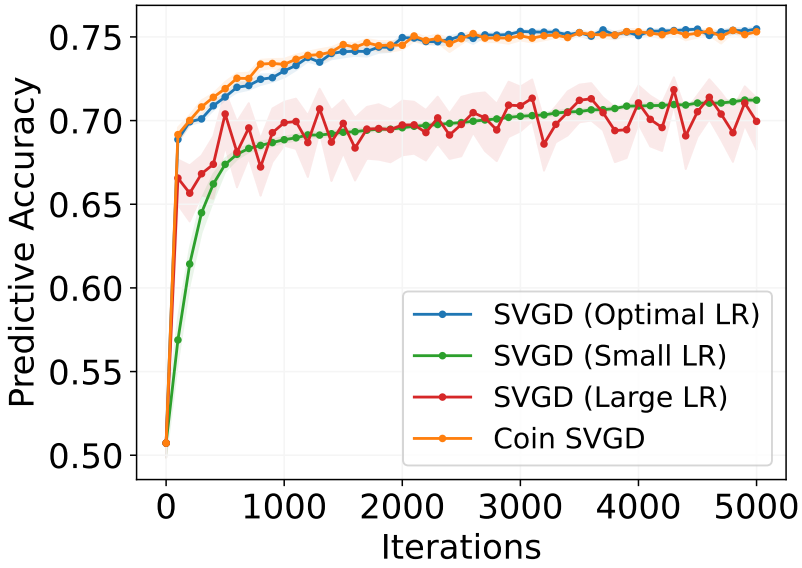
(a) Test Accuracy



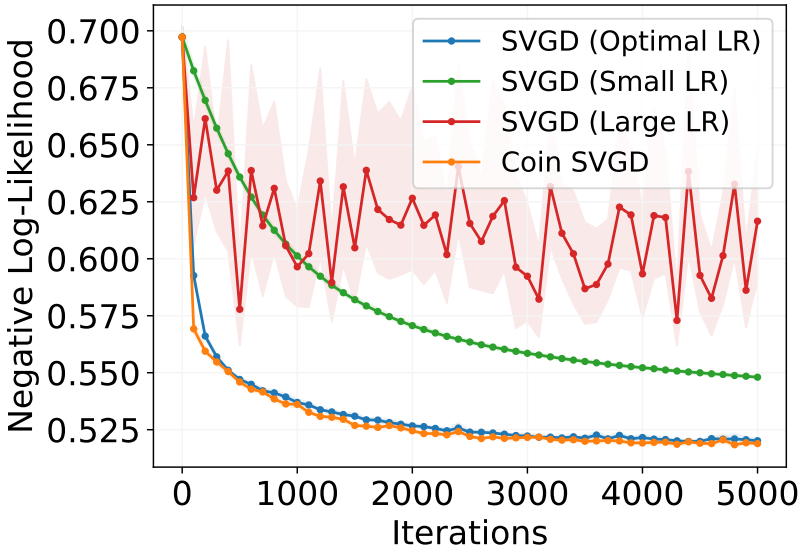
(b) Negative Log-Likelihood

Figure: **Results for the Bayesian logistic regression model**: test-accuracy and the log-likelihood for SVGD and Coin SVGD.

Bayesian Logistic Regression



(a) Test Accuracy



(b) Negative Log-Likelihood

Figure: **Results for the Bayesian logistic regression model**: test-accuracy and the log-likelihood for SVGD (three learning rates) and Coin SVGD as a function of the number of iterations.

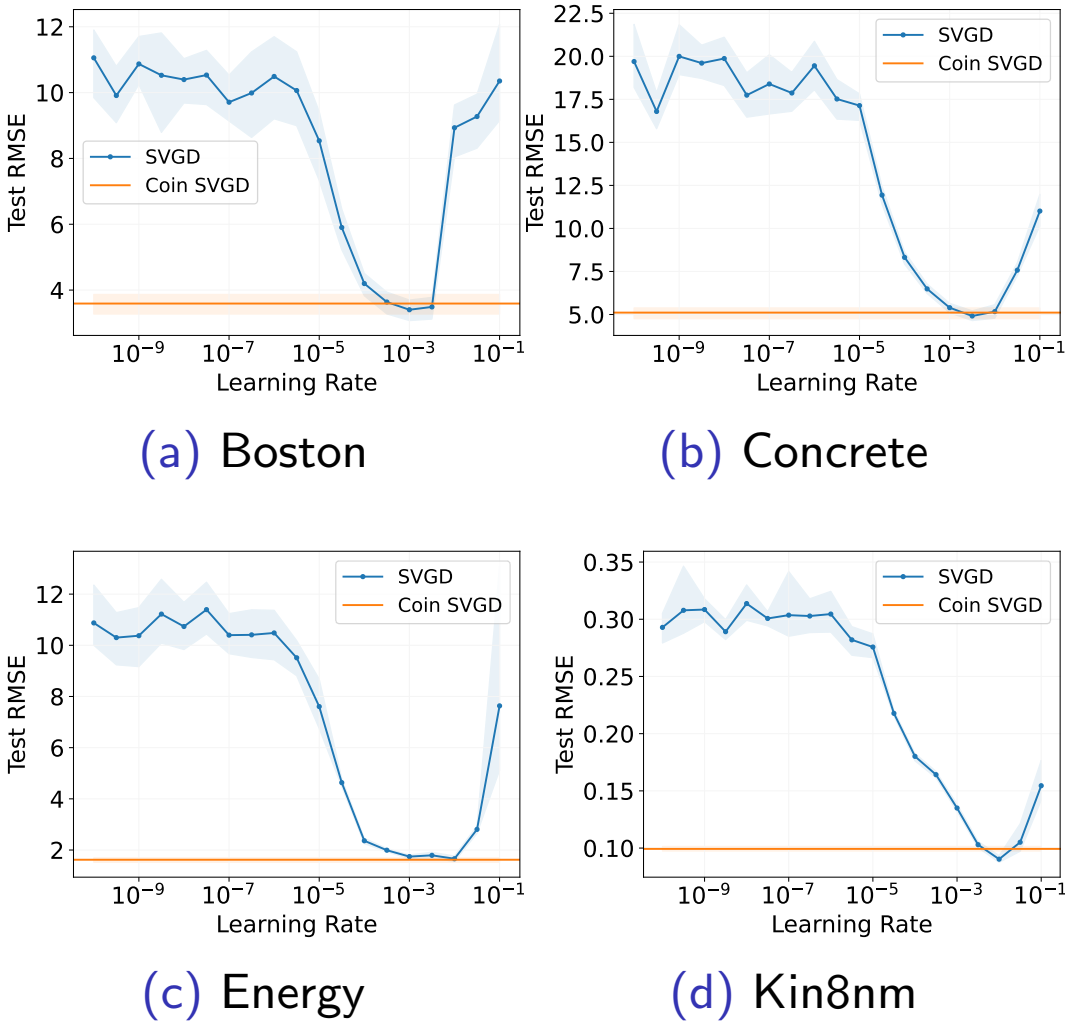
Bayesian Neural Network

We use a **two-layer neural network** with 50 hidden units with $\mathrm{RELU}(x) = \max(0, x)$ as the activation function (see also [LW16]).

We assume the output is **normal**, and place a $\mathrm{Gamma}(1, 0.1)$ prior on the **inverse covariance**. Meanwhile, we assign an **isotropic Gaussian prior** to the **neural network weights**.

We test the performance of our algorithm on several benchmark **UCI datasets**.

Bayesian Neural Network



(a) Boston (b) Concrete

(c) Energy (d) Kin8nm

Figure: **Results for the Bayesian neural network**: test RMSE for SVGD and Coin SVGD after $T = 2000$ iterations for four UCI benchmark datasets.

Probabilistic Matrix Factorisation

Let $\mathbf{R} \in \mathbb{R}^{N \times M}$ be a matrix of ratings for N users and M movies, where R_{ij} is the rating user i gave to movie j.

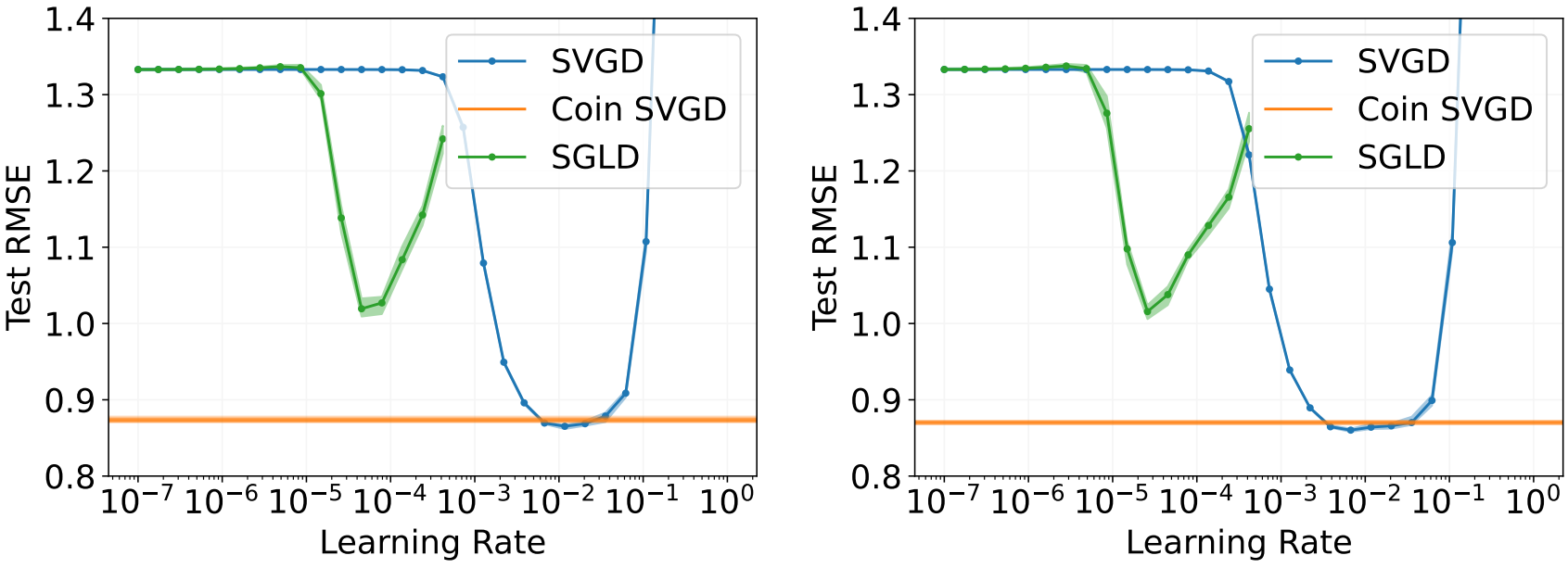
Define matrices $\mathbf{U}$ and $\mathbf{V}$ for users and movies, respectively, where $\mathbf{U}_i \in \mathbb{R}^d$ and $\mathbf{V}_j \in \mathbb{R}^d$ are latent feature vectors for user i and movie j. The **likelihood for the rating matrix** is given by

$$p(\mathbf{R}|\mathbf{U}, \mathbf{V}, \alpha) = \prod_{i=1}^{N} \prod_{j=1}^{M} \left[\mathcal{N}(R_{ij}|\mathbf{U}_i^T \mathbf{V}_j, \alpha^{-1}) \right]^{I_{ij}}$$

where I_{ij} denotes an indicator variable which equals 1 if users i gave a rating for movie j. See paper for further details of the setup.

We test our algorithm on the **MovieLens dataset** [HK15], which consists of 100,000 ratings, taking values $\{1,2,3,4,5\}$, for 1,682 movies from 943 users.

Probabilistic Matrix Factorisation



(a) $T = 1000$. (b) $T = 2000$.

Figure: **Results for the Bayesian probabilistic matrix factorisation model**: RMSE for SGLD, SVGD, and Coin SVGD.

Conclusions

Summary

The efficiency of many sampling algorithms is dependent on the **learning rate**.

By leveraging ideas from convex optimisation, namely **coin betting**, we can obtain **learning-rate free analogues** of many existing particle-based algorithms (e.g., SVGD, KSDD, etc.)

Coin sampling algorithms are **easy to implement**, and offer **performance comparable to the optimal performance** of existing particle-based algorithms.

Extensions

Recently we have extended this work in a couple of different directions:

- Learning-rate free Bayesian inference in **constrained domains** [SMN23].
- Learning-rate free **marginal maximum likelihood training** of **latent variable models** [SDN23].

In both cases, the key idea is to reformulate the problem as an **optimisation problem on the space of probability measures**, before applying our techniques.

Future Work

There are a few interesting directions for future work:

- Which other particle-based sampling algorithms can be **'coinified'**?
- Is there a learning-rate free analogue of LMC based on coin betting?
- Can we improve the convergence rate using other **betting strategies**?
- Is it possible to establish convergence under more **standard assumptions** (e.g., LSI)?
- Can we use similar ideas to develop learning-rate free algorithms for sampling in **constrained domains**?

Some References

- L. Sharrock and C. Nemeth (2023). Coin Sampling: Gradient-Based Bayesian Inference without Learning Rates. *Proceedings of the 40th International Conference on Machine Learning (ICML)*. arXiv: 2301.11294.
- L. Sharrock, L. Mackey, and C. Nemeth (2023). Learning Rate Free Bayesian Inference in Constrained Domains. arXiv: 2305.14943.
- L. Sharrock, D. Dodd, and C. Nemeth (2023). CoinEM: Tuning-Free Particle-Based Variational Inference for Latent Variable Models. arXiv: 2305.14916.

Any Questions?

Bibliography I

L. Ambrosio, N. Gigli, and Giuseppe Savaré.
Gradient Flows: In Metric Spaces and in the Space of Probability Measures.
Birkhäuser, Basel, 2008.

M. Arbel, A. Korba, A. Salim, and A. Gretton.
Maximum Mean Discrepancy Gradient Flow.
In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS 2019)*, Vancouver, Canada, 2019.

H. Brézis.
Opérateurs maximaux monotones et semi-groupes de contractions dans les espaces de Hilbert.
Elsevier Science, Burlington, MA, 1973.

Bibliography II


C. Chen, D. Carlson, Z. Gan, C. Li, and L. Carin.
Bridging the Gap between Stochastic Gradient MCMC and Stochastic Optimization.
In *Proceedings of the 19th International Conference on Aritificial Intelligence and Statistics (AISTATS 2016)*, Cadiz, Spain, 2016.


S. Chewi, T. Le Gouic, C. Lu, T. Maunu, and P. Rigollet.
SVGD as a kernelized Wasserstein gradient flow of the chi-squared divergence.
In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 2020)*, pages 2098–2109, 2020.

Bibliography III


A. S. Dalalyan.
Further and stronger analogy between sampling and optimization: Langevin Monte Carlo and gradient descent.
In *Proceedings of the 30th Conference on Learning Theory (COLT 2017)*, Amsterdam, The Netherlands, 2017.


E. De Giorgi.
New problems on minimizing movements.
In *Boundary Value Problems for PDE and Applications*, pages 81–98. Masson, Paris, 1993.


A. Durmus and É. Moulines.
Nonasymptotic convergence analysis for the unadjusted Langevin algorithm.
The Annals of Applied Probability, 27(3):1551–1587, 2017.

Bibliography IV


A. Durmus and É. Moulines.
High-dimensional Bayesian inference via the unadjusted Langevin algorithm.
Bernoulli, 25(4A):2854–2882, 2019.


A. Durmus, S. Majewski, and B. Miasojedow.
Analysis of Langevin Monte Carlo via Convex Optimization.
Journal of Machine Learning Research, 20(1):2666–2711, 2019.


A. Duncan, N. Nüsken, and L. Szpruch.
On the geometry of Stein variational gradient descent.
Journal of Machine Learning Research, 24:1–40, 2023.

Bibliography V


W. Guo, Y. Hur, T. Liang, and C. T. Ryan.
Online Learning to Transport via the Minimal Selection Principle.
In *Proceedings of the 35th Annual Conference on Learning Theory (COLT 2022)*, London, UK, 2022.


O. Güler.
On the Convergence of the Proximal Point Algorithm for Convex Minimization.
SIAM Journal on Control and Optimization, 29(2):403–419, mar 1991.


F. M. Harper and J. A. Konstan.
The MovieLens Datasets: History and Context.
ACM Transactions on Interactive Intelligent Systems, 5(4), 2015.

Bibliography VI

R. Jordan, D. Kinderlehrer, and F. Otto.
The Variational Formulation of the Fokker–Planck Equation.
SIAM Journal on Mathematical Analysis, 29(1):1–17, 1998.

J. L. Kelly.
A new interpretation of information rate.
The Bell System Technical Journal, 35(4):917–926, 1956.

A. Korba, Pierre-Cyril, Aubin-Frankowski, S. Majewski, and P. Ablin.
Kernel Stein Discrepancy Descent.
In *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)*, Online, 2021.

Bibliography VII


A. Korba, A. Salim, M. Arbel, G. Luise, and A. Gretton.
A Non-Asymptotic Analysis for Stein Variational Gradient Descent.
In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 2020)*, Vancouver, Canada, 2020.


S. Kim, Q. Song, and F. Liang.
Stochastic gradient Langevin dynamics with adaptive drifts.
Journal of Statistical Computation and Simulation, 92(2):318–336, 2022.


R. E. Krichevsky and V. K. Trofimov.
The performance of universal encoding.
IEEE Transactions on Information Theory, 27(2):199–207, 1981.

Bibliography VIII


C. Li, C. Chen, D. Carlson, and L. Carin.
Preconditioned Stochastic Gradient Langevin Dynamics for Deep Neural Networks.
In *Proceedings of the 30th AAAI Conference on Aritificial Intelligence (AAAI-16)*, Phoenix, AZ, 2016.


Q. Liu and D. Wang.
Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm.
In *Proceedings of the 30th Conference on Neural Information Processings Systems (NIPS 2016)*, Barcelona, Spain, 2016.


F. Orabona and D. Pal.
Coin Betting and Parameter-Free Online Learning.
In *Proceedings of the 30th Conference on Neural Information Processings Systems (NIPS 2016)*, Barcelona, Spain, 2016.

Bibliography IX


J. Peypouquet and S. Sorin.
Evolution Equations for Maximal Monotone Operators: Asymptotic Analysis in Continuous and Discrete Time.
Journal of Convex Analysis, 17(3-4):1113–1163, 2010.


L. Sharrock, D. Dodd, and C. Nemeth.
CoinEM: Tuning-Free Particle-Based Variational Inference for Latent Variable Models.
arXiv preprint, 2023.


N. Z. Shor.
Minimization Methods for Non-Differentiable Functions.
Springer, Berlin, Heidelberg, 1985.


L. Sharrock, L. Mackey, and C. Nemeth.
Learning Rate Free Bayesian Inference in Constrained Domains.
arXiv preprint, 2023.

Bibliography X

L. Sharrock and C. Nemeth.
Coin Sampling: Gradient-Based Bayesian Inference without Learning Rates.
To appear in Proceedings of the 40th International Conference on Machine Learning (ICML 2023), 2023.

L. Sun and P. Richtárik.
Improved Stein Variational Gradient Descent with Importance Weights.
arXiv preprint, 2022.

A. Salim, L. Sun, and Peter Richtárik.
A Convergence Theory for SVGD in the Population Limit under Talagrand's Inequality T1.
In *Proceedings of the 39th International Conference on Machine Learning (ICML 2022)*, Online, 2022.

Bibliography XI


A. Wibisono.
Sampling as optimization in the space of measures: The Langevin dynamics as a composite optimization problem.
In *Proceedings of the 31st Annual Conference on Learning Theory (COLT 2018)*, Stockholm, Sweden, 2018.


M. Zinkevich.
Online Convex Programming and Generalized Infinitesimal Gradient Ascent.
In *Proceedings of the 20th International Conference on Machine Learning (ICML 2003)*, pages 928–935, Washington DC, 2003.