

# Lecture 7: Energy Based Models

## Deep Generative Models

Sajjad Amini

Department of Electrical Engineering  
Sharif University of Technology

# Contents

- 1 Intuition
- 2 Why EBMs
- 3 Sampling EBM
- 4 Contrastive Divergence
- 5 Score Matching
  - Denoising Score Matching
  - Sliced Score Matching
- 6 Noise Conditional Score Networks

# Section 1

## Intuition

# Intuition

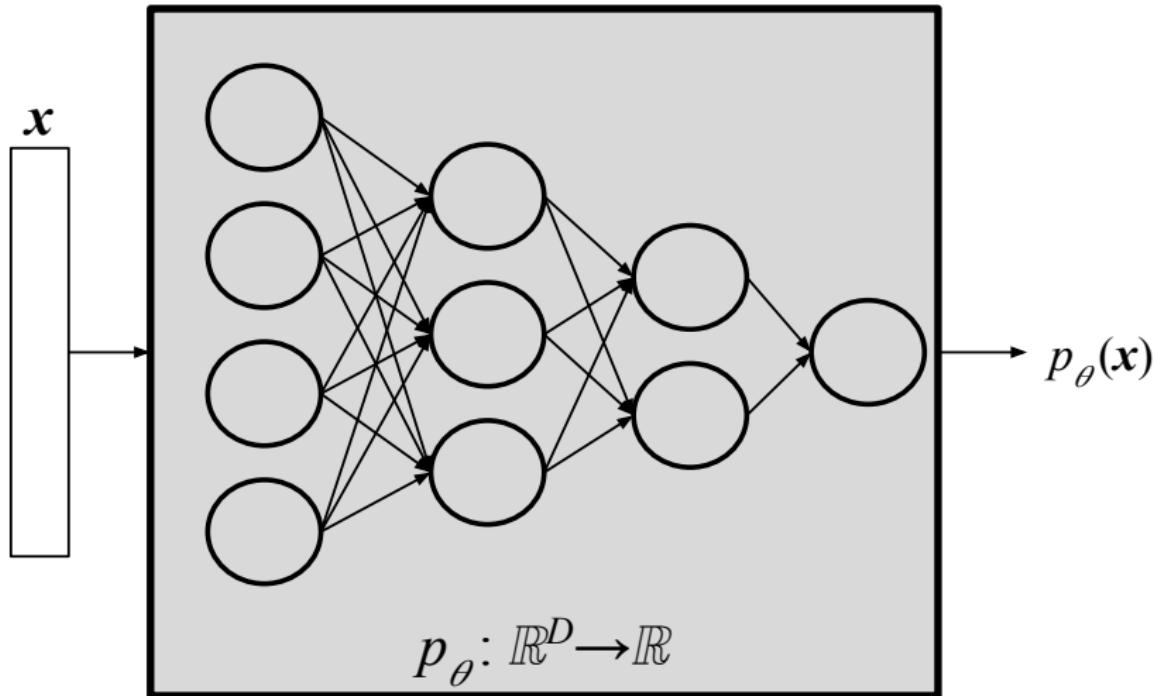


Figure: General idea for Energy Based Models

## PDF Function Properties

Any probability density function must satisfy the following properties:

- Nonnegativity:

$$p_{\theta}(\boldsymbol{x}) \geq 0, \quad \forall \boldsymbol{x}$$

## PDF Function Properties

Any probability density function must satisfy the following properties:

- Nonnegativity:

$$p_{\theta}(\boldsymbol{x}) \geq 0, \quad \forall \boldsymbol{x}$$

- Unit volume:

$$\int_{\boldsymbol{x}} p_{\theta}(\boldsymbol{x}) d\boldsymbol{x} = 1$$

# Nonnegativity

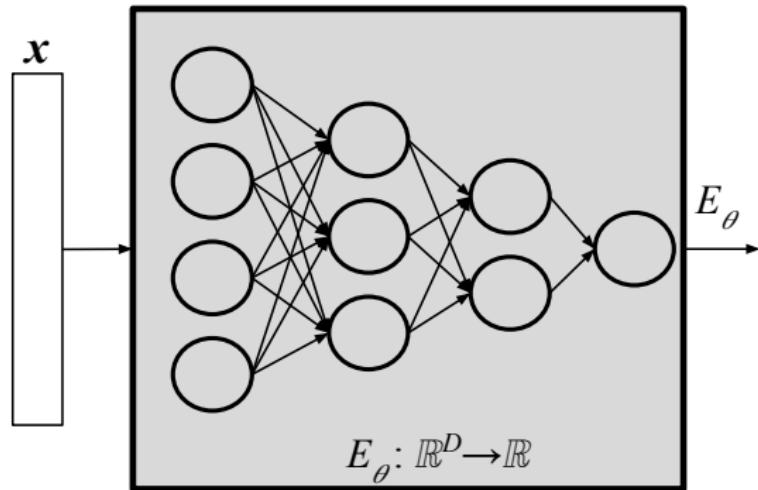


Figure:  $p_{\theta}(\mathbf{x})$  which satisfies the nonnegativity constraint

# Nonnegativity

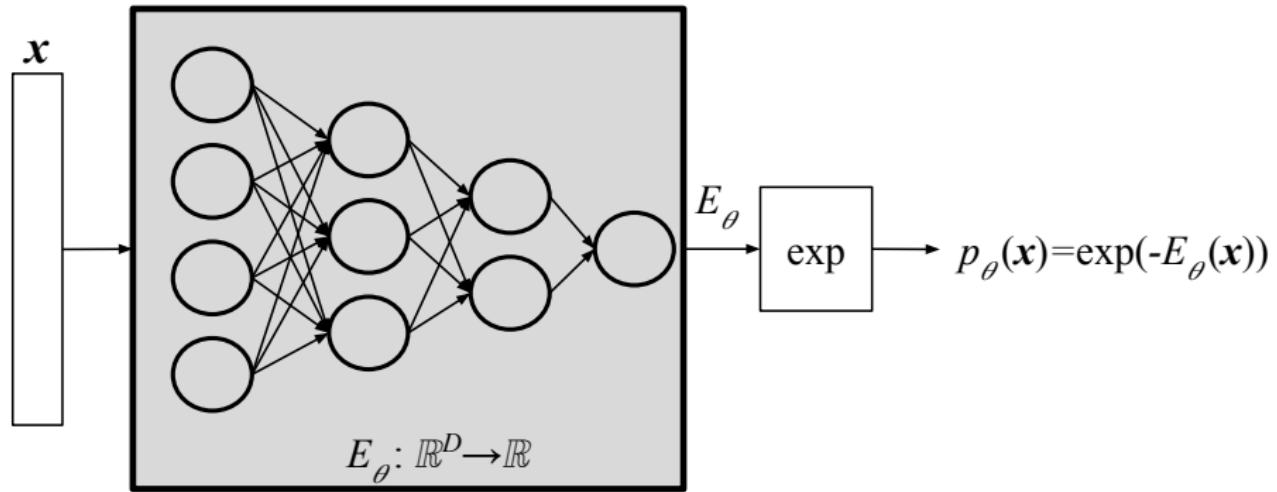


Figure:  $p_\theta(\mathbf{x})$  which satisfies the nonnegativity constraint

# Why Exponential Form

## Intuition

- If you need a fast variation in  $\exp\{-E_\theta(\mathbf{x})\}$ , then the need variation in  $E_\theta(\mathbf{x})$  is much smoother which makes the optimization easier.

# Why Exponential Form

## Intuition

- If you need a fast variation in  $\exp\{-E_\theta(\mathbf{x})\}$ , then the need variation in  $E_\theta(\mathbf{x})$  is much smoother which makes the optimization easier.
- This option enables us to use tricks from the big family exponential distributions.

# Why Exponential Form

## Intuition

- If you need a fast variation in  $\exp\{-E_\theta(\mathbf{x})\}$ , then the need variation in  $E_\theta(\mathbf{x})$  is much smoother which makes the optimization easier.
- This option enables us to use tricks from the big family exponential distributions.
- This option has some correspondence with statistical physics and the name *Energy* based models also comes from this correspondence.

# Why Exponential Form

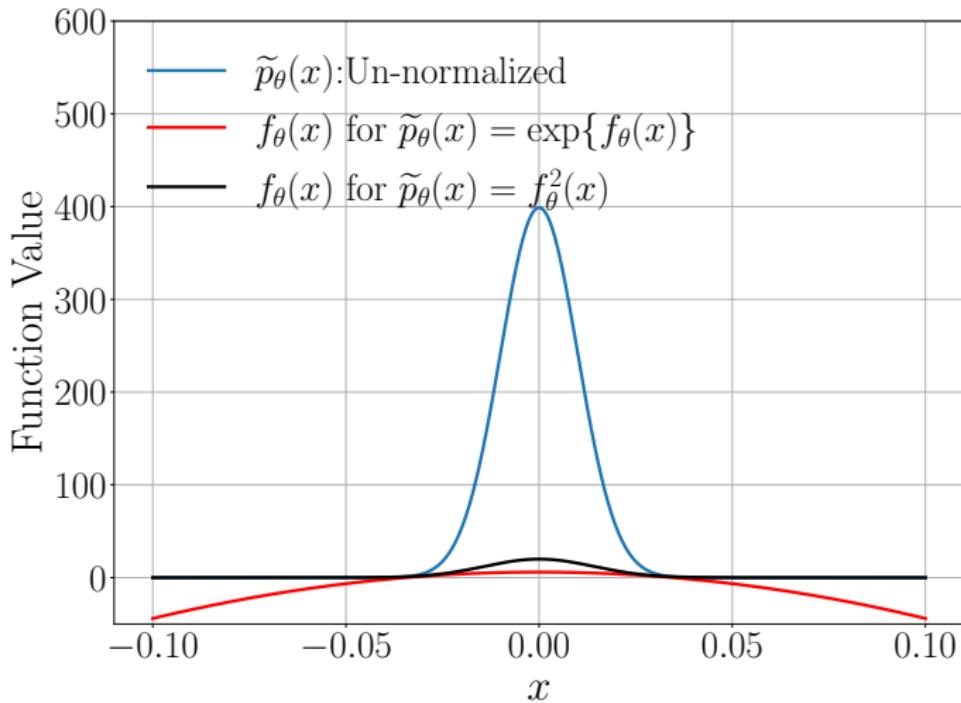


Figure: Comparing two scenarios (exponential and power) for meeting nonnegativity. The red curve is smoother and easier to optimize

# Unit Volume

## Normalizing

To have a unit volume PDF, we need to normalize  $p_\theta(\mathbf{x})$  as:

$$p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$$

# Unit Volume

## Normalizing

To have a unit volume PDF, we need to normalize  $p_\theta(\mathbf{x})$  as:

$$p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$$

where  $Z_\theta$ , known as ***Partition Function***, is:

$$Z_\theta = \int_{\mathbf{x}} \exp(-E_\theta(\mathbf{x})) d\mathbf{x}$$

# Unit Volume

## Normalizing

To have a unit volume PDF, we need to normalize  $p_\theta(\mathbf{x})$  as:

$$p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$$

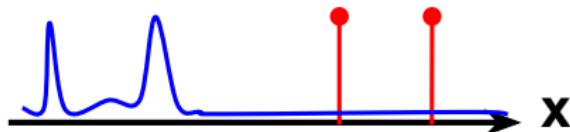
where  $Z_\theta$ , known as ***Partition Function***, is:

$$Z_\theta = \int_{\mathbf{x}} \exp(-E_\theta(\mathbf{x})) d\mathbf{x}$$

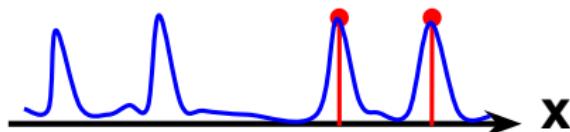
The above PDF is normalized but calculating  $Z_\theta$  is intractable even for simple Neural Networks.

# Intuition for Unit Volume

—  $p_{\theta}(\mathbf{x})$  Data Samples



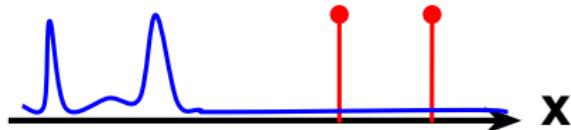
$$\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log p_{\theta}(\mathbf{x})] \simeq \frac{1}{|\mathcal{D}|} \sum_{i=1}^N \log p_{\theta}(\mathbf{x}_i)$$



(a) Without unit volume constraint

# Intuition for Unit Volume

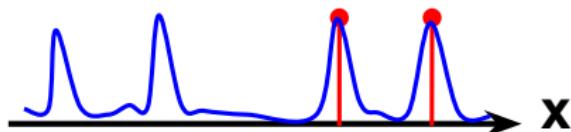
—  $p_{\theta}(\mathbf{x})$  Data Samples



—  $p_{\theta}(\mathbf{x})$  Data Samples

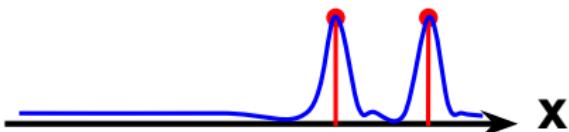


$$\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log p_{\theta}(\mathbf{x})] \simeq \frac{1}{|\mathcal{D}|} \sum_{i=1}^N \log p_{\theta}(\mathbf{x}_i)$$



(a) Without unit volume constraint

$$\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log p_{\theta}(\mathbf{x})] \simeq \frac{1}{|\mathcal{D}|} \sum_{i=1}^N \log p_{\theta}(\mathbf{x}_i)$$



(b) With unit volume constraint

**Figure:** The effect of unit-volume constraint on wiping out the spurious peaks in the generative model

## Section 2

Why EBMs

## Cons

- Sampling an EBM is hard

## Cons

- Sampling an EBM is hard
- Likelihood calculation is hard (due to partition function calculation)

## Cons

- Sampling an EBM is hard
- Likelihood calculation is hard (due to partition function calculation)
- There is no latent representation (but can be added in architectures such as Restricted Boltzmann machine)

# EBMs Cons and Pros

## Cons

- Sampling an EBM is hard
- Likelihood calculation is hard (due to partition function calculation)
- There is no latent representation (but can be added in architectures such as Restricted Boltzmann machine)

## Pros

- The only limitation over the architecture is input and output dimension  $E_\theta : \mathbb{R}^d \rightarrow \mathbb{R}$  which can be easily satisfied in:

# EBMs Cons and Pros

## Cons

- Sampling an EBM is hard
- Likelihood calculation is hard (due to partition function calculation)
- There is no latent representation (but can be added in architectures such as Restricted Boltzmann machine)

## Pros

- The only limitation over the architecture is input and output dimension  $E_\theta : \mathbb{R}^d \rightarrow \mathbb{R}$  which can be easily satisfied in:
  - Convolutional Neural Networks

# EBMs Cons and Pros

## Cons

- Sampling an EBM is hard
- Likelihood calculation is hard (due to partition function calculation)
- There is no latent representation (but can be added in architectures such as Restricted Boltzmann machine)

## Pros

- The only limitation over the architecture is input and output dimension  $E_\theta : \mathbb{R}^d \rightarrow \mathbb{R}$  which can be easily satisfied in:
  - Convolutional Neural Networks
  - Recurrent Neural Networks

# EBMs Cons and Pros

## Cons

- Sampling an EBM is hard
- Likelihood calculation is hard (due to partition function calculation)
- There is no latent representation (but can be added in architectures such as Restricted Boltzmann machine)

## Pros

- The only limitation over the architecture is input and output dimension  $E_\theta : \mathbb{R}^d \rightarrow \mathbb{R}$  which can be easily satisfied in:
  - Convolutional Neural Networks
  - Recurrent Neural Networks
  - Graph Neural Networks

# EBMs Cons and Pros

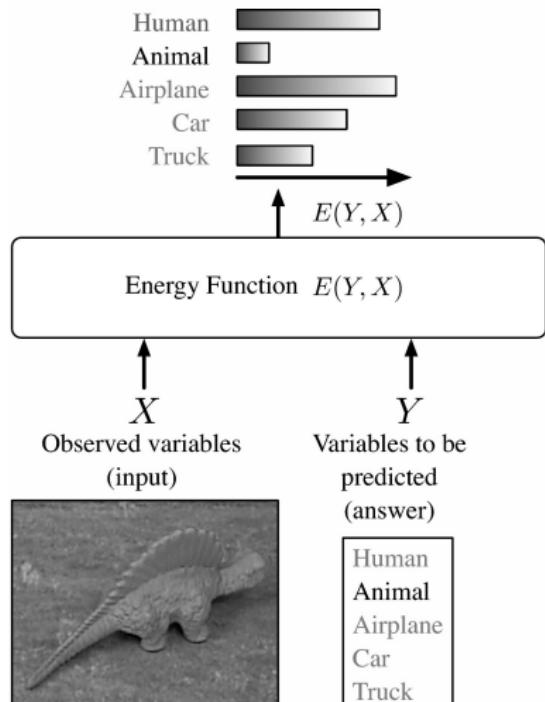
## Cons

- Sampling an EBM is hard
- Likelihood calculation is hard (due to partition function calculation)
- There is no latent representation (but can be added in architectures such as Restricted Boltzmann machine)

## Pros

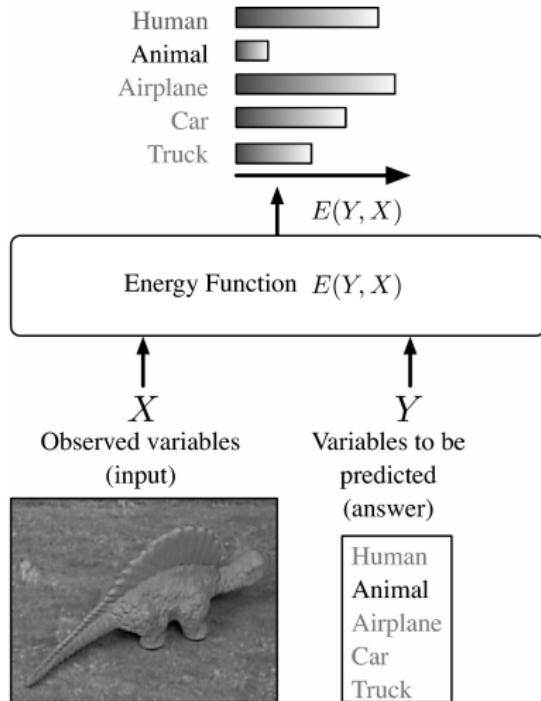
- The only limitation over the architecture is input and output dimension  $E_\theta : \mathbb{R}^d \rightarrow \mathbb{R}$  which can be easily satisfied in:
  - Convolutional Neural Networks
  - Recurrent Neural Networks
  - Graph Neural Networks
  - Attention-based Neural Networks

# EBMs Cons and Pros



**Figure:** EBM for classification  
(source: [1])

# EBMs Cons and Pros



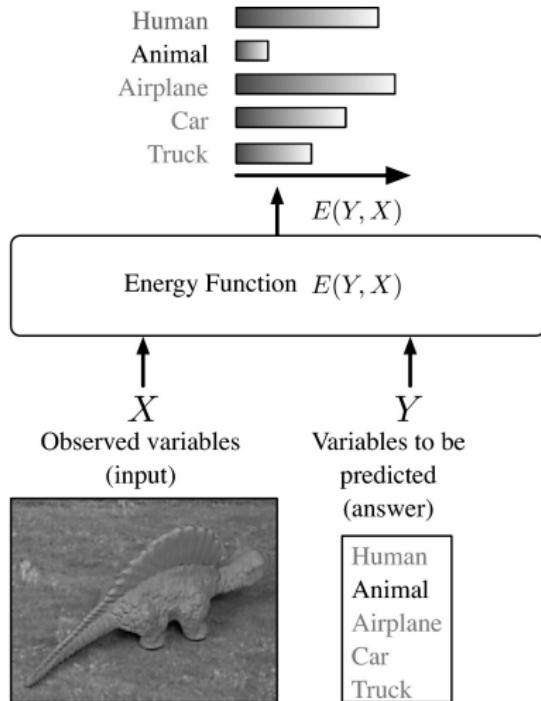
## Pros

Any problem castable as input to scalar mapping is solvable via EBMs, such as [2]:

- Image classification

**Figure:** EBM for classification  
(source: [1])

# EBMs Cons and Pros



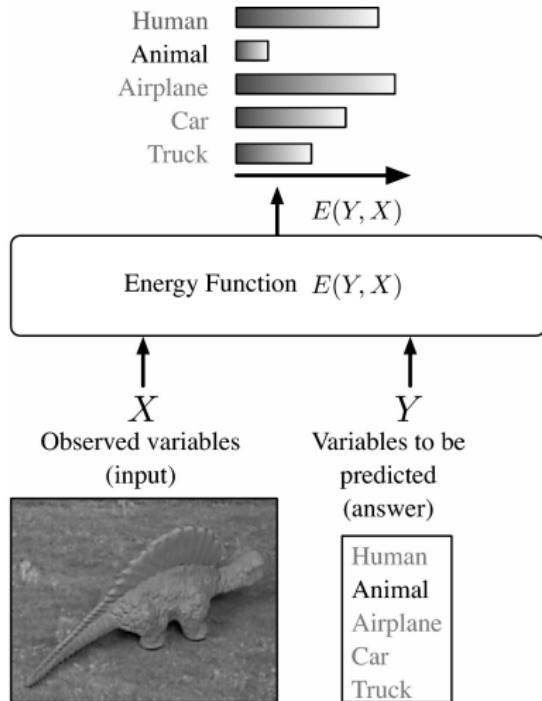
## Pros

Any problem castable as input to scalar mapping is solvable via EBMs, such as [2]:

- Image classification
- Bounding box detection

**Figure:** EBM for classification  
(source: [1])

# EBMs Cons and Pros



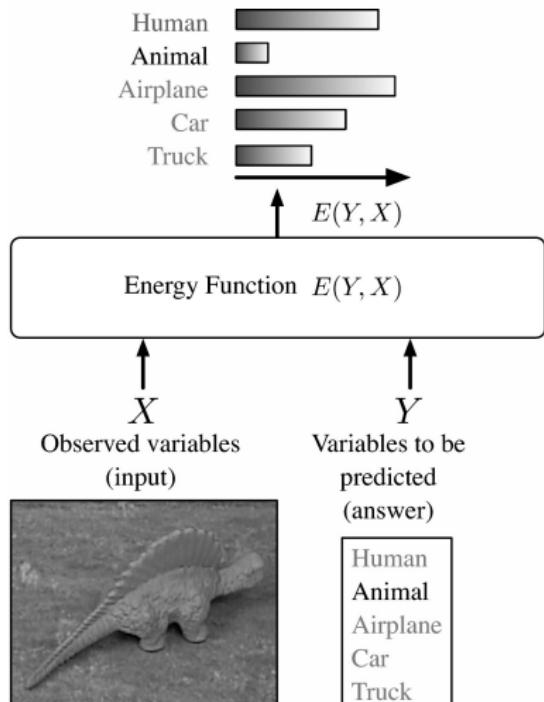
## Pros

Any problem castable as input to scalar mapping is solvable via EBMs, such as [2]:

- Image classification
- Bounding box detection
- Pixel-wise classification

**Figure:** EBM for classification  
(source: [1])

# EBMs Cons and Pros



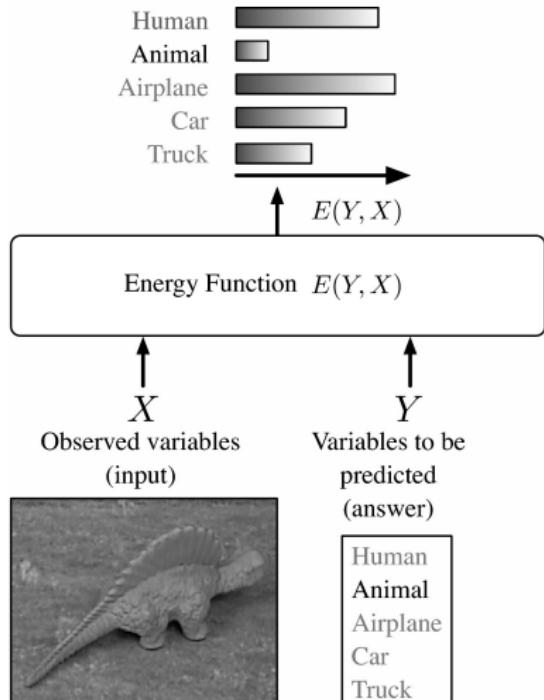
**Figure:** EBM for classification  
(source: [1])

## Pros

Any problem castable as input to scalar mapping is solvable via EBMs, such as [2]:

- Image classification
- Bounding box detection
- Pixel-wise classification
- Speech tagging

# EBMs Cons and Pros



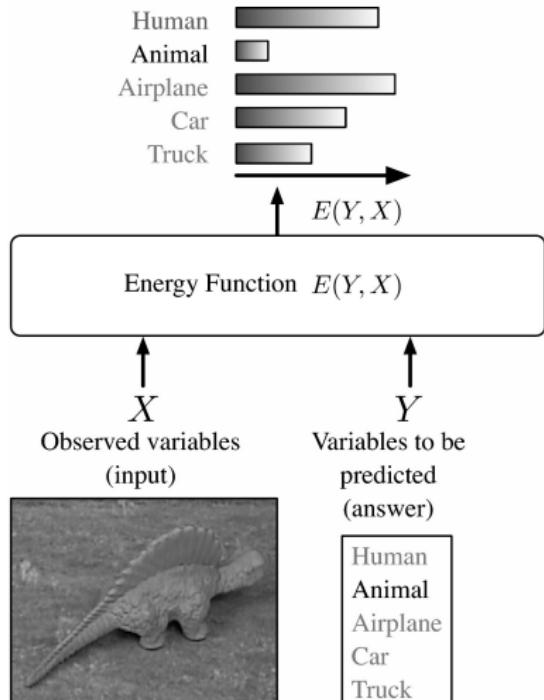
## Pros

Any problem castable as input to scalar mapping is solvable via EBMs, such as [2]:

- Image classification
- Bounding box detection
- Pixel-wise classification
- Speech tagging
- Image registration

**Figure:** EBM for classification  
(source: [1])

# EBMs Cons and Pros



## Pros

Any problem castable as input to scalar mapping is solvable via EBMs, such as [2]:

- Image classification
- Bounding box detection
- Pixel-wise classification
- Speech tagging
- Image registration
- Optical Character Recognition

**Figure:** EBM for classification  
(source: [1])

## Section 3

### Sampling EBM

# General Concept

## Sampling Challenge in EBM

Till now, different generative models follow a simple rule for sampling, such as ancestral sampling, while this procedure for EBMs cannot be used.

# General Concept

## Sampling Challenge in EBM

Till now, different generative models follow a simple rule for sampling, such as ancestral sampling, while this procedure for EBMs cannot be used.

- ☞ The Energy function can compare two samples from the likelihood perspective.

# General Concept

## Sampling Challenge in EBM

Till now, different generative models follow a simple rule for sampling, such as ancestral sampling, while this procedure for EBMs cannot be used.

- ☞ The Energy function can compare two samples from the likelihood perspective.

## Likelihood Comparison Based on Energy Function

As we see before  $p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$ . Now assume:

# General Concept

## Sampling Challenge in EBM

Till now, different generative models follow a simple rule for sampling, such as ancestral sampling, while this procedure for EBMs cannot be used.

- ☞ The Energy function can compare two samples from the likelihood perspective.

## Likelihood Comparison Based on Energy Function

As we see before  $p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$ . Now assume:

- You have a sample  $\mathbf{x}^s$ .

# General Concept

## Sampling Challenge in EBM

Till now, different generative models follow a simple rule for sampling, such as ancestral sampling, while this procedure for EBMs cannot be used.

- ☞ The Energy function can compare two samples from the likelihood perspective.

## Likelihood Comparison Based on Energy Function

As we see before  $p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$ . Now assume:

- You have a sample  $\mathbf{x}^s$ .
- You are given a proposed sample  $\mathbf{x}^p$ .

# General Concept

## Sampling Challenge in EBM

Till now, different generative models follow a simple rule for sampling, such as ancestral sampling, while this procedure for EBMs cannot be used.

- ☞ The Energy function can compare two samples from the likelihood perspective.

## Likelihood Comparison Based on Energy Function

As we see before  $p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$ . Now assume:

- You have a sample  $\mathbf{x}^s$ .
- You are given a proposed sample  $\mathbf{x}^p$ .

Then you can check  $\mathbf{x}^p$  is more probable than  $\mathbf{x}^s$  as:

# General Concept

## Sampling Challenge in EBM

Till now, different generative models follow a simple rule for sampling, such as ancestral sampling, while this procedure for EBMs cannot be used.

- ☞ The Energy function can compare two samples from the likelihood perspective.

## Likelihood Comparison Based on Energy Function

As we see before  $p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$ . Now assume:

- You have a sample  $\mathbf{x}^s$ .
- You are given a proposed sample  $\mathbf{x}^p$ .

Then you can check  $\mathbf{x}^p$  is more probable than  $\mathbf{x}^s$  as:

$$p_\theta(\mathbf{x}^p) \geq p_\theta(\mathbf{x}^s)$$

# General Concept

## Sampling Challenge in EBM

Till now, different generative models follow a simple rule for sampling, such as ancestral sampling, while this procedure for EBMs cannot be used.

- ☞ The Energy function can compare two samples from the likelihood perspective.

## Likelihood Comparison Based on Energy Function

As we see before  $p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$ . Now assume:

- You have a sample  $\mathbf{x}^s$ .
- You are given a proposed sample  $\mathbf{x}^p$ .

Then you can check  $\mathbf{x}^p$  is more probable than  $\mathbf{x}^s$  as:

$$p_\theta(\mathbf{x}^p) \geq p_\theta(\mathbf{x}^s) \Leftrightarrow \frac{\exp(-E_\theta(\mathbf{x}^p))}{Z_\theta} \geq \frac{\exp(-E_\theta(\mathbf{x}^s))}{Z_\theta}$$

# General Concept

## Sampling Challenge in EBM

Till now, different generative models follow a simple rule for sampling, such as ancestral sampling, while this procedure for EBMs cannot be used.

- ☞ The Energy function can compare two samples from the likelihood perspective.

## Likelihood Comparison Based on Energy Function

As we see before  $p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$ . Now assume:

- You have a sample  $\mathbf{x}^s$ .
- You are given a proposed sample  $\mathbf{x}^p$ .

Then you can check  $\mathbf{x}^p$  is more probable than  $\mathbf{x}^s$  as:

$$\begin{aligned} p_\theta(\mathbf{x}^p) \geq p_\theta(\mathbf{x}^s) &\Leftrightarrow \frac{\exp(-E_\theta(\mathbf{x}^p))}{Z_\theta} \geq \frac{\exp(-E_\theta(\mathbf{x}^s))}{Z_\theta} \\ &\Leftrightarrow E_\theta(\mathbf{x}^p) \leq E_\theta(\mathbf{x}^s) \end{aligned}$$

# Metropolis-Hastings Algorithm

## Basic Idea

Metropolis-Hastings (HM) Algorithm is a kind of Markov-chain Monte Carlo algorithm that produces samples from an unnormalized distribution  $p_{\theta}(\mathbb{X})$  (HM just uses energy function).

# Metropolis-Hastings Algorithm

## Basic Idea

Metropolis-Hastings (HM) Algorithm is a kind of Markov-chain Monte Carlo algorithm that produces samples from an unnormalized distribution  $p_\theta(\mathbb{X})$  (HM just uses energy function). The basic idea is:

- Initialize vector  $\mathbf{x}^s$  randomly

# Metropolis-Hastings Algorithm

## Basic Idea

Metropolis-Hastings (HM) Algorithm is a kind of Markov-chain Monte Carlo algorithm that produces samples from an unnormalized distribution  $p_\theta(\mathbb{X})$  (HM just uses energy function). The basic idea is:

- Initialize vector  $\mathbf{x}^s$  randomly
- Propose a new candidate sample  $\mathbf{x}^p$

# Metropolis-Hastings Algorithm

## Basic Idea

Metropolis-Hastings (HM) Algorithm is a kind of Markov-chain Monte Carlo algorithm that produces samples from an unnormalized distribution  $p_\theta(\mathbb{X})$  (HM just uses energy function). The basic idea is:

- Initialize vector  $\mathbf{x}^s$  randomly
- Propose a new candidate sample  $\mathbf{x}^p$
- Accept/Reject  $\mathbf{x}^p$  such that after a sufficient number of iterations, the generated samples follow distribution  $p_\theta(\mathbb{X})$

# Metropolis-Hastings Algorithm Concept

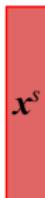


Figure: The current sample

# Metropolis-Hastings Algorithm Concept

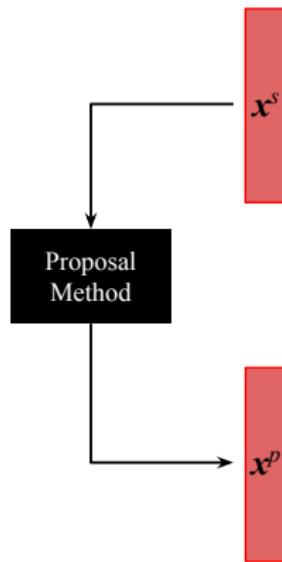


Figure: Propose a new sample using a *Proposal Method*

# Metropolis-Hastings Algorithm Concept

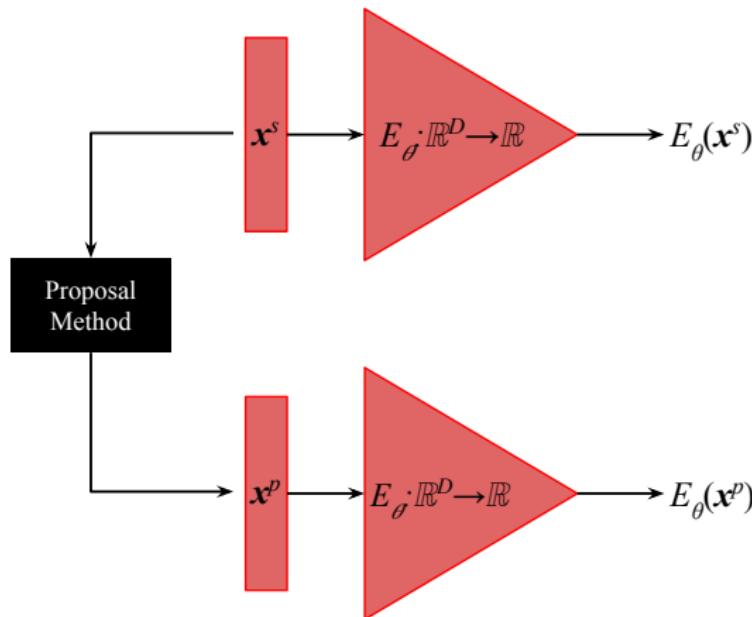


Figure: Evaluate the enrgy function for current sample  $\mathbf{x}^s$  and proposed sample  $\mathbf{x}^p$

# Metropolis-Hastings Algorithm Concept

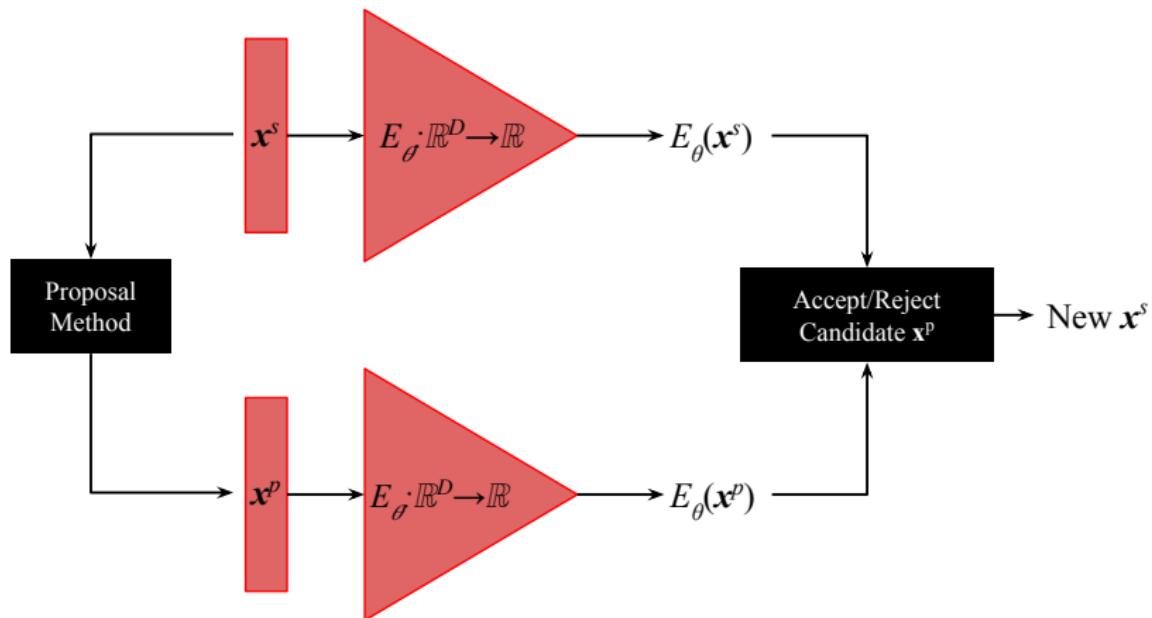


Figure: Accept or Reject proposed sample  $x^p$  and generate new  $x^s$

# Metropolis-Hastings Algorithm

---

**Algorithm 1** Sampling from  $p_\theta(\mathbb{X})$  using Metropolis-Hastings Algorithm

---

1: **procedure** METROPOLIS-HASTINGS(Proposal Distribution  $q$ )

# Metropolis-Hastings Algorithm

---

**Algorithm 2** Sampling from  $p_\theta(\mathbb{X})$  using Metropolis-Hastings Algorithm

---

1: **procedure** METROPOLIS-HASTINGS(Proposal Distribution  $q$ )  
2:     **Initialization:**  $x^1$

# Metropolis-Hastings Algorithm

---

**Algorithm 3** Sampling from  $p_\theta(\mathbb{X})$  using Metropolis-Hastings Algorithm

---

```
1: procedure METROPOLIS-HASTINGS(Proposal Distribution  $q$ )
2:   Initialization:  $x^1$ 
3:   for  $s = 1 : T$  do
```

# Metropolis-Hastings Algorithm

---

**Algorithm 4** Sampling from  $p_\theta(\mathbb{X})$  using Metropolis-Hastings Algorithm

---

```
1: procedure METROPOLIS-HASTINGS(Proposal Distribution  $q$ )
2:   Initialization:  $\mathbf{x}^1$ 
3:   for  $s = 1 : T$  do
4:     Sample  $\mathbf{x}^p \sim q(\mathbb{X}|\mathbf{x}^s)$  #Example:  $q(\mathbb{X}|\mathbf{x}^s) = \mathcal{N}(\mathbb{X}|\mathbf{x}^s, \tau^2 \mathbf{I})x$ 
```

# Metropolis-Hastings Algorithm

---

**Algorithm 5** Sampling from  $p_\theta(\mathbb{X})$  using Metropolis-Hastings Algorithm

---

```
1: procedure METROPOLIS-HASTINGS(Proposal Distribution  $q$ )
2:   Initialization:  $\mathbf{x}^1$ 
3:   for  $s = 1 : T$  do
4:     Sample  $\mathbf{x}^p \sim q(\mathbb{X}|\mathbf{x}^s)$  #Example:  $q(\mathbb{X}|\mathbf{x}^s) = \mathcal{N}(\mathbb{X}|\mathbf{x}^s, \tau^2 \mathbf{I})$ 
5:     Compute:
```

# Metropolis-Hastings Algorithm

---

**Algorithm 6** Sampling from  $p_\theta(\mathbb{X})$  using Metropolis-Hastings Algorithm

---

```
1: procedure METROPOLIS-HASTINGS(Proposal Distribution  $q$ )
2:   Initialization:  $\mathbf{x}^1$ 
3:   for  $s = 1 : T$  do
4:     Sample  $\mathbf{x}^p \sim q(\mathbb{X}|\mathbf{x}^s)$  #Example:  $q(\mathbb{X}|\mathbf{x}^s) = \mathcal{N}(\mathbb{X}|\mathbf{x}^s, \tau^2 \mathbf{I})$ 
5:     Compute:
```

$$\alpha = \frac{p_\theta(\mathbf{x}^p)q(\mathbf{x}^s|\mathbf{x}^p)}{p_\theta(\mathbf{x}^s)q(\mathbf{x}^p|\mathbf{x}^s)}$$

# Metropolis-Hastings Algorithm

---

**Algorithm 7** Sampling from  $p_\theta(\mathbb{X})$  using Metropolis-Hastings Algorithm

---

```
1: procedure METROPOLIS-HASTINGS(Proposal Distribution  $q$ )
2:   Initialization:  $\mathbf{x}^1$ 
3:   for  $s = 1 : T$  do
4:     Sample  $\mathbf{x}^p \sim q(\mathbb{X}|\mathbf{x}^s)$  #Example:  $q(\mathbb{X}|\mathbf{x}^s) = \mathcal{N}(\mathbb{X}|\mathbf{x}^s, \tau^2 \mathbf{I})$ 
5:     Compute:
```

$$\alpha = \frac{p_\theta(\mathbf{x}^p)q(\mathbf{x}^s|\mathbf{x}^p)}{p_\theta(\mathbf{x}^s)q(\mathbf{x}^p|\mathbf{x}^s)} = \frac{\exp(-E_\theta(\mathbf{x}^p))q(\mathbf{x}^s|\mathbf{x}^p)}{\exp(-E_\theta(\mathbf{x}^s))q(\mathbf{x}^p|\mathbf{x}^s)}$$

# Metropolis-Hastings Algorithm

---

**Algorithm 8** Sampling from  $p_\theta(\mathbb{X})$  using Metropolis-Hastings Algorithm

---

```
1: procedure METROPOLIS-HASTINGS(Proposal Distribution  $q$ )
2:   Initialization:  $\mathbf{x}^1$ 
3:   for  $s = 1 : T$  do
4:     Sample  $\mathbf{x}^p \sim q(\mathbb{X}|\mathbf{x}^s)$  #Example:  $q(\mathbb{X}|\mathbf{x}^s) = \mathcal{N}(\mathbb{X}|\mathbf{x}^s, \tau^2 \mathbf{I})$ 
5:     Compute:
         
$$\alpha = \frac{p_\theta(\mathbf{x}^p)q(\mathbf{x}^s|\mathbf{x}^p)}{p_\theta(\mathbf{x}^s)q(\mathbf{x}^p|\mathbf{x}^s)} = \frac{\exp(-E_\theta(\mathbf{x}^p))q(\mathbf{x}^s|\mathbf{x}^p)}{\exp(-E_\theta(\mathbf{x}^s))q(\mathbf{x}^p|\mathbf{x}^s)}$$

6:     Compute:  $A = \min(1, \alpha)$ 
```

# Metropolis-Hastings Algorithm

---

**Algorithm 9** Sampling from  $p_\theta(\mathbb{X})$  using Metropolis-Hastings Algorithm

---

```
1: procedure METROPOLIS-HASTINGS(Proposal Distribution  $q$ )
2:   Initialization:  $\mathbf{x}^1$ 
3:   for  $s = 1 : T$  do
4:     Sample  $\mathbf{x}^p \sim q(\mathbb{X}|\mathbf{x}^s)$  #Example:  $q(\mathbb{X}|\mathbf{x}^s) = \mathcal{N}(\mathbb{X}|\mathbf{x}^s, \tau^2 \mathbf{I})$ 
5:     Compute:
         
$$\alpha = \frac{p_\theta(\mathbf{x}^p)q(\mathbf{x}^s|\mathbf{x}^p)}{p_\theta(\mathbf{x}^s)q(\mathbf{x}^p|\mathbf{x}^s)} = \frac{\exp(-E_\theta(\mathbf{x}^p))q(\mathbf{x}^s|\mathbf{x}^p)}{\exp(-E_\theta(\mathbf{x}^s))q(\mathbf{x}^p|\mathbf{x}^s)}$$

6:     Compute:  $A = \min(1, \alpha)$ 
7:     Sample  $u \sim U(0, 1)$ 
```

# Metropolis-Hastings Algorithm

---

**Algorithm 10** Sampling from  $p_\theta(\mathbb{X})$  using Metropolis-Hastings Algorithm

---

```
1: procedure METROPOLIS-HASTINGS(Proposal Distribution  $q$ )
2:   Initialization:  $\mathbf{x}^1$ 
3:   for  $s = 1 : T$  do
4:     Sample  $\mathbf{x}^p \sim q(\mathbb{X}|\mathbf{x}^s)$  #Example:  $q(\mathbb{X}|\mathbf{x}^s) = \mathcal{N}(\mathbb{X}|\mathbf{x}^s, \tau^2 \mathbf{I})$ 
5:     Compute:
         
$$\alpha = \frac{p_\theta(\mathbf{x}^p)q(\mathbf{x}^s|\mathbf{x}^p)}{p_\theta(\mathbf{x}^s)q(\mathbf{x}^p|\mathbf{x}^s)} = \frac{\exp(-E_\theta(\mathbf{x}^p))q(\mathbf{x}^s|\mathbf{x}^p)}{\exp(-E_\theta(\mathbf{x}^s))q(\mathbf{x}^p|\mathbf{x}^s)}$$

6:     Compute:  $A = \min(1, \alpha)$ 
7:     Sample  $u \sim U(0, 1)$ 
8:     Generate new sample:
```

$$\mathbf{x}^{s+1} = \begin{cases} \mathbf{x}^p & \text{if } u \leq A \text{(Accept)} \\ \mathbf{x}^s & \text{if } u > A \text{(Reject)} \end{cases}$$

# Metropolis-Hastings Algorithm

---

**Algorithm 11** Sampling from  $p_\theta(\mathbb{X})$  using Metropolis-Hastings Algorithm

---

```
1: procedure METROPOLIS-HASTINGS(Proposal Distribution  $q$ )
2:   Initialization:  $\mathbf{x}^1$ 
3:   for  $s = 1 : T$  do
4:     Sample  $\mathbf{x}^p \sim q(\mathbb{X}|\mathbf{x}^s)$  #Example:  $q(\mathbb{X}|\mathbf{x}^s) = \mathcal{N}(\mathbb{X}|\mathbf{x}^s, \tau^2 \mathbf{I})$ 
5:     Compute:
         
$$\alpha = \frac{p_\theta(\mathbf{x}^p)q(\mathbf{x}^s|\mathbf{x}^p)}{p_\theta(\mathbf{x}^s)q(\mathbf{x}^p|\mathbf{x}^s)} = \frac{\exp(-E_\theta(\mathbf{x}^p))q(\mathbf{x}^s|\mathbf{x}^p)}{\exp(-E_\theta(\mathbf{x}^s))q(\mathbf{x}^p|\mathbf{x}^s)}$$

6:     Compute:  $A = \min(1, \alpha)$ 
7:     Sample  $u \sim U(0, 1)$ 
8:     Generate new sample:
         
$$\mathbf{x}^{s+1} = \begin{cases} \mathbf{x}^p & \text{if } u \leq A \text{(Accept)} \\ \mathbf{x}^s & \text{if } u > A \text{(Reject)} \end{cases}$$

9:   end for
```

---

# Metropolis-Hastings Algorithm

---

**Algorithm 12** Sampling from  $p_\theta(\mathbb{X})$  using Metropolis-Hastings Algorithm

---

```
1: procedure METROPOLIS-HASTINGS(Proposal Distribution  $q$ )
2:   Initialization:  $\mathbf{x}^1$ 
3:   for  $s = 1 : T$  do
4:     Sample  $\mathbf{x}^p \sim q(\mathbb{X}|\mathbf{x}^s)$  #Example:  $q(\mathbb{X}|\mathbf{x}^s) = \mathcal{N}(\mathbb{X}|\mathbf{x}^s, \tau^2 \mathbf{I})$ 
5:     Compute:
         
$$\alpha = \frac{p_\theta(\mathbf{x}^p)q(\mathbf{x}^s|\mathbf{x}^p)}{p_\theta(\mathbf{x}^s)q(\mathbf{x}^p|\mathbf{x}^s)} = \frac{\exp(-E_\theta(\mathbf{x}^p))q(\mathbf{x}^s|\mathbf{x}^p)}{\exp(-E_\theta(\mathbf{x}^s))q(\mathbf{x}^p|\mathbf{x}^s)}$$

6:     Compute:  $A = \min(1, \alpha)$ 
7:     Sample  $u \sim U(0, 1)$ 
8:     Generate new sample:
         
$$\mathbf{x}^{s+1} = \begin{cases} \mathbf{x}^p & \text{if } u \leq A \text{(Accept)} \\ \mathbf{x}^s & \text{if } u > A \text{(Reject)} \end{cases}$$

9:   end for
10: end procedure
```

---

# Metropolis-Hastings Algorithm

## Proposal Distribution

As we see in the MH algorithm the proposal method is done by sampling a ***Proposal Distribution***  $q$ .

# Metropolis-Hastings Algorithm

## Proposal Distribution

As we see in the MH algorithm the proposal method is done by sampling a ***Proposal Distribution***  $q$ .

## Notes on Proposal Distribution

The value of  $\alpha$  in the MH algorithm can be rewritten as the multiplication of two terms:

$$\alpha = \frac{\exp(-E_\theta(\mathbf{x}^p))}{\exp(-E_\theta(\mathbf{x}^s))} \times \frac{q(\mathbf{x}^s | \mathbf{x}^p)}{q(\mathbf{x}^p | \mathbf{x}^s)}$$

# Metropolis-Hastings Algorithm

## Proposal Distribution

As we see in the MH algorithm the proposal method is done by sampling a ***Proposal Distribution***  $q$ .

## Notes on Proposal Distribution

The value of  $\alpha$  in the MH algorithm can be rewritten as the multiplication of two terms:

$$\alpha = \frac{\exp(-E_\theta(\mathbf{x}^p))}{\exp(-E_\theta(\mathbf{x}^s))} \times \frac{q(\mathbf{x}^s | \mathbf{x}^p)}{q(\mathbf{x}^p | \mathbf{x}^s)}$$

- The first term in RHS:
  - Compute ratio between the likelihood of  $\mathbf{x}^p$  and  $\mathbf{x}^s$

# Metropolis-Hastings Algorithm

## Proposal Distribution

As we see in the MH algorithm the proposal method is done by sampling a ***Proposal Distribution***  $q$ .

## Notes on Proposal Distribution

The value of  $\alpha$  in the MH algorithm can be rewritten as the multiplication of two terms:

$$\alpha = \frac{\exp(-E_\theta(\mathbf{x}^p))}{\exp(-E_\theta(\mathbf{x}^s))} \times \frac{q(\mathbf{x}^s | \mathbf{x}^p)}{q(\mathbf{x}^p | \mathbf{x}^s)}$$

- The first term in RHS:
  - Compute ratio between the likelihood of  $\mathbf{x}^p$  and  $\mathbf{x}^s$
  - Increase the acceptance of  $\mathbf{x}^p$  as the likelihood ratio increase

# Metropolis-Hastings Algorithm

## Proposal Distribution

As we see in the MH algorithm the proposal method is done by sampling a ***Proposal Distribution***  $q$ .

## Notes on Proposal Distribution

The value of  $\alpha$  in the MH algorithm can be rewritten as the multiplication of two terms:

$$\alpha = \frac{\exp(-E_\theta(\mathbf{x}^p))}{\exp(-E_\theta(\mathbf{x}^s))} \times \frac{q(\mathbf{x}^s | \mathbf{x}^p)}{q(\mathbf{x}^p | \mathbf{x}^s)}$$

- The first term in RHS:
  - Compute ratio between the likelihood of  $\mathbf{x}^p$  and  $\mathbf{x}^s$
  - Increase the acceptance of  $\mathbf{x}^p$  as the likelihood ratio increase
- The second term in RHS known as *Hastings correction*:

# Metropolis-Hastings Algorithm

## Proposal Distribution

As we see in the MH algorithm the proposal method is done by sampling a ***Proposal Distribution***  $q$ .

## Notes on Proposal Distribution

The value of  $\alpha$  in the MH algorithm can be rewritten as the multiplication of two terms:

$$\alpha = \frac{\exp(-E_\theta(\mathbf{x}^p))}{\exp(-E_\theta(\mathbf{x}^s))} \times \frac{q(\mathbf{x}^s | \mathbf{x}^p)}{q(\mathbf{x}^p | \mathbf{x}^s)}$$

- The first term in RHS:
  - Compute ratio between the likelihood of  $\mathbf{x}^p$  and  $\mathbf{x}^s$
  - Increase the acceptance of  $\mathbf{x}^p$  as the likelihood ratio increase
- The second term in RHS known as *Hastings correction*:
  - Correction for the cases that the proposal distribution itself (rather than just the target distribution) might favor either  $\mathbf{x}^p$  or  $\mathbf{x}^s$ .

# Metropolis-Hastings Algorithm

## Notes on Proposal Distribution (cont.)

- A proposal distribution  $q$  is valid if it “covers” the support of the target:

# Metropolis-Hastings Algorithm

## Notes on Proposal Distribution (cont.)

- A proposal distribution  $q$  is valid if it “covers” the support of the target:

$$\text{supp}(p_\theta) \subseteq \cup_{\mathbf{x}} \text{supp}(q(\cdot | \mathbf{x}))$$

# Metropolis-Hastings Algorithm

## Notes on Proposal Distribution (cont.)

- A proposal distribution  $q$  is valid if it “covers” the support of the target:

$$\text{supp}(p_\theta) \subseteq \cup_{\mathbf{x}} \text{supp}(q(\cdot | \mathbf{x}))$$

- You can use  $q(\mathbb{X} | \mathbf{x}^s) = q(\mathbb{X})$ . The resulting method is known as the *independence sampler*.

# Metropolis-Hastings Algorithm

## Notes on Proposal Distribution (cont.)

- A proposal distribution  $q$  is valid if it “covers” the support of the target:

$$\text{supp}(p_\theta) \subseteq \cup_{\mathbf{x}} \text{supp}(q(\cdot | \mathbf{x}))$$

- You can use  $q(\mathbb{X} | \mathbf{x}^s) = q(\mathbb{X})$ . The resulting method is known as the *independence sampler*.
- If you select  $q(\mathbb{X} | \mathbf{x}^s) = \mathcal{N}(\mathbb{X} | \mathbf{x}^s, \tau^2 \mathbf{I})$ , then the resulting algorithm is known as *random walk Metropolis algorithm* (RWM).

# Metropolis-Hastings Algorithm

## Notes on Proposal Distribution (cont.)

- A proposal distribution  $q$  is valid if it “covers” the support of the target:

$$\text{supp}(p_\theta) \subseteq \cup_{\mathbf{x}} \text{supp}(q(\cdot | \mathbf{x}))$$

- You can use  $q(\mathbb{X} | \mathbf{x}^s) = q(\mathbb{X})$ . The resulting method is known as the *independence sampler*.
- If you select  $q(\mathbb{X} | \mathbf{x}^s) = \mathcal{N}(\mathbb{X} | \mathbf{x}^s, \tau^2 \mathbf{I})$ , then the resulting algorithm is known as *random walk Metropolis algorithm* (RWM).
- In several cases such as RWM,  $q(\mathbf{x}^s | \mathbf{x}^p) = q(\mathbf{x}^p | \mathbf{x}^s)$ ,

# Metropolis-Hastings Algorithm

## Notes on Proposal Distribution (cont.)

- A proposal distribution  $q$  is valid if it “covers” the support of the target:

$$\text{supp}(p_\theta) \subseteq \cup_{\mathbf{x}} \text{supp}(q(\cdot | \mathbf{x}))$$

- You can use  $q(\mathbb{X} | \mathbf{x}^s) = q(\mathbb{X})$ . The resulting method is known as the *independence sampler*.
- If you select  $q(\mathbb{X} | \mathbf{x}^s) = \mathcal{N}(\mathbb{X} | \mathbf{x}^s, \tau^2 \mathbf{I})$ , then the resulting algorithm is known as *random walk Metropolis algorithm* (RWM).
- In several cases such as RWM,  $q(\mathbf{x}^s | \mathbf{x}^p) = q(\mathbf{x}^p | \mathbf{x}^s)$ , and thus:

$$\alpha = \frac{\exp(-E_\theta(\mathbf{x}^p))}{\exp(-E_\theta(\mathbf{x}^s))}$$

# Metropolis-Hastings Algorithm

## Notes on Proposal Distribution (cont.)

- A proposal distribution  $q$  is valid if it “covers” the support of the target:

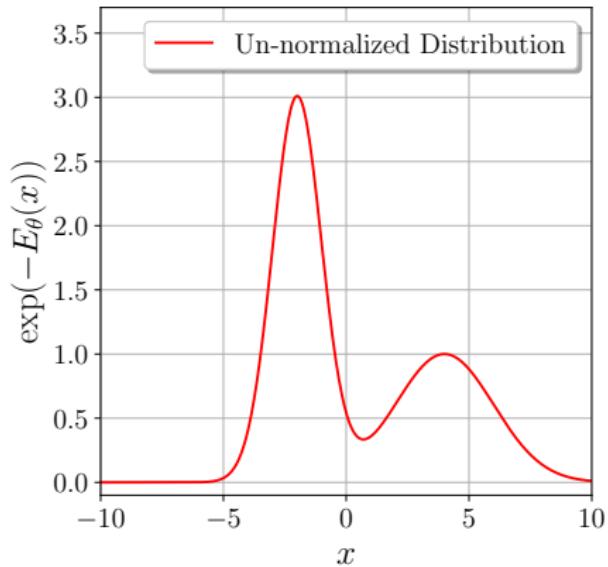
$$\text{supp}(p_\theta) \subseteq \cup_{\mathbf{x}} \text{supp}(q(\cdot | \mathbf{x}))$$

- You can use  $q(\mathbb{X} | \mathbf{x}^s) = q(\mathbb{X})$ . The resulting method is known as the *independence sampler*.
- If you select  $q(\mathbb{X} | \mathbf{x}^s) = \mathcal{N}(\mathbb{X} | \mathbf{x}^s, \tau^2 \mathbf{I})$ , then the resulting algorithm is known as *random walk Metropolis algorithm* (RWM).
- In several cases such as RWM,  $q(\mathbf{x}^s | \mathbf{x}^p) = q(\mathbf{x}^p | \mathbf{x}^s)$ , and thus:

$$\alpha = \frac{\exp(-E_\theta(\mathbf{x}^p))}{\exp(-E_\theta(\mathbf{x}^s))}$$

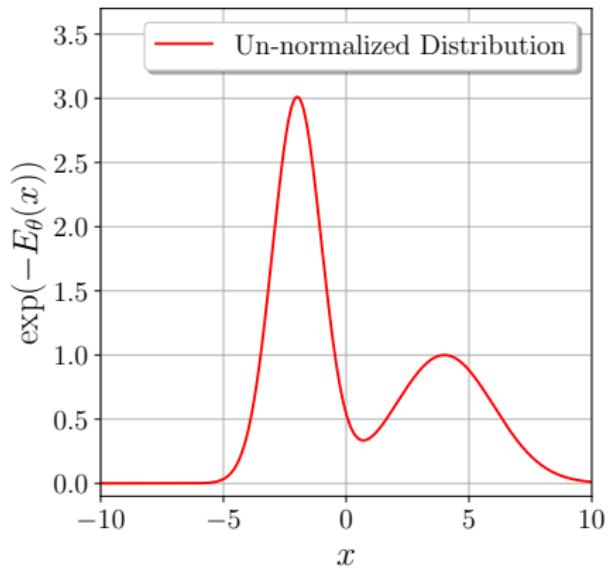
- When using the MH algorithm, the initial samples, known as *Burned in Samples* are usually omitted as those samples are generated before the chain reaches its stationary distribution.

# Metropolis-Hastings Algorithm

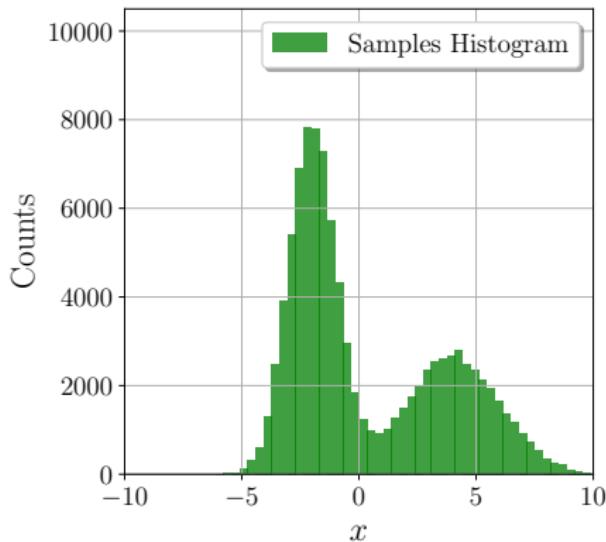


(a) Un-normalized distribution

# Metropolis-Hastings Algorithm



(a) Un-normalized distribution



(b) MH algorithm samples

**Figure:** The result of MH algorithm. (Un-normalized distribution  $3 \exp\left(-\frac{(x+2)^2}{2}\right) + \exp\left(-\frac{(x-4)^2}{8}\right)$ , number of samples:  $10^5$ , burned-in samples:  $10^3$ , and  $q(X|x^s) = \mathcal{N}(X|x^s, 1)$ ) (source: [3])

# Metropolis-Hastings Algorithm

## Cons and Pros

- Pros:
  - HM can sample from un-normalized distributions.

# Metropolis-Hastings Algorithm

## Cons and Pros

- Pros:
  - HM can sample from un-normalized distributions.
  - HM algorithm guarantees convergence to the target distribution irrespective of the initial point under mild conditions.

# Metropolis-Hastings Algorithm

## Cons and Pros

- Pros:

- HM can sample from un-normalized distributions.
- HM algorithm guarantees convergence to the target distribution irrespective of the initial point under mild conditions.
- Based on the problem, one can design the proposal distribution that can improve convergence

# Metropolis-Hastings Algorithm

## Cons and Pros

- Pros:
  - HM can sample from un-normalized distributions.
  - HM algorithm guarantees convergence to the target distribution irrespective of the initial point under mild conditions.
  - Based on the problem, one can design the proposal distribution that can improve convergence
- Cons:
  - Selecting a bad proposal distribution can decrease the convergence rate impressively

# Metropolis-Hastings Algorithm

## Cons and Pros

- Pros:

- HM can sample from un-normalized distributions.
- HM algorithm guarantees convergence to the target distribution irrespective of the initial point under mild conditions.
- Based on the problem, one can design the proposal distribution that can improve convergence

- Cons:

- Selecting a bad proposal distribution can decrease the convergence rate impressively
- The generated samples by MH are correlated.

# Metropolis-Hastings Algorithm

## Cons and Pros

- Pros:

- HM can sample from un-normalized distributions.
- HM algorithm guarantees convergence to the target distribution irrespective of the initial point under mild conditions.
- Based on the problem, one can design the proposal distribution that can improve convergence

- Cons:

- Selecting a bad proposal distribution can decrease the convergence rate impressively
- The generated samples by MH are correlated.
- It needs a burn-in period to converge the target distribution.

# Metropolis-Hastings Algorithm

## Cons and Pros

- Pros:

- HM can sample from un-normalized distributions.
- HM algorithm guarantees convergence to the target distribution irrespective of the initial point under mild conditions.
- Based on the problem, one can design the proposal distribution that can improve convergence

- Cons:

- Selecting a bad proposal distribution can decrease the convergence rate impressively
- The generated samples by MH are correlated.
- It needs a burn-in period to converge the target distribution.
- It is challenging to assess the convergence of the chain to target distribution.

## Resolving Proposal Distribution Problem

The energy function  $E_\theta$  can provide two types of information:

- Comparing the likelihood of two Samples

## Resolving Proposal Distribution Problem

The energy function  $E_\theta$  can provide two types of information:

- Comparing the likelihood of two Samples
- Calculating the direction  $-\nabla_{\mathbf{x}} E_\theta(\mathbf{x})$  which results in decreasing the energy and equivalently increasing the likelihood

## Resolving Proposal Distribution Problem

The energy function  $E_\theta$  can provide two types of information:

- Comparing the likelihood of two Samples
- Calculating the direction  $-\nabla_{\mathbf{x}} E_\theta(\mathbf{x})$  which results in decreasing the energy and equivalently increasing the likelihood

Thus to resolve the challenge of selecting proposal distribution in MH, we can use  $-\nabla_{\mathbf{x}} E_\theta(\mathbf{x})$  to propose a new point  $\mathbf{x}^p$

## Resolving Proposal Distribution Problem

The energy function  $E_\theta$  can provide two types of information:

- Comparing the likelihood of two Samples
- Calculating the direction  $-\nabla_{\mathbf{x}} E_\theta(\mathbf{x})$  which results in decreasing the energy and equivalently increasing the likelihood

Thus to resolve the challenge of selecting proposal distribution in MH, we can use  $-\nabla_{\mathbf{x}} E_\theta(\mathbf{x})$  to propose a new point  $\mathbf{x}^p$

## Basic Idea

The *Hamiltonian Monte Carlo* (HMC) algorithm uses the information from  $-\nabla_{\mathbf{x}} E_\theta(\mathbf{x})$  to generate the proposed sample  $\mathbf{x}^p$ .

# Hamiltonian Monte Carlo

## From Physics to Statistics

The Hamiltonian Monte Carlo (HMC) is a physics-inspired sampling method that uses the Hamiltonian function defined as follows:

$$\mathcal{H}(\boldsymbol{x}, \boldsymbol{v}) = E_\theta(\boldsymbol{x}) + \mathcal{K}(\boldsymbol{v})$$

# Hamiltonian Monte Carlo

## From Physics to Statistics

The Hamiltonian Monte Carlo (HMC) is a physics-inspired sampling method that uses the Hamiltonian function defined as follows:

$$\mathcal{H}(\boldsymbol{x}, \boldsymbol{v}) = E_\theta(\boldsymbol{x}) + \mathcal{K}(\boldsymbol{v})$$

- In physics:
  - $E_\theta(\boldsymbol{x})$ : Represents the potential energy at position  $\boldsymbol{x}$ .
  - $\mathcal{K}(\boldsymbol{v})$ : Represents the kinetic energy at speed  $\boldsymbol{v}$ .

# Hamiltonian Monte Carlo

## From Physics to Statistics

The Hamiltonian Monte Carlo (HMC) is a physics-inspired sampling method that uses the Hamiltonian function defined as follows:

$$\mathcal{H}(\boldsymbol{x}, \boldsymbol{v}) = E_\theta(\boldsymbol{x}) + \mathcal{K}(\boldsymbol{v})$$

- In physics:
  - $E_\theta(\boldsymbol{x})$ : Represents the potential energy at position  $\boldsymbol{x}$ .
  - $\mathcal{K}(\boldsymbol{v})$ : Represents the kinetic energy at speed  $\boldsymbol{v}$ .
- In Statistical Setting:
  - $E_\theta(\boldsymbol{x}) = -\log \tilde{p}(\boldsymbol{x})$  ( $\tilde{p}(\boldsymbol{x})$  is not necessarily normalized).
  - $\mathcal{K}(\boldsymbol{v}) = \frac{1}{2} \boldsymbol{v}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{v}$  (Negative log-likelihood of a Gaussian distribution).

# Hamiltonian Monte Carlo

## From Physics to Statistics

The Hamiltonian Monte Carlo (HMC) is a physics-inspired sampling method that uses the Hamiltonian function defined as follows:

$$\mathcal{H}(\boldsymbol{x}, \boldsymbol{v}) = E_\theta(\boldsymbol{x}) + \mathcal{K}(\boldsymbol{v})$$

- In physics:
  - $E_\theta(\boldsymbol{x})$ : Represents the potential energy at position  $\boldsymbol{x}$ .
  - $\mathcal{K}(\boldsymbol{v})$ : Represents the kinetic energy at speed  $\boldsymbol{v}$ .
- In Statistical Setting:
  - $E_\theta(\boldsymbol{x}) = -\log \tilde{p}(\boldsymbol{x})$  ( $\tilde{p}(\boldsymbol{x})$  is not necessarily normalized).
  - $\mathcal{K}(\boldsymbol{v}) = \frac{1}{2} \boldsymbol{v}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{v}$  (Negative log-likelihood of a Gaussian distribution).

*Langevin Monte Carlo*, also known as *Metropolis Adjusted Langevin Algorithm* (MALA), is a special case of Hamiltonian Monte Carlo.

# Langevin Monte Carlo Algorithm

---

**Algorithm 13** Sampling from  $p_\theta(\mathbb{X})$  using Langevin Monte Carlo Algorithm

---

1: **procedure** LANGEVIN MONTE CARLO

# Langevin Monte Carlo Algorithm

---

**Algorithm 14** Sampling from  $p_\theta(\mathbb{X})$  using Langevin Monte Carlo Algorithm

---

- 1: **procedure** LANGEVIN MONTE CARLO
- 2:     **Initialization:**  $x^1$

# Langevin Monte Carlo Algorithm

---

**Algorithm 15** Sampling from  $p_\theta(\mathbb{X})$  using Langevin Monte Carlo Algorithm

---

```
1: procedure LANGEVIN MONTE CARLO
2:   Initialization:  $x^1$ 
3:   for  $s = 1 : T$  do
```

# Langevin Monte Carlo Algorithm

---

**Algorithm 16** Sampling from  $p_\theta(\mathbb{X})$  using Langevin Monte Carlo Algorithm

---

```
1: procedure LANGEVIN MONTE CARLO
2:   Initialization:  $x^1$ 
3:   for  $s = 1 : T$  do
4:     Sample momentum  $v^s \sim \mathcal{N}(\mathbf{0}, \Sigma)$ 
```

# Langevin Monte Carlo Algorithm

---

**Algorithm 17** Sampling from  $p_\theta(\mathbb{X})$  using Langevin Monte Carlo Algorithm

---

```
1: procedure LANGEVIN MONTE CARLO
2:   Initialization:  $\mathbf{x}^1$ 
3:   for  $s = 1 : T$  do
4:     Sample momentum  $\mathbf{v}^s \sim \mathcal{N}(\mathbf{0}, \Sigma)$ 
5:      $\mathbf{x}^p = \mathbf{x}^s - \frac{\eta^2}{2} \Sigma^{-1} \nabla E_\theta(\mathbf{x}^s) + \eta \Sigma^{-1} \mathbf{v}^s$ 
```

# Langevin Monte Carlo Algorithm

---

**Algorithm 18** Sampling from  $p_\theta(\mathbb{X})$  using Langevin Monte Carlo Algorithm

---

```
1: procedure LANGEVIN MONTE CARLO
2:   Initialization:  $\mathbf{x}^1$ 
3:   for  $s = 1 : T$  do
4:     Sample momentum  $\mathbf{v}^s \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ 
5:      $\mathbf{x}^p = \mathbf{x}^s - \frac{\eta^2}{2} \boldsymbol{\Sigma}^{-1} \nabla E_\theta(\mathbf{x}^s) + \eta \boldsymbol{\Sigma}^{-1} \mathbf{v}^s$ 
6:      $\mathbf{v}^p = \mathbf{v}^s - \frac{\eta^2}{2} \nabla E_\theta(\mathbf{x}^s) - \frac{\eta}{2} \nabla E_\theta(\mathbf{x}^p)$ 
```

# Langevin Monte Carlo Algorithm

---

**Algorithm 19** Sampling from  $p_\theta(\mathbb{X})$  using Langevin Monte Carlo Algorithm

---

1: **procedure** LANGEVIN MONTE CARLO

2:     **Initialization:**  $\mathbf{x}^1$

3:     **for**  $s = 1 : T$  **do**

4:         Sample momentum  $\mathbf{v}^s \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$

5:          $\mathbf{x}^p = \mathbf{x}^s - \frac{\eta^2}{2} \boldsymbol{\Sigma}^{-1} \nabla E_\theta(\mathbf{x}^s) + \eta \boldsymbol{\Sigma}^{-1} \mathbf{v}^s$

6:          $\mathbf{v}^p = \mathbf{v}^s - \frac{\eta^2}{2} \nabla E_\theta(\mathbf{x}^s) - \frac{\eta}{2} \nabla E_\theta(\mathbf{x}^p)$

7:         Compute:

$$A = \min \left( 1, \frac{\exp(-\mathcal{H}(\mathbf{x}^p, \mathbf{v}^p))}{\exp(-\mathcal{H}(\mathbf{x}^s, \mathbf{v}^s))} \right)$$

# Langevin Monte Carlo Algorithm

---

**Algorithm 20** Sampling from  $p_\theta(\mathbb{X})$  using Langevin Monte Carlo Algorithm

---

```
1: procedure LANGEVIN MONTE CARLO
2:   Initialization:  $\mathbf{x}^1$ 
3:   for  $s = 1 : T$  do
4:     Sample momentum  $\mathbf{v}^s \sim \mathcal{N}(\mathbf{0}, \Sigma)$ 
5:      $\mathbf{x}^p = \mathbf{x}^s - \frac{\eta^2}{2} \Sigma^{-1} \nabla E_\theta(\mathbf{x}^s) + \eta \Sigma^{-1} \mathbf{v}^s$ 
6:      $\mathbf{v}^p = \mathbf{v}^s - \frac{\eta^2}{2} \nabla E_\theta(\mathbf{x}^s) - \frac{\eta}{2} \nabla E_\theta(\mathbf{x}^p)$ 
7:     Compute:
        
$$A = \min \left( 1, \frac{\exp(-\mathcal{H}(\mathbf{x}^p, \mathbf{v}^p))}{\exp(-\mathcal{H}(\mathbf{x}^s, \mathbf{v}^s))} \right)$$

8:     Sample  $u \sim U(0, 1)$ 
```

## Langevin Monte Carlo Algorithm

**Algorithm 21** Sampling from  $p_\theta(\mathbb{X})$  using Langevin Monte Carlo Algorithm

### 1: procedure LANGEVIN MONTE CARLO

## 2: Initialization: $x^1$

3:       **for**  $s = 1 : T$  **do**

4: Sample momentum  $\mathbf{v}^s \sim \mathcal{N}(\mathbf{0}, \Sigma)$

$$5: \quad \quad \quad \boldsymbol{x}^p = \boldsymbol{x}^s - \frac{\eta^2}{2} \boldsymbol{\Sigma}^{-1} \nabla E_{\theta}(\boldsymbol{x}^s) + \eta \boldsymbol{\Sigma}^{-1} \boldsymbol{v}^s$$

$$6: \quad \quad \quad \boldsymbol{v}^p = \boldsymbol{v}^s - \frac{\eta^2}{2} \nabla E_{\theta}(\boldsymbol{x}^s) - \frac{\eta}{2} \nabla E_{\theta}(\boldsymbol{x}^p)$$

7: Compute:

$$A = \min \left( 1, \frac{\exp(-\mathcal{H}(\mathbf{x}^p, \mathbf{v}^p))}{\exp(-\mathcal{H}(\mathbf{x}^s, \mathbf{v}^s))} \right)$$

8:                   Sample  $u \sim U(0, 1)$

9: Generate new sample:

#Metropolis-Hastings Correction

$$\boldsymbol{x}^{s+1} = \begin{cases} \boldsymbol{x}^p & \text{if } u \leq A(\text{Accept}) \\ \boldsymbol{x}^s & \text{if } u > A(\text{Reject}) \end{cases}$$

# Langevin Monte Carlo Algorithm

---

**Algorithm 22** Sampling from  $p_\theta(\mathbb{X})$  using Langevin Monte Carlo Algorithm

---

```
1: procedure LANGEVIN MONTE CARLO
2:   Initialization:  $\mathbf{x}^1$ 
3:   for  $s = 1 : T$  do
4:     Sample momentum  $\mathbf{v}^s \sim \mathcal{N}(\mathbf{0}, \Sigma)$ 
5:      $\mathbf{x}^p = \mathbf{x}^s - \frac{\eta^2}{2} \Sigma^{-1} \nabla E_\theta(\mathbf{x}^s) + \eta \Sigma^{-1} \mathbf{v}^s$ 
6:      $\mathbf{v}^p = \mathbf{v}^s - \frac{\eta^2}{2} \nabla E_\theta(\mathbf{x}^s) - \frac{\eta}{2} \nabla E_\theta(\mathbf{x}^p)$ 
7:     Compute:
        
$$A = \min \left( 1, \frac{\exp(-\mathcal{H}(\mathbf{x}^p, \mathbf{v}^p))}{\exp(-\mathcal{H}(\mathbf{x}^s, \mathbf{v}^s))} \right)$$

8:     Sample  $u \sim U(0, 1)$ 
9:     Generate new sample: #Metropolis-Hastings Correction
        
$$\mathbf{x}^{s+1} = \begin{cases} \mathbf{x}^p & \text{if } u \leq A(\text{Accept}) \\ \mathbf{x}^s & \text{if } u > A(\text{Reject}) \end{cases}$$

10:    end for
```

---

# Langevin Monte Carlo Algorithm

---

**Algorithm 23** Sampling from  $p_\theta(\mathbb{X})$  using Langevin Monte Carlo Algorithm

---

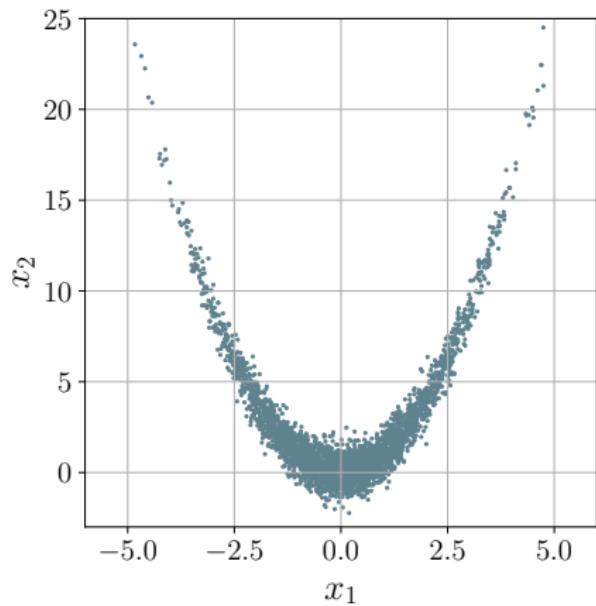
```
1: procedure LANGEVIN MONTE CARLO
2:   Initialization:  $\mathbf{x}^1$ 
3:   for  $s = 1 : T$  do
4:     Sample momentum  $\mathbf{v}^s \sim \mathcal{N}(\mathbf{0}, \Sigma)$ 
5:      $\mathbf{x}^p = \mathbf{x}^s - \frac{\eta^2}{2} \Sigma^{-1} \nabla E_\theta(\mathbf{x}^s) + \eta \Sigma^{-1} \mathbf{v}^s$ 
6:      $\mathbf{v}^p = \mathbf{v}^s - \frac{\eta^2}{2} \nabla E_\theta(\mathbf{x}^s) - \frac{\eta}{2} \nabla E_\theta(\mathbf{x}^p)$ 
7:     Compute:
        
$$A = \min \left( 1, \frac{\exp(-\mathcal{H}(\mathbf{x}^p, \mathbf{v}^p))}{\exp(-\mathcal{H}(\mathbf{x}^s, \mathbf{v}^s))} \right)$$

8:     Sample  $u \sim U(0, 1)$ 
9:     Generate new sample: #Metropolis-Hastings Correction
        
$$\mathbf{x}^{s+1} = \begin{cases} \mathbf{x}^p & \text{if } u \leq A(\text{Accept}) \\ \mathbf{x}^s & \text{if } u > A(\text{Reject}) \end{cases}$$

10:    end for
11: end procedure
```

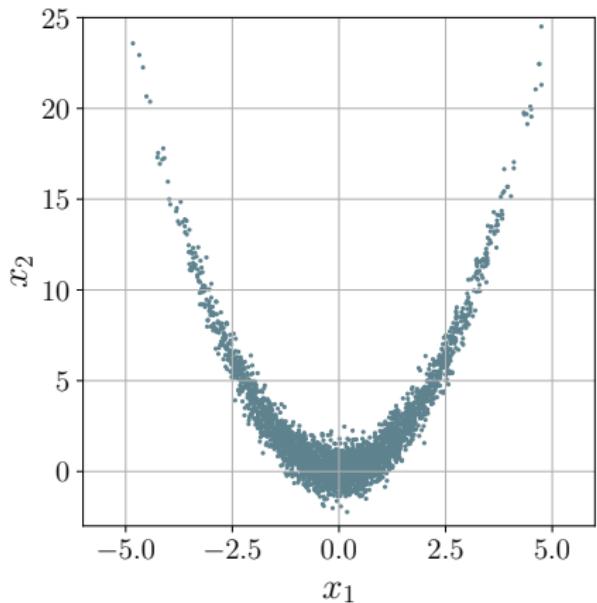
---

# Langevin Monte Carlo Algorithm

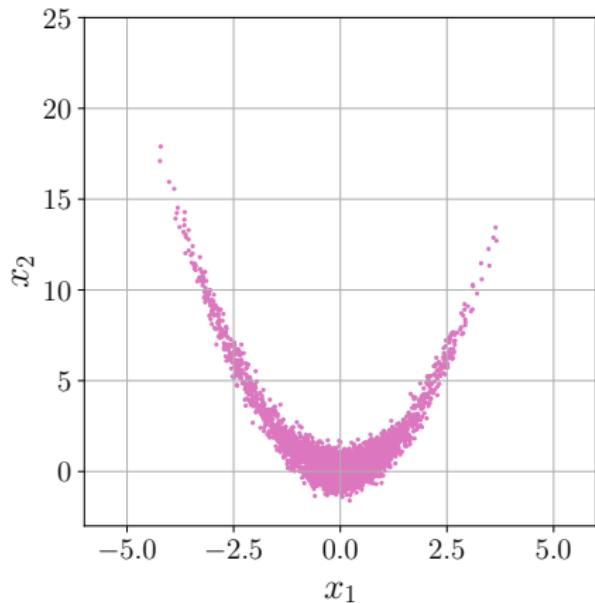


(a) Samples from true distribution

# Langevin Monte Carlo Algorithm



(a) Samples from true distribution



(b) Samples from LMC

**Figure:** The result of LMC algorithm for toy *banana* distribution. (number of samples:  $10^4$  and burned-in samples: 200) (source: [4])

# Langevin Monte Carlo Algorithm

## Cons and Pros

- Pros:
  - LMC is more efficient than MH in high-dimensional data

# Langevin Monte Carlo Algorithm

## Cons and Pros

- Pros:
  - LMC is more efficient than MH in high-dimensional data
  - LMC utilizes the gradient information and does not need proposal distribution

# Langevin Monte Carlo Algorithm

## Cons and Pros

- Pros:
  - LMC is more efficient than MH in high-dimensional data
  - LMC utilizes the gradient information and does not need proposal distribution
  - LMC is guaranteed to converge to true distribution under certain conditions

# Langevin Monte Carlo Algorithm

## Cons and Pros

- Pros:
  - LMC is more efficient than MH in high-dimensional data
  - LMC utilizes the gradient information and does not need proposal distribution
  - LMC is guaranteed to converge to true distribution under certain conditions
- Cons:
  - LMC requires gradient computations which are more complex than MH

# Langevin Monte Carlo Algorithm

## Cons and Pros

- Pros:
  - LMC is more efficient than MH in high-dimensional data
  - LMC utilizes the gradient information and does not need proposal distribution
  - LMC is guaranteed to converge to true distribution under certain conditions
- Cons:
  - LMC requires gradient computations which are more complex than MH
  - LMC is sensitive to stepsize  $\eta$  value

# Langevin Monte Carlo Algorithm

## Cons and Pros

- Pros:
  - LMC is more efficient than MH in high-dimensional data
  - LMC utilizes the gradient information and does not need proposal distribution
  - LMC is guaranteed to converge to true distribution under certain conditions
- Cons:
  - LMC requires gradient computations which are more complex than MH
  - LMC is sensitive to stepsize  $\eta$  value
  - LMC may struggle with sampling efficiently from distributions with multiple modes and may trapped in a local mode.

# Langevin Monte Carlo Algorithm

## Cons and Pros

- Pros:
  - LMC is more efficient than MH in high-dimensional data
  - LMC utilizes the gradient information and does not need proposal distribution
  - LMC is guaranteed to converge to true distribution under certain conditions
- Cons:
  - LMC requires gradient computations which are more complex than MH
  - LMC is sensitive to stepsize  $\eta$  value
  - LMC may struggle with sampling efficiently from distributions with multiple modes and may trapped in a local mode.
  - It is challenging to assess the convergence of the chain to target distribution.

## Section 4

### Contrastive Divergence

# Minimizing KL Divergence

## Likelihood-based Training

One approach to minimize the distance between  $p_{\text{data}}$  and  $p_{\theta}$  is:

$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \frac{1}{|\mathcal{D}|} \sum_{i=1}^N \log p_{\theta}(\mathbf{x}_i)$$

Thus we need to calculate  $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x})$ . So:

# Minimizing KL Divergence

## Likelihood-based Training

One approach to minimize the distance between  $p_{\text{data}}$  and  $p_{\theta}$  is:

$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \frac{1}{|\mathcal{D}|} \sum_{i=1}^N \log p_{\theta}(\mathbf{x}_i)$$

Thus we need to calculate  $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x})$ . So:

$$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x})$$

# Minimizing KL Divergence

## Likelihood-based Training

One approach to minimize the distance between  $p_{\text{data}}$  and  $p_{\theta}$  is:

$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \frac{1}{|\mathcal{D}|} \sum_{i=1}^N \log p_{\theta}(\mathbf{x}_i)$$

Thus we need to calculate  $\nabla_{\theta} \log p_{\theta}(\mathbf{x})$ . So:

$$\nabla_{\theta} \log p_{\theta}(\mathbf{x}) \stackrel{(a)}{=} \nabla_{\theta} \log \frac{\exp(-E_{\theta}(\mathbf{x}))}{Z_{\theta}}$$

# Minimizing KL Divergence

## Likelihood-based Training

One approach to minimize the distance between  $p_{\text{data}}$  and  $p_{\theta}$  is:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \frac{1}{|\mathcal{D}|} \sum_{i=1}^N \log p_{\theta}(\mathbf{x}_i)$$

Thus we need to calculate  $\nabla_{\theta} \log p_{\theta}(\mathbf{x})$ . So:

$$\nabla_{\theta} \log p_{\theta}(\mathbf{x}) \stackrel{(a)}{=} \nabla_{\theta} \log \frac{\exp(-E_{\theta}(\mathbf{x}))}{Z_{\theta}} = -\underbrace{\nabla_{\theta} E_{\theta}(\mathbf{x})}_{\text{I}} - \underbrace{\nabla_{\theta} \log Z_{\theta}}_{\text{II}}$$

# Minimizing KL Divergence

## Likelihood-based Training

One approach to minimize the distance between  $p_{\text{data}}$  and  $p_{\theta}$  is:

$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \frac{1}{|\mathcal{D}|} \sum_{i=1}^N \log p_{\theta}(\mathbf{x}_i)$$

Thus we need to calculate  $\nabla_{\boldsymbol{\theta}} \log p_{\theta}(\mathbf{x})$ . So:

$$\nabla_{\boldsymbol{\theta}} \log p_{\theta}(\mathbf{x}) \stackrel{(a)}{=} \nabla_{\boldsymbol{\theta}} \log \frac{\exp(-E_{\theta}(\mathbf{x}))}{Z_{\theta}} = -\underbrace{\nabla_{\boldsymbol{\theta}} E_{\theta}(\mathbf{x})}_{\text{I}} - \underbrace{\nabla_{\boldsymbol{\theta}} \log Z_{\theta}}_{\text{II}}$$

where (a) results from the definition of EBM and:

- I is easy to calculate.

# Minimizing KL Divergence

## Likelihood-based Training

One approach to minimize the distance between  $p_{\text{data}}$  and  $p_{\theta}$  is:

$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} \frac{1}{|\mathcal{D}|} \sum_{i=1}^N \log p_{\theta}(\mathbf{x}_i)$$

Thus we need to calculate  $\nabla_{\theta} \log p_{\theta}(\mathbf{x})$ . So:

$$\nabla_{\theta} \log p_{\theta}(\mathbf{x}) \stackrel{(a)}{=} \nabla_{\theta} \log \frac{\exp(-E_{\theta}(\mathbf{x}))}{Z_{\theta}} = -\underbrace{\nabla_{\theta} E_{\theta}(\mathbf{x})}_{\text{I}} - \underbrace{\nabla_{\theta} \log Z_{\theta}}_{\text{II}}$$

where (a) results from the definition of EBM and:

- I is easy to calculate.
- II is intractable as calculating  $Z_{\theta}$  is intractable for EBMs in general.

# Gradient of Logarithm of Partition Function

Revisiting

# Gradient of Logarithm of Partition Function

## Revisiting

$$\nabla_{\theta} \log Z_{\theta} = \nabla_{\theta} \log \int \exp(-E_{\theta}(\boldsymbol{x})) d\boldsymbol{x}$$

# Gradient of Logarithm of Partition Function

## Revisiting

$$\begin{aligned}\nabla_{\theta} \log Z_{\theta} &= \nabla_{\theta} \log \int \exp(-E_{\theta}(\boldsymbol{x})) d\boldsymbol{x} \\ &\stackrel{(a)}{=} \left( \underbrace{\int \exp(-E_{\theta}(\boldsymbol{x})) d\boldsymbol{x}}_{Z_{\theta}} \right)^{-1} \nabla_{\theta} \int \exp(-E_{\theta}(\boldsymbol{x})) d\boldsymbol{x}\end{aligned}$$

# Gradient of Logarithm of Partition Function

## Revisiting

$$\begin{aligned}\nabla_{\theta} \log Z_{\theta} &= \nabla_{\theta} \log \int \exp(-E_{\theta}(\boldsymbol{x})) d\boldsymbol{x} \\ &\stackrel{(a)}{=} \left( \underbrace{\int \exp(-E_{\theta}(\boldsymbol{x})) d\boldsymbol{x}}_{Z_{\theta}} \right)^{-1} \nabla_{\theta} \int \exp(-E_{\theta}(\boldsymbol{x})) d\boldsymbol{x} \\ &\stackrel{(b)}{=} Z_{\theta}^{-1} \int \exp(-E_{\theta}(\boldsymbol{x})) (-\nabla_{\theta} E_{\theta}(\boldsymbol{x})) d\boldsymbol{x}\end{aligned}$$

# Gradient of Logarithm of Partition Function

## Revisiting

$$\begin{aligned}\nabla_{\theta} \log Z_{\theta} &= \nabla_{\theta} \log \int \exp(-E_{\theta}(\boldsymbol{x})) d\boldsymbol{x} \\ &\stackrel{(a)}{=} \left( \underbrace{\int \exp(-E_{\theta}(\boldsymbol{x})) d\boldsymbol{x}}_{Z_{\theta}} \right)^{-1} \nabla_{\theta} \int \exp(-E_{\theta}(\boldsymbol{x})) d\boldsymbol{x} \\ &\stackrel{(b)}{=} Z_{\theta}^{-1} \int \exp(-E_{\theta}(\boldsymbol{x})) (-\nabla_{\theta} E_{\theta}(\boldsymbol{x})) d\boldsymbol{x} \\ &= \int \left( \frac{\exp(-E_{\theta}(\boldsymbol{x}))}{Z_{\theta}} \right) (-\nabla_{\theta} E_{\theta}(\boldsymbol{x})) d\boldsymbol{x}\end{aligned}$$

# Gradient of Logarithm of Partition Function

## Revisiting

$$\begin{aligned}\nabla_{\theta} \log Z_{\theta} &= \nabla_{\theta} \log \int \exp(-E_{\theta}(\boldsymbol{x})) d\boldsymbol{x} \\ &\stackrel{(a)}{=} \left( \underbrace{\int \exp(-E_{\theta}(\boldsymbol{x})) d\boldsymbol{x}}_{Z_{\theta}} \right)^{-1} \nabla_{\theta} \int \exp(-E_{\theta}(\boldsymbol{x})) d\boldsymbol{x} \\ &\stackrel{(b)}{=} Z_{\theta}^{-1} \int \exp(-E_{\theta}(\boldsymbol{x})) (-\nabla_{\theta} E_{\theta}(\boldsymbol{x})) d\boldsymbol{x} \\ &= \int \left( \frac{\exp(-E_{\theta}(\boldsymbol{x}))}{Z_{\theta}} \right) (-\nabla_{\theta} E_{\theta}(\boldsymbol{x})) d\boldsymbol{x} \\ &= \int p_{\theta}(\boldsymbol{x}) (-\nabla_{\theta} E_{\theta}(\boldsymbol{x})) d\boldsymbol{x}\end{aligned}$$

# Gradient of Logarithm of Partition Function

## Revisiting

$$\begin{aligned}\nabla_{\theta} \log Z_{\theta} &= \nabla_{\theta} \log \int \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &\stackrel{(a)}{=} \left( \underbrace{\int \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x}}_{Z_{\theta}} \right)^{-1} \nabla_{\theta} \int \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &\stackrel{(b)}{=} Z_{\theta}^{-1} \int \exp(-E_{\theta}(\mathbf{x})) (-\nabla_{\theta} E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &= \int \left( \frac{\exp(-E_{\theta}(\mathbf{x}))}{Z_{\theta}} \right) (-\nabla_{\theta} E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &= \int p_{\theta}(\mathbf{x}) (-\nabla_{\theta} E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &= \mathbb{E}_{\mathbf{x} \sim p_{\theta}(\mathbb{X})} [-\nabla_{\theta} E_{\theta}(\mathbf{x})]\end{aligned}$$

# Gradient of Logarithm of Partition Function

## Revisiting

$$\begin{aligned}\nabla_{\theta} \log Z_{\theta} &= \nabla_{\theta} \log \int \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &\stackrel{(a)}{=} \left( \underbrace{\int \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x}}_{Z_{\theta}} \right)^{-1} \nabla_{\theta} \int \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &\stackrel{(b)}{=} Z_{\theta}^{-1} \int \exp(-E_{\theta}(\mathbf{x})) (-\nabla_{\theta} E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &= \int \left( \frac{\exp(-E_{\theta}(\mathbf{x}))}{Z_{\theta}} \right) (-\nabla_{\theta} E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &= \int p_{\theta}(\mathbf{x}) (-\nabla_{\theta} E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &= \mathbb{E}_{\mathbf{x} \sim p_{\theta}(\mathbb{X})} [-\nabla_{\theta} E_{\theta}(\mathbf{x})]\end{aligned}$$

where (a) and (b) results from the following equalities:

# Gradient of Logarithm of Partition Function

## Revisiting

$$\begin{aligned}\nabla_{\theta} \log Z_{\theta} &= \nabla_{\theta} \log \int \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &\stackrel{(a)}{=} \left( \underbrace{\int \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x}}_{Z_{\theta}} \right)^{-1} \nabla_{\theta} \int \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &\stackrel{(b)}{=} Z_{\theta}^{-1} \int \exp(-E_{\theta}(\mathbf{x})) (-\nabla_{\theta} E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &= \int \left( \frac{\exp(-E_{\theta}(\mathbf{x}))}{Z_{\theta}} \right) (-\nabla_{\theta} E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &= \int p_{\theta}(\mathbf{x}) (-\nabla_{\theta} E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &= \mathbb{E}_{\mathbf{x} \sim p_{\theta}(\mathbb{X})} [-\nabla_{\theta} E_{\theta}(\mathbf{x})]\end{aligned}$$

where (a) and (b) results from the following equalities:

$$(a) : \nabla_{\theta} \log f(\boldsymbol{\theta}) = (f(\boldsymbol{\theta}))^{-1} \nabla_{\theta} f(\boldsymbol{\theta})$$

# Gradient of Logarithm of Partition Function

## Revisiting

$$\begin{aligned}\nabla_{\theta} \log Z_{\theta} &= \nabla_{\theta} \log \int \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &\stackrel{(a)}{=} \left( \underbrace{\int \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x}}_{Z_{\theta}} \right)^{-1} \nabla_{\theta} \int \exp(-E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &\stackrel{(b)}{=} Z_{\theta}^{-1} \int \exp(-E_{\theta}(\mathbf{x})) (-\nabla_{\theta} E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &= \int \left( \frac{\exp(-E_{\theta}(\mathbf{x}))}{Z_{\theta}} \right) (-\nabla_{\theta} E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &= \int p_{\theta}(\mathbf{x}) (-\nabla_{\theta} E_{\theta}(\mathbf{x})) d\mathbf{x} \\ &= \mathbb{E}_{\mathbf{x} \sim p_{\theta}(\mathbb{X})} [-\nabla_{\theta} E_{\theta}(\mathbf{x})]\end{aligned}$$

where (a) and (b) results from the following equalities:

$$(a) : \nabla_{\theta} \log f(\boldsymbol{\theta}) = (f(\boldsymbol{\theta}))^{-1} \nabla_{\theta} f(\boldsymbol{\theta}), \quad (b) : \nabla_{\theta} \exp(f(\theta)) = \exp(f(\theta)) \nabla_{\theta} f(\theta)$$

# Gradient of Logarithm of Partition Function

## Revisiting (cont.)

As we see, we have:

$$\nabla_{\theta} \log Z_{\theta} = \mathbb{E}_{\mathbf{x} \sim p_{\theta}(\mathbb{X})} [-\nabla_{\theta} E_{\theta}(\mathbf{x})]$$

# Gradient of Logarithm of Partition Function

## Revisiting (cont.)

As we see, we have:

$$\nabla_{\theta} \log Z_{\theta} = \mathbb{E}_{\mathbf{x} \sim p_{\theta}(\mathbb{X})} [-\nabla_{\theta} E_{\theta}(\mathbf{x})]$$

Thus we can calculate using the following steps:

# Gradient of Logarithm of Partition Function

## Revisiting (cont.)

As we see, we have:

$$\nabla_{\theta} \log Z_{\theta} = \mathbb{E}_{\mathbf{x} \sim p_{\theta}(\mathbb{X})} [-\nabla_{\theta} E_{\theta}(\mathbf{x})]$$

Thus we can calculate using the following steps:

- Sample EBM using MH or LMC  $\{\tilde{\mathbf{x}}_s\}_{s=1}^S$ .

# Gradient of Logarithm of Partition Function

## Revisiting (cont.)

As we see, we have:

$$\nabla_{\theta} \log Z_{\theta} = \mathbb{E}_{\mathbf{x} \sim p_{\theta}(\mathbb{X})} [-\nabla_{\theta} E_{\theta}(\mathbf{x})]$$

Thus we can calculate using the following steps:

- Sample EBM using MH or LMC  $\{\tilde{\mathbf{x}}_s\}_{s=1}^S$ .
- Estimate gradient using Monte Carlo as:

$$\nabla_{\theta} \log Z_{\theta} = -\frac{1}{S} \sum_{s=1}^S \nabla_{\theta} E_{\theta}(\tilde{\mathbf{x}}_s)$$

# Gradient of Logarithm of Partition Function

## Revisiting (cont.)

As we see, we have:

$$\nabla_{\theta} \log Z_{\theta} = \mathbb{E}_{\mathbf{x} \sim p_{\theta}(\mathbb{X})} [-\nabla_{\theta} E_{\theta}(\mathbf{x})]$$

Thus we can calculate using the following steps:

- Sample EBM using MH or LMC  $\{\tilde{\mathbf{x}}_s\}_{s=1}^S$ .
- Estimate gradient using Monte Carlo as:

$$\nabla_{\theta} \log Z_{\theta} = -\frac{1}{S} \sum_{s=1}^S \nabla_{\theta} E_{\theta}(\tilde{\mathbf{x}}_s)$$

## Challenge

- MCMC generates correlated samples so all the adjacent samples cannot be used in the calculation (you need to run several MCMC samplers)

# Gradient of Logarithm of Partition Function

## Revisiting (cont.)

As we see, we have:

$$\nabla_{\theta} \log Z_{\theta} = \mathbb{E}_{\mathbf{x} \sim p_{\theta}(\mathbb{X})} [-\nabla_{\theta} E_{\theta}(\mathbf{x})]$$

Thus we can calculate using the following steps:

- Sample EBM using MH or LMC  $\{\tilde{\mathbf{x}}_s\}_{s=1}^S$ .
- Estimate gradient using Monte Carlo as:

$$\nabla_{\theta} \log Z_{\theta} = -\frac{1}{S} \sum_{s=1}^S \nabla_{\theta} E_{\theta}(\tilde{\mathbf{x}}_s)$$

## Challenge

- MCMC generates correlated samples so all the adjacent samples cannot be used in the calculation (you need to run several MCMC samplers)
- Assessing the convergence of MCMC is challenging.

# Gradient of Logarithm of Partition Function

## Revisiting (cont.)

As we see, we have:

$$\nabla_{\theta} \log Z_{\theta} = \mathbb{E}_{\mathbf{x} \sim p_{\theta}(\mathbb{X})} [-\nabla_{\theta} E_{\theta}(\mathbf{x})]$$

Thus we can calculate using the following steps:

- Sample EBM using MH or LMC  $\{\tilde{\mathbf{x}}_s\}_{s=1}^S$ .
- Estimate gradient using Monte Carlo as:

$$\nabla_{\theta} \log Z_{\theta} = -\frac{1}{S} \sum_{s=1}^S \nabla_{\theta} E_{\theta}(\tilde{\mathbf{x}}_s)$$

## Challenge

- MCMC generates correlated samples so all the adjacent samples cannot be used in the calculation (you need to run several MCMC samplers)
  - Assessing the convergence of MCMC is challenging.
- ☞ Thus the above solution is not practical.

# Contrastive Divergence

## Intuition

*Contrastive Divergence (CD)* approximate maximum likelihood such that:

# Contrastive Divergence

## Intuition

*Contrastive Divergence (CD)* approximate maximum likelihood such that:

- You don't need to calculate  $\nabla_{\theta} \log Z_{\theta}$  anymore.

# Contrastive Divergence

## Intuition

*Contrastive Divergence (CD)* approximate maximum likelihood such that:

- You don't need to calculate  $\nabla_{\theta} \log Z_{\theta}$  anymore.
- You can use the sample in  $t$ -th iteration of any MCMC sampling method.

# Contrastive Divergence

## Intuition

*Contrastive Divergence (CD)* approximate maximum likelihood such that:

- You don't need to calculate  $\nabla_{\theta} \log Z_{\theta}$  anymore.
- You can use the sample in  $t$ -th iteration of any MCMC sampling method.

## Contrastive Divergence

Assume we define:

# Contrastive Divergence

## Intuition

*Contrastive Divergence (CD)* approximate maximum likelihood such that:

- You don't need to calculate  $\nabla_{\theta} \log Z_{\theta}$  anymore.
- You can use the sample in  $t$ -th iteration of any MCMC sampling method.

## Contrastive Divergence

Assume we define:

$$\mathbf{x} \sim p_{\theta}^t(\mathbb{X})$$

# Contrastive Divergence

## Intuition

*Contrastive Divergence (CD)* approximate maximum likelihood such that:

- You don't need to calculate  $\nabla_{\theta} \log Z_{\theta}$  anymore.
- You can use the sample in  $t$ -th iteration of any MCMC sampling method.

## Contrastive Divergence

Assume we define:

$$\mathbf{x} \sim p_{\theta}^t(\mathbb{X}) \Leftrightarrow \mathbf{x} \sim \text{MCMC}(\text{target} = p_{\theta}, T = t, \text{init} = \mathbf{x}_0), \mathbf{x}_0 \sim p_{\text{data}}(\mathbb{X})$$

# Contrastive Divergence

## Intuition

*Contrastive Divergence (CD)* approximate maximum likelihood such that:

- You don't need to calculate  $\nabla_{\theta} \log Z_{\theta}$  anymore.
- You can use the sample in  $t$ -th iteration of any MCMC sampling method.

## Contrastive Divergence

Assume we define:

$$\mathbf{x} \sim p_{\theta}^t(\mathbb{X}) \Leftrightarrow \mathbf{x} \sim \text{MCMC}(\text{target} = p_{\theta}, T = t, \text{init} = \mathbf{x}_0), \mathbf{x}_0 \sim p_{\text{data}}(\mathbb{X})$$

We can readily conclude  $p^0 = p_{\text{data}}$  and  $p_{\theta}^{\infty} = p_{\theta}$ . Then CD is defined as:

# Contrastive Divergence

## Intuition

*Contrastive Divergence* (CD) approximate maximum likelihood such that:

- You don't need to calculate  $\nabla_{\theta} \log Z_{\theta}$  anymore.
- You can use the sample in  $t$ -th iteration of any MCMC sampling method.

## Contrastive Divergence

Assume we define:

$$\mathbf{x} \sim p_{\theta}^t(\mathbb{X}) \Leftrightarrow \mathbf{x} \sim \text{MCMC}(\text{target} = p_{\theta}, T = t, \text{init} = \mathbf{x}_0), \mathbf{x}_0 \sim p_{\text{data}}(\mathbb{X})$$

We can readily conclude  $p^0 = p_{\text{data}}$  and  $p_{\theta}^{\infty} = p_{\theta}$ . Then CD is defined as:

$$\text{CD}_t = \text{KL}(p_{\text{data}} \| p_{\theta}) - \text{KL}(p_{\theta}^t \| p_{\theta})$$

# Contrastive Divergence

## Intuition

*Contrastive Divergence (CD)* approximate maximum likelihood such that:

- You don't need to calculate  $\nabla_{\theta} \log Z_{\theta}$  anymore.
- You can use the sample in  $t$ -th iteration of any MCMC sampling method.

## Contrastive Divergence

Assume we define:

$$\mathbf{x} \sim p_{\theta}^t(\mathbb{X}) \Leftrightarrow \mathbf{x} \sim \text{MCMC}(\text{target} = p_{\theta}, T = t, \text{init} = \mathbf{x}_0), \mathbf{x}_0 \sim p_{\text{data}}(\mathbb{X})$$

We can readily conclude  $p^0 = p_{\text{data}}$  and  $p_{\theta}^{\infty} = p_{\theta}$ . Then CD is defined as:

$$\begin{aligned}\text{CD}_t &= \text{KL}(p_{\text{data}} \| p_{\theta}) - \text{KL}(p_{\theta}^t \| p_{\theta}) \\ &= \text{KL}(p^0 \| p_{\theta}^{\infty}) - \text{KL}(p_{\theta}^t \| p_{\theta}^{\infty})\end{aligned}$$

# Contrastive Divergence

$p_{data}$   
 $p^{\theta}$  

Figure: Geometrical interpretation of CD (circles can move when  $\theta$  changes)

# Contrastive Divergence



Figure: Geometrical interpretation of CD (circles can move when  $\theta$  changes)

# Contrastive Divergence

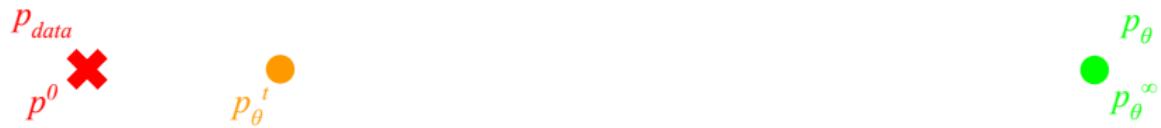


Figure: Geometrical interpretation of CD (circles can move when  $\theta$  changes)

# Contrastive Divergence

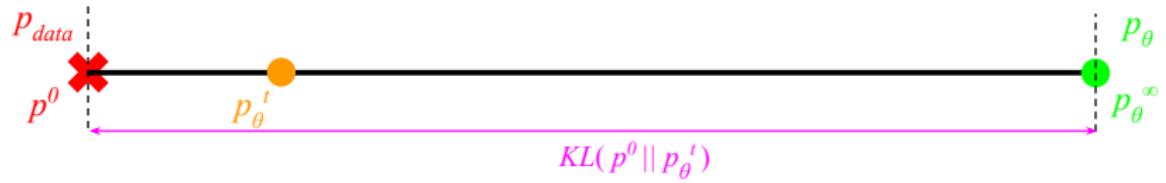


Figure: Geometrical interpretation of CD (circles can move when  $\theta$  changes)

# Contrastive Divergence

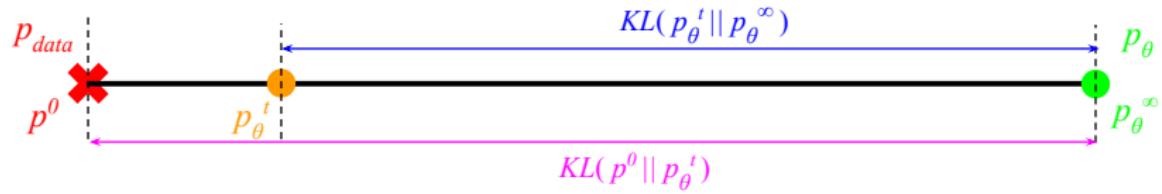


Figure: Geometrical interpretation of CD (circles can move when  $\theta$  changes)

# Contrastive Divergence

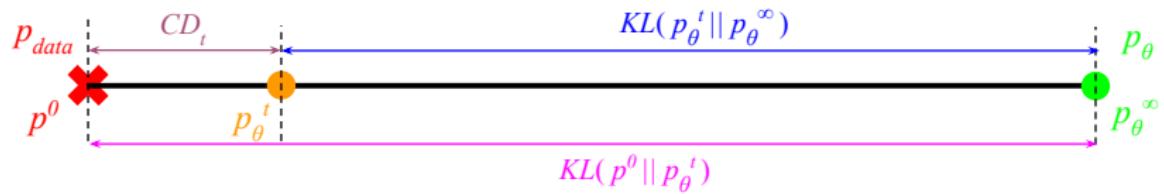


Figure: Geometrical interpretation of CD (circles can move when  $\theta$  changes)

# Contrastive Divergence

## Properties

- Non-negativity:  $\text{KL}(p_{\text{data}} \| p_{\theta}) \geq \text{KL}(p_{\theta}^t \| p_{\theta}) \Rightarrow \text{CD}_t \geq 0$

# Contrastive Divergence

## Properties

- Non-negativity:  $\text{KL}(p_{\text{data}} \| p_{\theta}) \geq \text{KL}(p_{\theta}^t \| p_{\theta}) \Rightarrow \text{CD}_t \geq 0$
- If  $p_{\text{data}} = p_{\theta}$ , then MCMC guarantees  $p_{\theta}^t = p_{\text{data}} = p_{\theta}$  and thus  $\text{CD}_t = 0$

# Contrastive Divergence

## Properties

- Non-negativity:  $\text{KL}(p_{\text{data}} \| p_{\theta}) \geq \text{KL}(p_{\theta}^t \| p_{\theta}) \Rightarrow \text{CD}_t \geq 0$
- If  $p_{\text{data}} = p_{\theta}$ , then MCMC guarantees  $p_{\theta}^t = p_{\text{data}} = p_{\theta}$  and thus  $\text{CD}_t = 0$
- $\lim_{t \rightarrow \infty} \text{CD}_t = \text{KL}(p_{\text{data}} \| p_{\theta})$

# Contrastive Divergence

## Properties

- Non-negativity:  $\text{KL}(p_{\text{data}} \| p_{\theta}) \geq \text{KL}(p_{\theta}^t \| p_{\theta}) \Rightarrow \text{CD}_t \geq 0$
- If  $p_{\text{data}} = p_{\theta}$ , then MCMC guarantees  $p_{\theta}^t = p_{\text{data}} = p_{\theta}$  and thus  $\text{CD}_t = 0$
- $\lim_{t \rightarrow \infty} \text{CD}_t = \text{KL}(p_{\text{data}} \| p_{\theta})$
- If  $p_{\text{data}} = p_{\theta}^t$ , then the Markov chain guarantees  $p_{\text{data}} = p_{\theta}$ . So the contrastive divergence can only be zero if the model is perfect.

# Contrastive Divergence

## Properties

- Non-negativity:  $\text{KL}(p_{\text{data}} \| p_{\theta}) \geq \text{KL}(p_{\theta}^t \| p_{\theta}) \Rightarrow \text{CD}_t \geq 0$
- If  $p_{\text{data}} = p_{\theta}$ , then MCMC guarantees  $p_{\theta}^t = p_{\text{data}} = p_{\theta}$  and thus  $\text{CD}_t = 0$
- $\lim_{t \rightarrow \infty} \text{CD}_t = \text{KL}(p_{\text{data}} \| p_{\theta})$
- If  $p_{\text{data}} = p_{\theta}^t$ , then the Markov chain guarantees  $p_{\text{data}} = p_{\theta}$ . So the contrastive divergence can only be zero if the model is perfect.

## Gradient Vector

$$\nabla_{\theta} \text{CD}_t = \nabla_{\theta} \text{KL}(p_{\text{data}} \| p_{\theta}) - \nabla_{\theta} \text{KL}(p_{\theta}^t \| p_{\theta})$$

So we need to calculate each of the gradient vectors in the RHS

# Contrastive Divergence

## First Gradient Vector in RHS

$$\nabla_{\theta} \text{KL}(p_{\text{data}} \| p_{\theta}) = \nabla_{\theta} \overbrace{\int_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x})}{p_{\theta}(\mathbf{x})} d\mathbf{x}}^{A(p_{\text{data}}, p_{\theta})}$$

# Contrastive Divergence

## First Gradient Vector in RHS

$$\begin{aligned}\nabla_{\theta} \text{KL}(p_{\text{data}} \| p_{\theta}) &= \nabla_{\theta} \int_{\boldsymbol{x}} \overbrace{p_{\text{data}}(\boldsymbol{x}) \log \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\theta}(\boldsymbol{x})}}^{A(p_{\text{data}}, p_{\theta})} d\boldsymbol{x} \\ &= \int_{\boldsymbol{x}} \left[ \frac{\partial A}{\partial p_{\text{data}}} \overbrace{\nabla_{\theta} p_{\text{data}}(\boldsymbol{x})}^{=0} + \frac{\partial A}{\partial p_{\theta}} \nabla_{\theta} p_{\theta}(\boldsymbol{x}) \right] d\boldsymbol{x}\end{aligned}$$

# Contrastive Divergence

## First Gradient Vector in RHS

$$\begin{aligned}\nabla_{\theta} \text{KL}(p_{\text{data}} \| p_{\theta}) &= \nabla_{\theta} \overbrace{\int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\theta}(\boldsymbol{x})} d\boldsymbol{x}}^{A(p_{\text{data}}, p_{\theta})} \\ &= \int_{\boldsymbol{x}} \left[ \frac{\partial A}{\partial p_{\text{data}}} \overbrace{\nabla_{\theta} p_{\text{data}}(\boldsymbol{x})}^{=0} + \frac{\partial A}{\partial p_{\theta}} \nabla_{\theta} p_{\theta}(\boldsymbol{x}) \right] d\boldsymbol{x} \\ &= \int_{\boldsymbol{x}} \left[ -\frac{p_{\text{data}}(\boldsymbol{x})}{p_{\theta}(\boldsymbol{x})} \nabla_{\theta} p_{\theta}(\boldsymbol{x}) \right] d\boldsymbol{x} = \int_{\boldsymbol{x}} \left[ p_{\text{data}}(\boldsymbol{x}) \underbrace{\left( -\frac{\nabla_{\theta} p_{\theta}(\boldsymbol{x})}{p_{\theta}(\boldsymbol{x})} \right)}_{-\nabla_{\theta} \log p_{\theta}(\boldsymbol{x})} \right] d\boldsymbol{x}\end{aligned}$$

# Contrastive Divergence

## First Gradient Vector in RHS

$$\begin{aligned}\nabla_{\theta} \text{KL}(p_{\text{data}} \| p_{\theta}) &= \nabla_{\theta} \overbrace{\int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\theta}(\boldsymbol{x})} d\boldsymbol{x}}^{A(p_{\text{data}}, p_{\theta})} \\&= \int_{\boldsymbol{x}} \left[ \frac{\partial A}{\partial p_{\text{data}}} \overbrace{\nabla_{\theta} p_{\text{data}}(\boldsymbol{x})}^{=0} + \frac{\partial A}{\partial p_{\theta}} \nabla_{\theta} p_{\theta}(\boldsymbol{x}) \right] d\boldsymbol{x} \\&= \int_{\boldsymbol{x}} \left[ -\frac{p_{\text{data}}(\boldsymbol{x})}{p_{\theta}(\boldsymbol{x})} \nabla_{\theta} p_{\theta}(\boldsymbol{x}) \right] d\boldsymbol{x} = \int_{\boldsymbol{x}} \left[ p_{\text{data}}(\boldsymbol{x}) \underbrace{\left( -\frac{\nabla_{\theta} p_{\theta}(\boldsymbol{x})}{p_{\theta}(\boldsymbol{x})} \right)}_{-\nabla_{\theta} \log p_{\theta}(\boldsymbol{x})} \right] d\boldsymbol{x} \\&= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \nabla_{\theta} [E_{\theta}(\boldsymbol{x}) + \log Z_{\theta}] d\boldsymbol{x}\end{aligned}$$

# Contrastive Divergence

## First Gradient Vector in RHS

$$\begin{aligned}\nabla_{\theta} \text{KL}(p_{\text{data}} \| p_{\theta}) &= \nabla_{\theta} \overbrace{\int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\theta}(\boldsymbol{x})} d\boldsymbol{x}}^{A(p_{\text{data}}, p_{\theta})} \\&= \int_{\boldsymbol{x}} \left[ \frac{\partial A}{\partial p_{\text{data}}} \overbrace{\nabla_{\theta} p_{\text{data}}(\boldsymbol{x})}^{=0} + \frac{\partial A}{\partial p_{\theta}} \nabla_{\theta} p_{\theta}(\boldsymbol{x}) \right] d\boldsymbol{x} \\&= \int_{\boldsymbol{x}} \left[ -\frac{p_{\text{data}}(\boldsymbol{x})}{p_{\theta}(\boldsymbol{x})} \nabla_{\theta} p_{\theta}(\boldsymbol{x}) \right] d\boldsymbol{x} = \int_{\boldsymbol{x}} \left[ p_{\text{data}}(\boldsymbol{x}) \underbrace{\left( -\frac{\nabla_{\theta} p_{\theta}(\boldsymbol{x})}{p_{\theta}(\boldsymbol{x})} \right)}_{-\nabla_{\theta} \log p_{\theta}(\boldsymbol{x})} \right] d\boldsymbol{x} \\&= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \nabla_{\theta} [E_{\theta}(\boldsymbol{x}) + \log Z_{\theta}] d\boldsymbol{x} \\&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\mathbb{X})} [\nabla_{\theta} E_{\theta}(\boldsymbol{x})] + \nabla_{\theta} \log Z_{\theta}\end{aligned}$$

# Contrastive Divergence

## Second Gradient Vector in RHS

$$\nabla_{\theta} \text{KL}(p_{\theta}^t \| p_{\theta}) = \nabla_{\theta} \underbrace{\int_{\mathbf{x}} p_{\theta}^t(\mathbf{x}) \log \frac{p_{\theta}^t(\mathbf{x})}{p_{\theta}(\mathbf{x})} d\mathbf{x}}_{B(p_{\theta}^t, p_{\theta})}$$

# Contrastive Divergence

## Second Gradient Vector in RHS

$$\begin{aligned}\nabla_{\theta} \text{KL}(p_{\theta}^t \| p_{\theta}) &= \nabla_{\theta} \underbrace{\int_{\mathbf{x}} p_{\theta}^t(\mathbf{x}) \log \frac{p_{\theta}^t(\mathbf{x})}{p_{\theta}(\mathbf{x})} d\mathbf{x}}_{B(p_{\theta}^t, p_{\theta})} \\ &= \int_{\mathbf{x}} \left[ \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) + \frac{\partial B}{\partial p_{\theta}} \nabla_{\theta} p_{\theta}(\mathbf{x}) \right] d\mathbf{x}\end{aligned}$$

# Contrastive Divergence

## Second Gradient Vector in RHS

$$\begin{aligned}\nabla_{\theta} \text{KL}(p_{\theta}^t \| p_{\theta}) &= \nabla_{\theta} \underbrace{\int_{\boldsymbol{x}} p_{\theta}^t(\boldsymbol{x}) \log \frac{p_{\theta}^t(\boldsymbol{x})}{p_{\theta}(\boldsymbol{x})} d\boldsymbol{x}}_{B(p_{\theta}^t, p_{\theta})} \\ &= \int_{\boldsymbol{x}} \left[ \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\boldsymbol{x}) + \frac{\partial B}{\partial p_{\theta}} \nabla_{\theta} p_{\theta}(\boldsymbol{x}) \right] d\boldsymbol{x} \\ &= \int_{\boldsymbol{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\boldsymbol{x}) d\boldsymbol{x} + \int_{\boldsymbol{x}} \left( -\frac{p_{\theta}^t(\boldsymbol{x})}{p_{\theta}(\boldsymbol{x})} \right) \nabla_{\theta} p_{\theta}(\boldsymbol{x}) d\boldsymbol{x}\end{aligned}$$

# Contrastive Divergence

## Second Gradient Vector in RHS

$$\begin{aligned}\nabla_{\theta} \text{KL}(p_{\theta}^t \| p_{\theta}) &= \nabla_{\theta} \underbrace{\int_{\mathbf{x}} p_{\theta}^t(\mathbf{x}) \log \frac{p_{\theta}^t(\mathbf{x})}{p_{\theta}(\mathbf{x})} d\mathbf{x}}_{B(p_{\theta}^t, p_{\theta})} \\&= \int_{\mathbf{x}} \left[ \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) + \frac{\partial B}{\partial p_{\theta}} \nabla_{\theta} p_{\theta}(\mathbf{x}) \right] d\mathbf{x} \\&= \int_{\mathbf{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) d\mathbf{x} + \int_{\mathbf{x}} \left( -\frac{p_{\theta}^t(\mathbf{x})}{p_{\theta}(\mathbf{x})} \right) \nabla_{\theta} p_{\theta}(\mathbf{x}) d\mathbf{x} \\&= \int_{\mathbf{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) d\mathbf{x} + \int_{\mathbf{x}} p_{\theta}^t(\mathbf{x}) \left( -\frac{\nabla_{\theta} p_{\theta}(\mathbf{x})}{p_{\theta}(\mathbf{x})} \right) d\mathbf{x}\end{aligned}$$

# Contrastive Divergence

## Second Gradient Vector in RHS

$$\nabla_{\theta} \text{KL}(p_{\theta}^t \| p_{\theta}) = \int_{\mathbf{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) d\mathbf{x} + \int_{\mathbf{x}} p_{\theta}^t(\mathbf{x}) \left( -\frac{\nabla_{\theta} p_{\theta}(\mathbf{x})}{p_{\theta}(\mathbf{x})} \right) d\mathbf{x}$$

# Contrastive Divergence

## Second Gradient Vector in RHS

$$\begin{aligned}\nabla_{\theta} \text{KL}(p_{\theta}^t \| p_{\theta}) &= \int_{\boldsymbol{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\boldsymbol{x}) d\boldsymbol{x} + \int_{\boldsymbol{x}} p_{\theta}^t(\boldsymbol{x}) \left( -\frac{\nabla_{\theta} p_{\theta}(\boldsymbol{x})}{p_{\theta}(\boldsymbol{x})} \right) d\boldsymbol{x} \\ &= \int_{\boldsymbol{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\boldsymbol{x}) d\boldsymbol{x} + \int_{\boldsymbol{x}} p_{\theta}^t(\boldsymbol{x}) (-\nabla_{\theta} \log p_{\theta}(\boldsymbol{x})) d\boldsymbol{x}\end{aligned}$$

# Contrastive Divergence

## Second Gradient Vector in RHS

$$\begin{aligned}\nabla_{\theta} \text{KL}(p_{\theta}^t \| p_{\theta}) &= \int_{\mathbf{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) d\mathbf{x} + \int_{\mathbf{x}} p_{\theta}^t(\mathbf{x}) \left( -\frac{\nabla_{\theta} p_{\theta}(\mathbf{x})}{p_{\theta}(\mathbf{x})} \right) d\mathbf{x} \\ &= \int_{\mathbf{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) d\mathbf{x} + \int_{\mathbf{x}} p_{\theta}^t(\mathbf{x}) (-\nabla_{\theta} \log p_{\theta}(\mathbf{x})) d\mathbf{x} \\ &= \int_{\mathbf{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) d\mathbf{x} + \int_{\mathbf{x}} p_{\theta}^t(\mathbf{x}) (\nabla_{\theta} E_{\theta}(\mathbf{x}) + \nabla_{\theta} \log Z_{\theta}) d\mathbf{x}\end{aligned}$$

# Contrastive Divergence

## Second Gradient Vector in RHS

$$\begin{aligned}\nabla_{\theta} \text{KL}(p_{\theta}^t \| p_{\theta}) &= \int_{\mathbf{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) d\mathbf{x} + \int_{\mathbf{x}} p_{\theta}^t(\mathbf{x}) \left( -\frac{\nabla_{\theta} p_{\theta}(\mathbf{x})}{p_{\theta}(\mathbf{x})} \right) d\mathbf{x} \\ &= \int_{\mathbf{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) d\mathbf{x} + \int_{\mathbf{x}} p_{\theta}^t(\mathbf{x}) (-\nabla_{\theta} \log p_{\theta}(\mathbf{x})) d\mathbf{x} \\ &= \int_{\mathbf{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) d\mathbf{x} + \int_{\mathbf{x}} p_{\theta}^t(\mathbf{x}) (\nabla_{\theta} E_{\theta}(\mathbf{x}) + \nabla_{\theta} \log Z_{\theta}) d\mathbf{x} \\ &= \int_{\mathbf{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) d\mathbf{x} + \mathbb{E}_{\mathbf{x} \sim p_{\theta}^t(\mathbb{X})} [\nabla_{\theta} E_{\theta}(\mathbf{x})] + \nabla_{\theta} \log Z_{\theta}\end{aligned}$$

# Contrastive Divergence

## Gradient Vector

Altogether, we have:

$$\begin{aligned}\nabla_{\theta} \text{CD}_t = & \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} [\nabla_{\theta} E_{\theta}(\mathbf{x})] + \nabla_{\theta} \log Z_{\theta} \\ & - \int_{\mathbf{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) d\mathbf{x} - \mathbb{E}_{\mathbf{x} \sim p_{\theta}^t(\mathbb{X})} [\nabla_{\theta} E_{\theta}(\mathbf{x})] - \nabla_{\theta} \log Z_{\theta}\end{aligned}$$

# Contrastive Divergence

## Gradient Vector

Altogether, we have:

$$\begin{aligned}\nabla_{\theta} \text{CD}_t = & \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} [\nabla_{\theta} E_{\theta}(\mathbf{x})] + \nabla_{\theta} \log Z_{\theta} \\ & - \int_{\mathbf{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) d\mathbf{x} - \mathbb{E}_{\mathbf{x} \sim p_{\theta}^t(\mathbb{X})} [\nabla_{\theta} E_{\theta}(\mathbf{x})] - \nabla_{\theta} \log Z_{\theta}\end{aligned}$$

Note that:

- You don't need to calculate  $\nabla_{\theta} \log Z_{\theta}$  as two terms in the RHL cancel each other.

# Contrastive Divergence

## Gradient Vector

Altogether, we have:

$$\begin{aligned}\nabla_{\theta} \text{CD}_t = & \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} [\nabla_{\theta} E_{\theta}(\mathbf{x})] + \nabla_{\theta} \log Z_{\theta} \\ & - \int_{\mathbf{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) d\mathbf{x} - \mathbb{E}_{\mathbf{x} \sim p_{\theta}^t(\mathbb{X})} [\nabla_{\theta} E_{\theta}(\mathbf{x})] - \nabla_{\theta} \log Z_{\theta}\end{aligned}$$

Note that:

- You don't need to calculate  $\nabla_{\theta} \log Z_{\theta}$  as two terms in the RHL cancel each other.
- In practice we can ignore  $\int_{\mathbf{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) d\mathbf{x}$  term [5].

# Contrastive Divergence

## Gradient Vector

Altogether, we have:

$$\begin{aligned}\nabla_{\theta} \text{CD}_t = & \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} [\nabla_{\theta} E_{\theta}(\mathbf{x})] + \nabla_{\theta} \log Z_{\theta} \\ & - \int_{\mathbf{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) d\mathbf{x} - \mathbb{E}_{\mathbf{x} \sim p_{\theta}^t(\mathbb{X})} [\nabla_{\theta} E_{\theta}(\mathbf{x})] - \nabla_{\theta} \log Z_{\theta}\end{aligned}$$

Note that:

- You don't need to calculate  $\nabla_{\theta} \log Z_{\theta}$  as two terms in the RHL cancel each other.
- In practice we can ignore  $\int_{\mathbf{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) d\mathbf{x}$  term [5].

Thus:

$$\nabla_{\theta} \text{CD}_t = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} [\nabla_{\theta} E_{\theta}(\mathbf{x})] - \mathbb{E}_{\mathbf{x} \sim p_{\theta}^t(\mathbb{X})} [\nabla_{\theta} E_{\theta}(\mathbf{x})]$$

# Contrastive Divergence

## Gradient Vector

Altogether, we have:

$$\begin{aligned}\nabla_{\theta} \text{CD}_t = & \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} [\nabla_{\theta} E_{\theta}(\mathbf{x})] + \nabla_{\theta} \log Z_{\theta} \\ & - \int_{\mathbf{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) d\mathbf{x} - \mathbb{E}_{\mathbf{x} \sim p_{\theta}^t(\mathbb{X})} [\nabla_{\theta} E_{\theta}(\mathbf{x})] - \nabla_{\theta} \log Z_{\theta}\end{aligned}$$

Note that:

- You don't need to calculate  $\nabla_{\theta} \log Z_{\theta}$  as two terms in the RHL cancel each other.
- In practice we can ignore  $\int_{\mathbf{x}} \frac{\partial B}{\partial p_{\theta}^t} \nabla_{\theta} p_{\theta}^t(\mathbf{x}) d\mathbf{x}$  term [5].

Thus:

$$\nabla_{\theta} \text{CD}_t = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} [\nabla_{\theta} E_{\theta}(\mathbf{x})] - \mathbb{E}_{\mathbf{x} \sim p_{\theta}^t(\mathbb{X})} [\nabla_{\theta} E_{\theta}(\mathbf{x})]$$

Note that sampling from both  $p_{\text{data}}(\mathbb{X})$  and  $p_{\theta}^t(\mathbb{X})$  is simple and fast.

# Contrastive Divergence

## Intuition

Assume we estimate  $\nabla_{\theta} \text{CD}_t$  for a large  $t$  using one samples as:

$$\nabla_{\theta} \text{CD}_t = \nabla_{\theta} E_{\theta}(\mathbf{x}_d) - \nabla_{\theta} E_{\theta}(\mathbf{x}_{\theta}), \begin{cases} \mathbf{x}_d \sim p_{\text{data}}(\mathbb{X}) \\ \mathbf{x}_{\theta} \sim p_{\theta}^t(\mathbb{X}) \end{cases}$$

# Contrastive Divergence

## Intuition

Assume we estimate  $\nabla_{\theta} \text{CD}_t$  for a large  $t$  using one samples as:

$$\nabla_{\theta} \text{CD}_t = \nabla_{\theta} E_{\theta}(\mathbf{x}_d) - \nabla_{\theta} E_{\theta}(\mathbf{x}_{\theta}), \begin{cases} \mathbf{x}_d \sim p_{\text{data}}(\mathbb{X}) \\ \mathbf{x}_{\theta} \sim p_{\theta}^t(\mathbb{X}) \end{cases}$$

Then:

# Contrastive Divergence

## Intuition

Assume we estimate  $\nabla_{\theta} \text{CD}_t$  for a large  $t$  using one samples as:

$$\nabla_{\theta} \text{CD}_t = \nabla_{\theta} E_{\theta}(\mathbf{x}_d) - \nabla_{\theta} E_{\theta}(\mathbf{x}_{\theta}), \begin{cases} \mathbf{x}_d \sim p_{\text{data}}(\mathbb{X}) \\ \mathbf{x}_{\theta} \sim p_{\theta}^t(\mathbb{X}) \end{cases}$$

Then:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \nabla_{\theta} \text{CD}_t$$

# Contrastive Divergence

## Intuition

Assume we estimate  $\nabla_{\theta} \text{CD}_t$  for a large  $t$  using one samples as:

$$\nabla_{\theta} \text{CD}_t = \nabla_{\theta} E_{\theta}(\mathbf{x}_d) - \nabla_{\theta} E_{\theta}(\mathbf{x}_{\theta}), \begin{cases} \mathbf{x}_d \sim p_{\text{data}}(\mathbb{X}) \\ \mathbf{x}_{\theta} \sim p_{\theta}^t(\mathbb{X}) \end{cases}$$

Then:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \nabla_{\theta} \text{CD}_t \Rightarrow \begin{cases} E_{\theta}(\mathbf{x}_d) \downarrow \downarrow \Rightarrow p_{\theta}(\mathbf{x}_d) \uparrow \uparrow \\ E_{\theta}(\mathbf{x}_{\theta}) \uparrow \uparrow \Rightarrow p_{\theta}(\mathbf{x}_{\theta}) \downarrow \downarrow \end{cases}$$

# Contrastive Divergence Visual Intuition

  $p_\theta(\mathbf{x})$   
 Data Samples

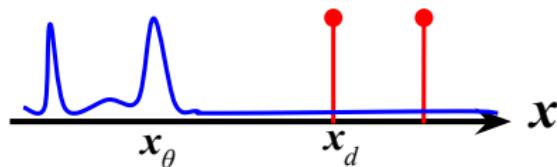


Figure: Before Update

# Contrastive Divergence Visual Intuition

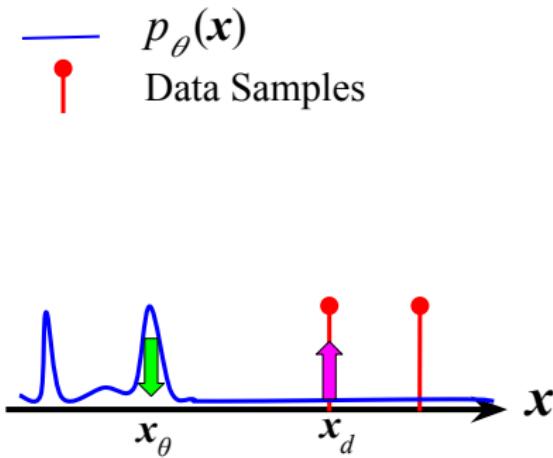


Figure: The way CD gradient affects model distribution

# Contrastive Divergence Visual Intuition

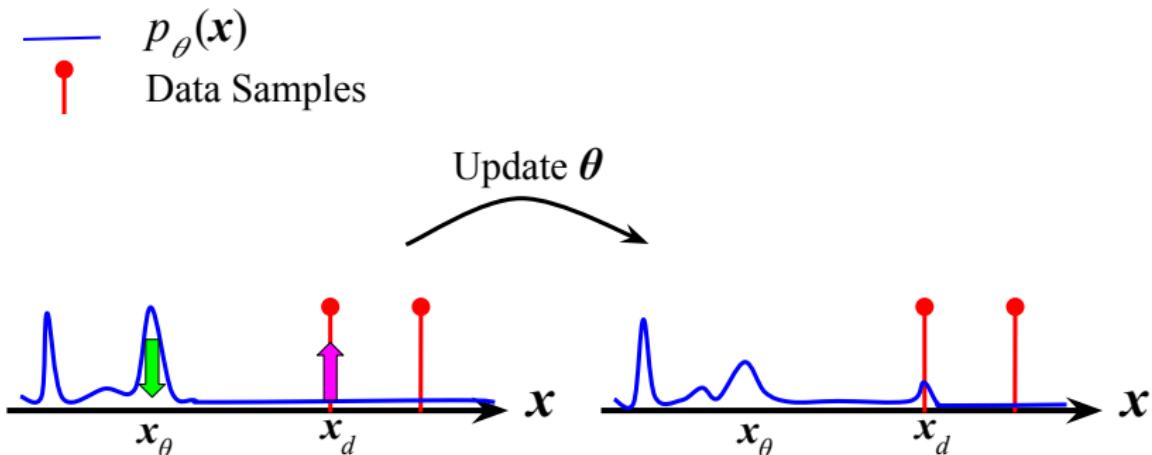
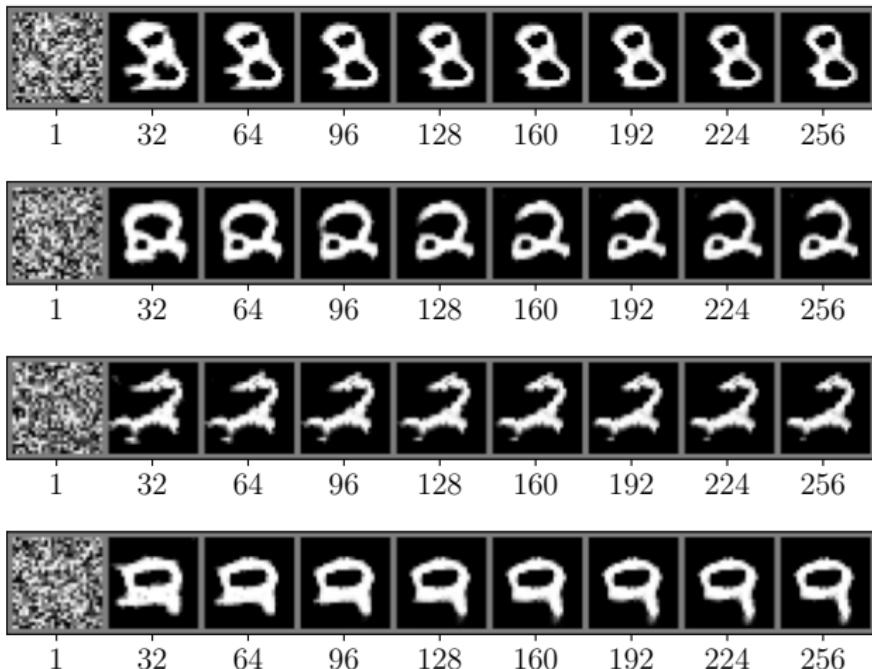


Figure: Updated model

# Contrastive Divergence



**Figure:** Samples from an EBM trained over MNIST dataset. The horizontal numbers show the iteration in the LMC generation process (source [6])

## Section 5

### Score Matching

# Score Function

## Training Challenges

Till now, we have learned two approaches for training generative models:

- Likelihood-based training (in Deep AE, VAE, Normalizing Flow)

# Score Function

## Training Challenges

Till now, we have learned two approaches for training generative models:

- Likelihood-based training (in Deep AE, VAE, Normalizing Flow)
- Likelihood-free training (in GAN)

# Score Function

## Training Challenges

Till now, we have learned two approaches for training generative models:

- Likelihood-based training (in Deep AE, VAE, Normalizing Flow)
- Likelihood-free training (in GAN)

But we can't train EBM:

- Using likelihood-based methods as we need to calculate partition functions which is intractable.

# Score Function

## Training Challenges

Till now, we have learned two approaches for training generative models:

- Likelihood-based training (in Deep AE, VAE, Normalizing Flow)
- Likelihood-free training (in GAN)

But we can't train EBM:

- Using likelihood-based methods as we need to calculate partition functions which is intractable.
- Using likelihood-free methods as we need to sample the model several times which is again intractable (in CD we bypass this challenge by sampling  $p_\theta^t(\mathbb{X})$ ).

# Score Function

## Training Challenges

Till now, we have learned two approaches for training generative models:

- Likelihood-based training (in Deep AE, VAE, Normalizing Flow)
- Likelihood-free training (in GAN)

But we can't train EBM:

- Using likelihood-based methods as we need to calculate partition functions which is intractable.
- Using likelihood-free methods as we need to sample the model several times which is again intractable (in CD we bypass this challenge by sampling  $p_\theta^t(\mathbb{X})$ ).

So we have seen CD as an alternative to the likelihood-based approach for training EBM. Now we want to explore another alternative known as *Score Matching*

# Score Function

## Intuition

For an EBM we have:

# Score Function

## Intuition

For an EBM we have:

$$\begin{cases} p(\boldsymbol{x}) = \frac{\exp(-E(\boldsymbol{x}))}{Z} \\ Z = \int_{\boldsymbol{x}} \exp(-E(\boldsymbol{x})) d\boldsymbol{x} \end{cases}$$

# Score Function

## Intuition

For an EBM we have:

$$\begin{cases} p(\boldsymbol{x}) = \frac{\exp(-E(\boldsymbol{x}))}{Z} \\ Z = \int_{\boldsymbol{x}} \exp(-E(\boldsymbol{x})) d\boldsymbol{x} \end{cases} \Rightarrow \mathbf{s}(\boldsymbol{x}) \triangleq \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) = -\nabla_{\boldsymbol{x}} E(\boldsymbol{x}) - \underbrace{\nabla_{\boldsymbol{x}} \log Z}_{=0}$$

# Score Function

## Intuition

For an EBM we have:

$$\begin{cases} p(\mathbf{x}) = \frac{\exp(-E(\mathbf{x}))}{Z} \\ Z = \int_{\mathbf{x}} \exp(-E(\mathbf{x})) d\mathbf{x} \end{cases} \Rightarrow \mathbf{s}(\mathbf{x}) \triangleq \nabla_{\mathbf{x}} \log p(\mathbf{x}) = -\nabla_{\mathbf{x}} E(\mathbf{x}) - \underbrace{\nabla_{\mathbf{x}} \log Z}_{=0}$$

So in  $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ , we can easily get rid of the partition function.

# Score Function

## Intuition

For an EBM we have:

$$\begin{cases} p(\mathbf{x}) = \frac{\exp(-E(\mathbf{x}))}{Z} \\ Z = \int_{\mathbf{x}} \exp(-E(\mathbf{x})) d\mathbf{x} \end{cases} \Rightarrow \mathbf{s}(\mathbf{x}) \triangleq \nabla_{\mathbf{x}} \log p(\mathbf{x}) = -\nabla_{\mathbf{x}} E(\mathbf{x}) - \underbrace{\nabla_{\mathbf{x}} \log Z}_{=0}$$

So in  $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ , we can easily get rid of the partition function.

- ☞  $\mathbf{s}(\mathbf{x}) = -\nabla_{\mathbf{x}} E(\mathbf{x})$  is known as *Score Function* and we have previously used in LMC algorithm.

# Score Function

## Score Function vs PDF

Score Function and PDF are alternative representations:

$$\begin{cases} \mathbf{s}(\mathbf{x}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}) \Rightarrow p(\mathbf{x}) = \exp \left\{ \int_{-\infty}^{\mathbf{x}} \mathbf{s}(\mathbf{x}') d\mathbf{x}' + K \right\} \\ \int_{-\infty}^{\infty} p(\mathbf{x}) d\mathbf{x} = 1 \end{cases} \Rightarrow \text{determine } K$$

# Score Function

## Score Function vs PDF

Score Function and PDF are alternative representations:

$$\begin{cases} \mathbf{s}(\mathbf{x}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}) \Rightarrow p(\mathbf{x}) = \exp \left\{ \int_{-\infty}^{\mathbf{x}} \mathbf{s}(\mathbf{x}') d\mathbf{x}' + K \right\} \\ \int_{-\infty}^{\infty} p(\mathbf{x}) d\mathbf{x} = 1 \end{cases} \Rightarrow \text{determine } K$$

While  $p(\mathbf{x})$  is a scalar,  $\mathbf{s}(\mathbf{x})$  is a vector with the same dimension as  $\mathbf{x}$ .

# Score Function

## Gaussian Distribution

For a Gaussian random variable  $x \sim \mathcal{N}(\mu, \sigma^2)$ , we have:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}$$

# Score Function

## Gaussian Distribution

For a Gaussian random variable  $x \sim \mathcal{N}(\mu, \sigma^2)$ , we have:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}$$

Then the score function is:

$$s(x) = \frac{d}{dx} \log p(x) = \frac{d}{dx} \left[ -\log \sqrt{2\pi\sigma^2} - \frac{(x-\mu)^2}{2\sigma^2} \right] = -\frac{x-\mu}{\sigma^2}$$

# Score Function

## Gaussian Distribution

For a Gaussian random variable  $x \sim \mathcal{N}(\mu, \sigma^2)$ , we have:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left\{ -\frac{(x-\mu)^2}{2\sigma^2} \right\}$$

Then the score function is:

$$s(x) = \frac{d}{dx} \log p(x) = \frac{d}{dx} \left[ -\log \sqrt{2\pi\sigma^2} - \frac{(x-\mu)^2}{2\sigma^2} \right] = -\frac{x-\mu}{\sigma^2}$$

We can also recover the distribution up to a constant value using the following formula:

$$p(x) = \exp \left\{ \int_{-\infty}^x s(x') dx' + K \right\}$$

# Score Function

## Gaussian Distribution

For a Gaussian random variable  $x \sim \mathcal{N}(\mu, \sigma^2)$ , we have:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}$$

Then the score function is:

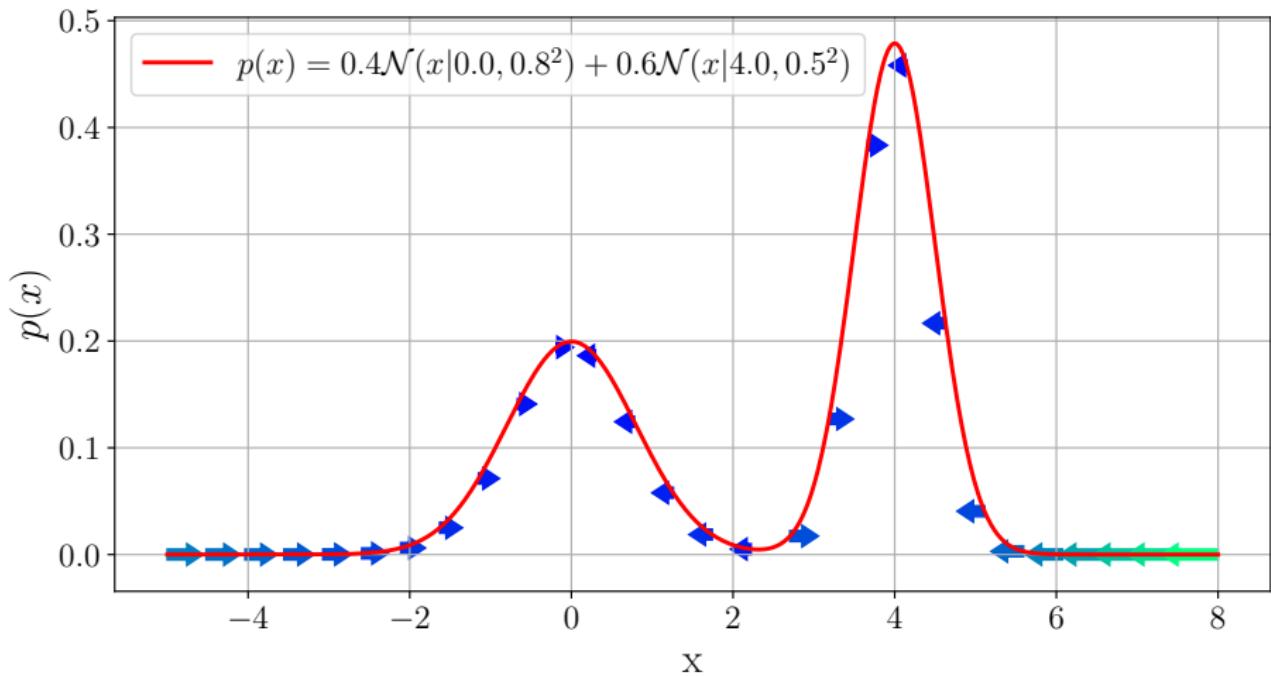
$$s(x) = \frac{d}{dx} \log p(x) = \frac{d}{dx} \left[ -\log \sqrt{2\pi\sigma^2} - \frac{(x-\mu)^2}{2\sigma^2} \right] = -\frac{x-\mu}{\sigma^2}$$

We can also recover the distribution up to a constant value using the following formula:

$$p(x) = \exp \left\{ \int_{-\infty}^x s(x') dx' + K \right\}$$

And  $K$  can also be identified easily using the unit volume property of PDF.

# Score Function for GMM



**Figure:** Score function for one-dimensional GMM. Each arrow represents the score function value at its start and the length and color of arrows are proportional to the score function value

# Score Matching

$$\mathbf{S} = \{s(\cdot)\}$$

$$s_{data}(x)$$


Figure: The general view to Score Matching

# Score Matching

$$\mathbf{S} = \{s(\cdot)\}$$

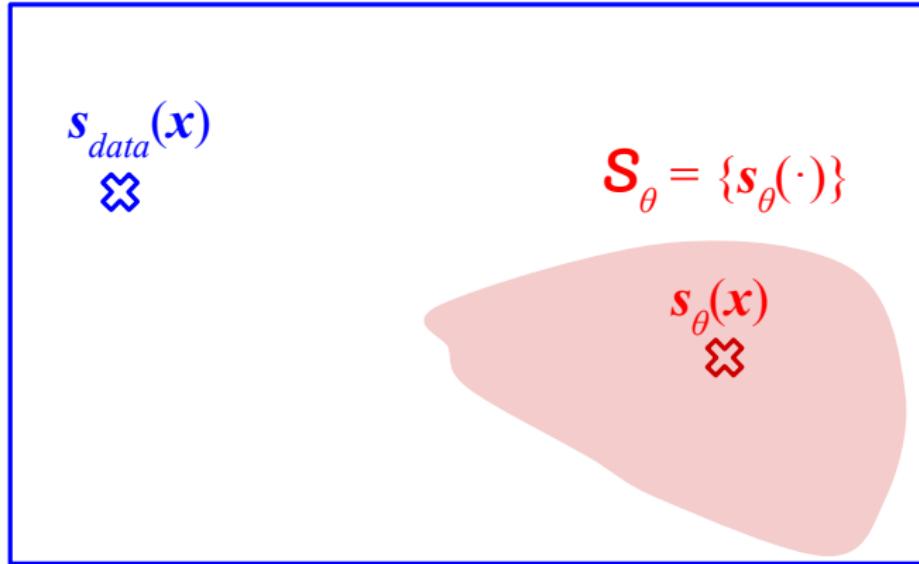


Figure: The general view to Score Matching

# Score Matching

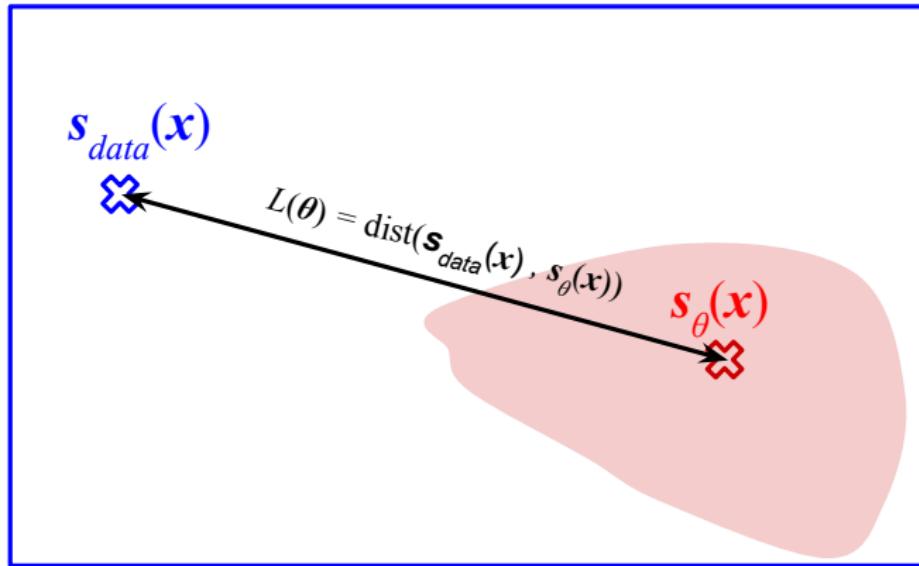
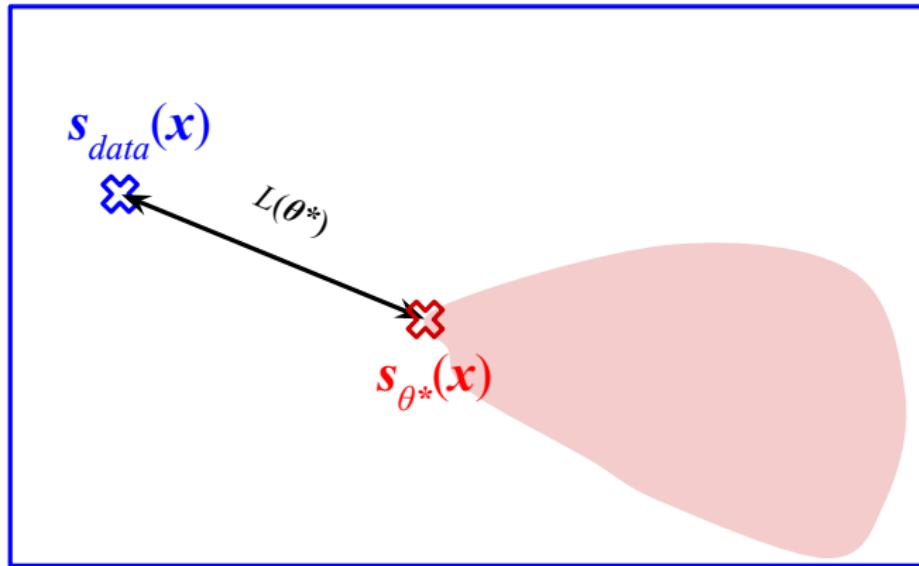


Figure: The general view to Score Matching

# Score Matching



$$\theta^* = \operatorname{argmin}_{\theta} L(\theta)$$

Figure: The general view to Score Matching

# Score Matching

## Challenegs

- Distance metric:
  - Previously we used KL divergence which compares two scalars.

# Score Matching

## Challenegs

- Distance metric:
  - Previously we used KL divergence which compares two scalars.
  - Now we should compare two vectors.

# Score Matching

## Challenegs

- Distance metric:
  - Previously we used KL divergence which compares two scalars.
  - Now we should compare two vectors.
- In theory,  $\mathbf{s}(\mathbf{x})$  and  $p(\mathbf{x})$  are equal, but computing  $p(\mathbf{x})$  from  $\mathbf{s}(\mathbf{x})$  needs a similar procedure as calculating partition function and thus intractable.

# Score Matching

## Challenegs

- Distance metric:
  - Previously we used KL divergence which compares two scalars.
  - Now we should compare two vectors.
- In theory,  $\mathbf{s}(\mathbf{x})$  and  $p(\mathbf{x})$  are equal, but computing  $p(\mathbf{x})$  from  $\mathbf{s}(\mathbf{x})$  needs a similar procedure as calculating partition function and thus intractable.
- Score Matching needs the calculation of  $\mathbf{s}_{\text{data}}(\mathbf{x}) \triangleq \nabla_{\mathbf{x}} p_{\text{data}}(\mathbf{x})$  which seems challenging as we just have access to samples from  $p_{\text{data}}(\mathbf{x})$

# Score Matching

## Challenegs

- Distance metric:
  - Previously we used KL divergence which compares two scalars.
  - Now we should compare two vectors.
- In theory,  $\mathbf{s}(\mathbf{x})$  and  $p(\mathbf{x})$  are equal, but computing  $p(\mathbf{x})$  from  $\mathbf{s}(\mathbf{x})$  needs a similar procedure as calculating partition function and thus intractable.
- Score Matching needs the calculation of  $\mathbf{s}_{\text{data}}(\mathbf{x}) \triangleq \nabla_{\mathbf{x}} p_{\text{data}}(\mathbf{x})$  which seems challenging as we just have access to samples from  $p_{\text{data}}(\mathbf{x})$
- The model to estimate  $\mathbf{s}_{\text{data}}(\mathbf{x})$  is a mapping with dimension  $\mathbb{R}^D \rightarrow \mathbb{R}^D$ , thus it is not as simple of an EBM anymore.

# Score Matching

## Challenegs

- Distance metric:
  - Previously we used KL divergence which compares two scalars.
  - Now we should compare two vectors.
- In theory,  $\mathbf{s}(\mathbf{x})$  and  $p(\mathbf{x})$  are equal, but computing  $p(\mathbf{x})$  from  $\mathbf{s}(\mathbf{x})$  needs a similar procedure as calculating partition function and thus intractable.
- Score Matching needs the calculation of  $\mathbf{s}_{\text{data}}(\mathbf{x}) \triangleq \nabla_{\mathbf{x}} p_{\text{data}}(\mathbf{x})$  which seems challenging as we just have access to samples from  $p_{\text{data}}(\mathbf{x})$
- The model to estimate  $\mathbf{s}_{\text{data}}(\mathbf{x})$  is a mapping with dimension  $\mathbb{R}^D \rightarrow \mathbb{R}^D$ , thus it is not as simple of an EBM anymore.
- If we train  $\mathbf{s}_\theta(\mathbf{x})$ , then sampling the resulting model needs MH or LMC (*Hard Sampling*).

# Score Matching

## Challenegs

- Distance metric:
  - Previously we used KL divergence which compares two scalars.
  - Now we should compare two vectors.
- In theory,  $\mathbf{s}(\mathbf{x})$  and  $p(\mathbf{x})$  are equal, but computing  $p(\mathbf{x})$  from  $\mathbf{s}(\mathbf{x})$  needs a similar procedure as calculating partition function and thus intractable.
- Score Matching needs the calculation of  $\mathbf{s}_{\text{data}}(\mathbf{x}) \triangleq \nabla_{\mathbf{x}} p_{\text{data}}(\mathbf{x})$  which seems challenging as we just have access to samples from  $p_{\text{data}}(\mathbf{x})$
- The model to estimate  $\mathbf{s}_{\text{data}}(\mathbf{x})$  is a mapping with dimension  $\mathbb{R}^D \rightarrow \mathbb{R}^D$ , thus it is not as simple of an EBM anymore.
- If we train  $\mathbf{s}_\theta(\mathbf{x})$ , then sampling the resulting model needs MH or LMC (*Hard Sampling*).
- Calculating the exact likelihood using trained  $\mathbf{s}_\theta(\mathbf{x})$  is intractable (*Hard Likelihood Calculation*).

# Score Matching

## Challenegs

- Distance metric:
  - Previously we used KL divergence which compares two scalars.
  - Now we should compare two vectors.
- In theory,  $\mathbf{s}(\mathbf{x})$  and  $p(\mathbf{x})$  are equal, but computing  $p(\mathbf{x})$  from  $\mathbf{s}(\mathbf{x})$  needs a similar procedure as calculating partition function and thus intractable.
- Score Matching needs the calculation of  $\mathbf{s}_{\text{data}}(\mathbf{x}) \triangleq \nabla_{\mathbf{x}} p_{\text{data}}(\mathbf{x})$  which seems challenging as we just have access to samples from  $p_{\text{data}}(\mathbf{x})$
- The model to estimate  $\mathbf{s}_{\text{data}}(\mathbf{x})$  is a mapping with dimension  $\mathbb{R}^D \rightarrow \mathbb{R}^D$ , thus it is not as simple of an EBM anymore.
- If we train  $\mathbf{s}_\theta(\mathbf{x})$ , then sampling the resulting model needs MH or LMC (*Hard Sampling*).
- Calculating the exact likelihood using trained  $\mathbf{s}_\theta(\mathbf{x})$  is intractable (*Hard Likelihood Calculation*).
- Our current formulation does not support hidden (latent) representation.

# Score Matching

## Explicit Score Matching

Hyvärinen and Dayan proposed to use the expected squared distance between the model score function  $\mathbf{s}_\theta(\mathbf{x})$  and the data score function  $\mathbf{s}_{\text{data}}(\mathbf{x})$ , also called *Fisher Dievergence* or *Explicit Score Matching* (ESM) loss, defined as [7]:

$$L_{\text{ESM}}(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} [\|\mathbf{s}_\theta(\mathbf{x}) - \mathbf{s}_{\text{data}}(\mathbf{x})\|_2^2]$$

# Score Matching

## Explicit Score Matching

Hyvärinen and Dayan proposed to use the expected squared distance between the model score function  $\mathbf{s}_\theta(\mathbf{x})$  and the data score function  $\mathbf{s}_{\text{data}}(\mathbf{x})$ , also called *Fisher Dievergence* or *Explicit Score Matching* (ESM) loss, defined as [7]:

$$L_{ESM}(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} [\|\mathbf{s}_\theta(\mathbf{x}) - \mathbf{s}_{\text{data}}(\mathbf{x})\|_2^2]$$

## Implicit Score Matching

Hyvärinen and Dayan define the *Implicit Score Matching* (ISM) loss as [7]:

$$L_{ISM}(\boldsymbol{\theta}) \triangleq \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \left[ \frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x})\|^2 + \text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x})) \right]$$

# Score Matching

## ESM and ISM are Equivalent

Hyvärinen and Dayan proved that  $L_{ESM}(\theta)$  and  $L_{ISM}(\theta)$  are equivalent optimization objectives with respect to  $\theta$

# Score Matching

## ESM and ISM are Equivalent

Hyvärinen and Dayan proved that  $L_{ESM}(\theta)$  and  $L_{ISM}(\theta)$  are equivalent optimization objectives with respect to  $\theta$  if  $s_\theta(x)$  is smooth and some weak regularity conditions are met [7].

# Score Matching

## ESM and ISM are Equivalent

Hyvärinen and Dayan proved that  $L_{ESM}(\boldsymbol{\theta})$  and  $L_{ISM}(\boldsymbol{\theta})$  are equivalent optimization objectives with respect to  $\boldsymbol{\theta}$  if  $s_\theta(\mathbf{x})$  is smooth and some weak regularity conditions are met [7]. In other words:

$$L_{ESM}(\boldsymbol{\theta}) \stackrel{\theta}{\equiv} L_{ISM}(\boldsymbol{\theta})$$

# Score Matching

## ESM and ISM are Equivalent

Hyvärinen and Dayan proved that  $L_{ESM}(\boldsymbol{\theta})$  and  $L_{ISM}(\boldsymbol{\theta})$  are equivalent optimization objectives with respect to  $\boldsymbol{\theta}$  if  $\mathbf{s}_\theta(\mathbf{x})$  is smooth and some weak regularity conditions are met [7]. In other words:

$$L_{ESM}(\boldsymbol{\theta}) \stackrel{\boldsymbol{\theta}}{=} L_{ISM}(\boldsymbol{\theta})$$

where:

$$L_{ESM}(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} [\|\mathbf{s}_\theta(\mathbf{x}) - \mathbf{s}_{\text{data}}(\mathbf{x})\|_2^2]$$

$$L_{ISM}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \left[ \frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x})\|_2^2 + \text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x})) \right]$$

# Score Matching

## Training

We see that the fisher divergence can be simplified as:

# Score Matching

## Training

We see that the fisher divergence can be simplified as:

$$L(\theta) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \left[ \frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x})\|_2^2 + \text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x})) \right]$$

# Score Matching

## Training

We see that the fisher divergence can be simplified as:

$$\begin{aligned} L(\theta) &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \left[ \frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x})\|_2^2 + \text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x})) \right] \\ &\stackrel{(a)}{\simeq} \frac{1}{N} \sum_{n=1}^N \left( \frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x}_n)\|_2^2 + \text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x}_n)) \right) \end{aligned}$$

# Score Matching

## Training

We see that the fisher divergence can be simplified as:

$$\begin{aligned} L(\theta) &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \left[ \frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x})\|_2^2 + \text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x})) \right] \\ &\stackrel{(a)}{\simeq} \frac{1}{N} \sum_{n=1}^N \left( \frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x}_n)\|_2^2 + \text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x}_n)) \right) \end{aligned}$$

where (a) results from Monte Carlo estimation.

# Score Matching

## Training

We see that the fisher divergence can be simplified as:

$$\begin{aligned} L(\theta) &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \left[ \frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x})\|_2^2 + \text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x})) \right] \\ &\stackrel{(a)}{\simeq} \frac{1}{N} \sum_{n=1}^N \left( \frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x}_n)\|_2^2 + \text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x}_n)) \right) \end{aligned}$$

where (a) results from Monte Carlo estimation.

- $\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x}_n)$  is Jacobian matrix with dimension  $D \times D$ . Assume we denote the  $(i, j)$ -th element in  $\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x}_n)$  by  $[\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x}_n)]_{(i,j)}$ , then:

# Score Matching

## Training

We see that the fisher divergence can be simplified as:

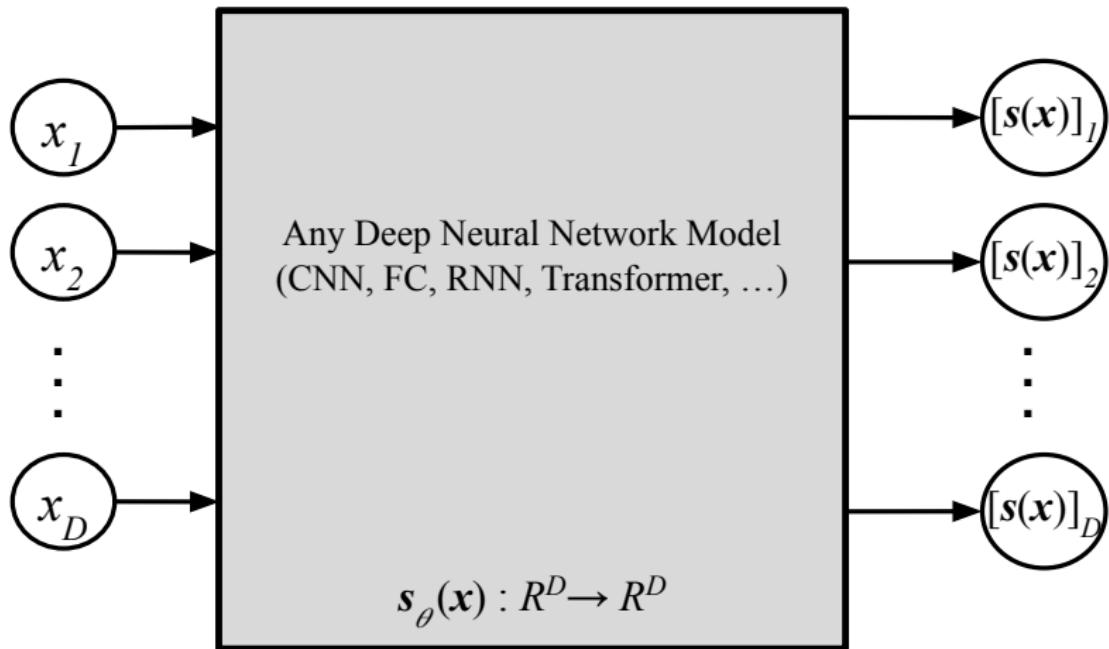
$$\begin{aligned} L(\theta) &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \left[ \frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x})\|_2^2 + \text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x})) \right] \\ &\stackrel{(a)}{\simeq} \frac{1}{N} \sum_{n=1}^N \left( \frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x}_n)\|_2^2 + \text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x}_n)) \right) \end{aligned}$$

where (a) results from Monte Carlo estimation.

- $\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x}_n)$  is Jacobian matrix with dimension  $D \times D$ . Assume we denote the  $(i, j)$ -th element in  $\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x}_n)$  by  $[\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x}_n)]_{(i,j)}$ , then:

$$L(\theta) = \frac{1}{N} \sum_{n=1}^N \left( \frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x}_n)\|_2^2 + \sum_{d=1}^D [\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x}_n)]_{d,d} \right)$$

# Model Score Function



**Figure:** The model score function used for training based on score matching (pay attention to the flexibility in selecting the model)

# Score Matching

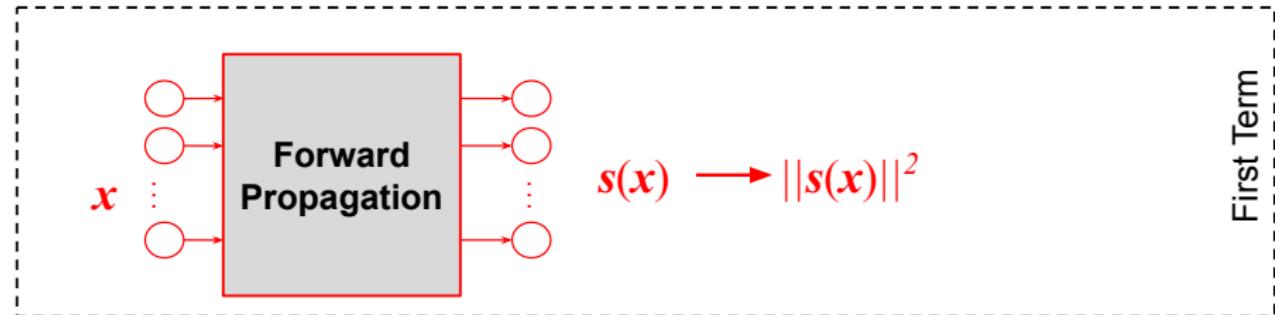


Figure: Calculating the first term

# Score Matching

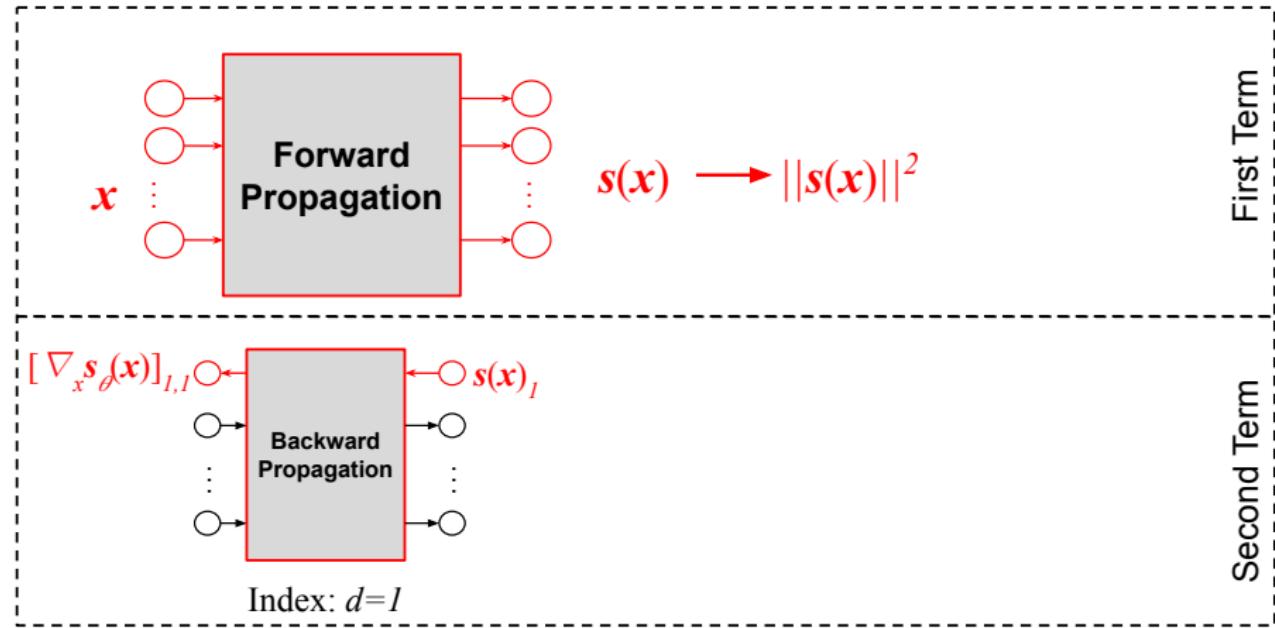


Figure: Calculating the second term index 1

# Score Matching

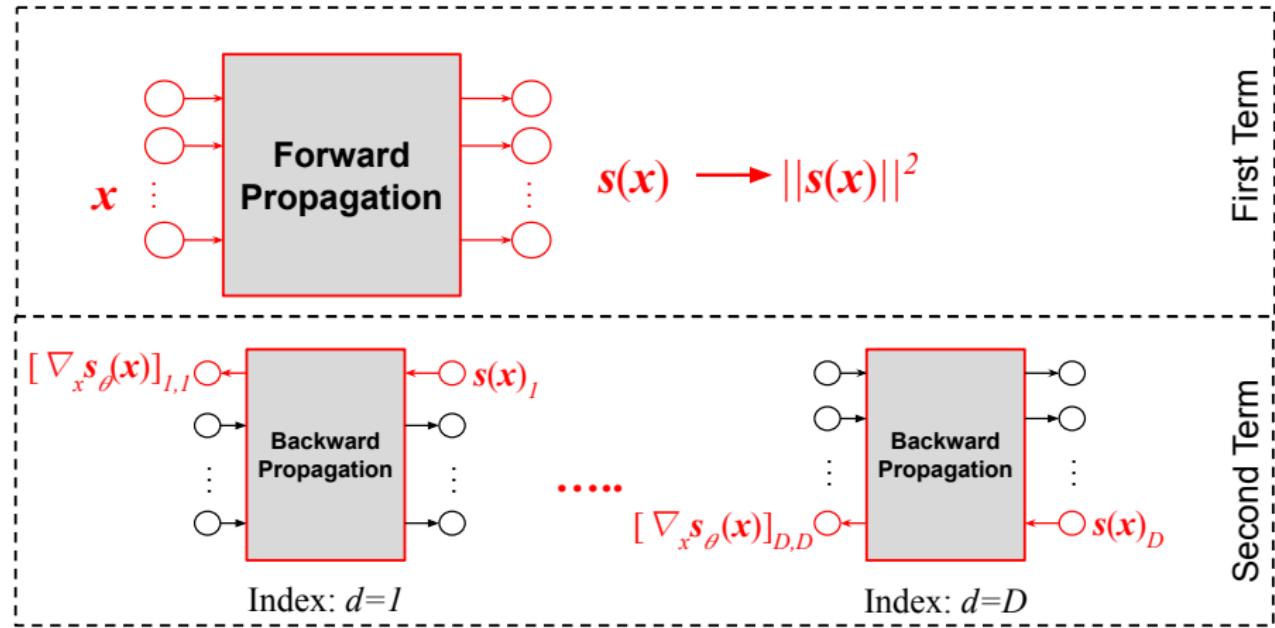


Figure: Calculating the second term index D

## Limitation

- The data score function  $s_{\text{data}}(\mathbf{x})$  is not presented in the score matching loss function. Thus one of our challenges is solved!

## Limitation

- The data score function  $\mathbf{s}_{\text{data}}(\mathbf{x})$  is not presented in the score matching loss function. Thus one of our challenges is solved!
- We have a new challenge! We need to compute the diagonal elements in the Jacobian matrix of  $\mathbf{s}_\theta(\mathbf{x})$ , which requires to call backpropagation for  $D$  times.

## Limitation

- The data score function  $\mathbf{s}_{\text{data}}(\mathbf{x})$  is not presented in the score matching loss function. Thus one of our challenges is solved!
- We have a new challenge! We need to compute the diagonal elements in the Jacobian matrix of  $\mathbf{s}_\theta(\mathbf{x})$ , which requires to call backpropagation for  $D$  times.
  - For example in the case of the ImageNet dataset with  $224 \times 224 \times 3$  dimension, you need to call 150528 backpropagation just to calculate the loss (note that we also need to calculate the gradient of the loss with respect to model parameters).

# Score Matching

## Limitation

- The data score function  $\mathbf{s}_{\text{data}}(\mathbf{x})$  is not presented in the score matching loss function. Thus one of our challenges is solved!
- We have a new challenge! We need to compute the diagonal elements in the Jacobian matrix of  $\mathbf{s}_\theta(\mathbf{x})$ , which requires to call backpropagation for  $D$  times.
  - For example in the case of the ImageNet dataset with  $224 \times 224 \times 3$  dimension, you need to call 150528 backpropagation just to calculate the loss (note that we also need to calculate the gradient of the loss with respect to model parameters).
- ☞ Score matching is not scalable with input dimension.

## Subsection 1

### Denoising Score Matching

# Noisy Dataset

## Definition

Assume  $\tilde{\mathbf{x}}$  to be the noisy version of  $\mathbf{x}$  as:

$$\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$$

# Noisy Dataset

## Definition

Assume  $\tilde{\mathbf{x}}$  to be the noisy version of  $\mathbf{x}$  as:

$$\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$$

Then the conditional distribution  $\tilde{\mathbf{x}}$  given  $\mathbf{x}$  (also known as *Smoothing kernel* or *Noise Model*), denoted by  $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})$ , is:

# Noisy Dataset

## Definition

Assume  $\tilde{\mathbf{x}}$  to be the noisy version of  $\mathbf{x}$  as:

$$\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$$

Then the conditional distribution  $\tilde{\mathbf{x}}$  given  $\mathbf{x}$  (also known as *Smoothing kernel* or *Noise Model*), denoted by  $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})$ , is:

$$q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2 \mathbf{I}) = \frac{1}{(2\pi)^{\frac{d}{2}} \sigma^d} \exp \left\{ -\frac{1}{2\sigma^2} \|\tilde{\mathbf{x}} - \mathbf{x}\|_2^2 \right\}$$

# Noisy Dataset

## Definition

Assume  $\tilde{\mathbf{x}}$  to be the noisy version of  $\mathbf{x}$  as:

$$\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$$

Then the conditional distribution  $\tilde{\mathbf{x}}$  given  $\mathbf{x}$  (also known as *Smoothing kernel* or *Noise Model*), denoted by  $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})$ , is:

$$q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2 \mathbf{I}) = \frac{1}{(2\pi)^{\frac{d}{2}} \sigma^d} \exp \left\{ -\frac{1}{2\sigma^2} \|\tilde{\mathbf{x}} - \mathbf{x}\|_2^2 \right\}$$

The marginal distribution of  $q_\sigma(\tilde{\mathbf{x}})$  can also be calculated using the law of total probability as:

$$q_\sigma(\tilde{\mathbf{x}}) = \int_{\mathbf{x}} q_\sigma(\mathbf{x}, \tilde{\mathbf{x}}) d\mathbf{x} = \int_{\mathbf{x}} q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) p_{\text{data}}(\mathbf{x}) d\mathbf{x}$$

# Noisy Dataset

## Intuition

Each sample of  $\tilde{\mathbf{x}}$  is the result of adding a Gaussian random variable to a sample of  $\mathbf{x}$  from the data distribution.

## Intuition

Each sample of  $\tilde{\mathbf{x}}$  is the result of adding a Gaussian random variable to a sample of  $\mathbf{x}$  from the data distribution. Thus:

- If  $\sigma \rightarrow 0$ , the added noise converts to zero constant and we expect the two distributions  $q_\sigma$  and  $p_{\text{data}}$  to be similar, equivalently:

$$\lim_{\sigma \rightarrow 0} q_\sigma(\tilde{\mathbf{x}}) = p_{\text{data}}(\mathbf{x})$$

# Noisy Dataset

## Mixture of Gaussian

Assume data distribution is a mixture of Gaussian as:

$$p_{\text{data}}(\mathbb{X}) = \sum_m \pi_m \mathcal{N}(\mathbb{X} | \mu_m, \sigma_m^2 \mathbf{I}), \quad \begin{cases} 0 \leq \pi_m \leq 1 \\ \sum_m \pi_m = 1 \end{cases}$$

and  $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ .

# Noisy Dataset

## Mixture of Gaussian

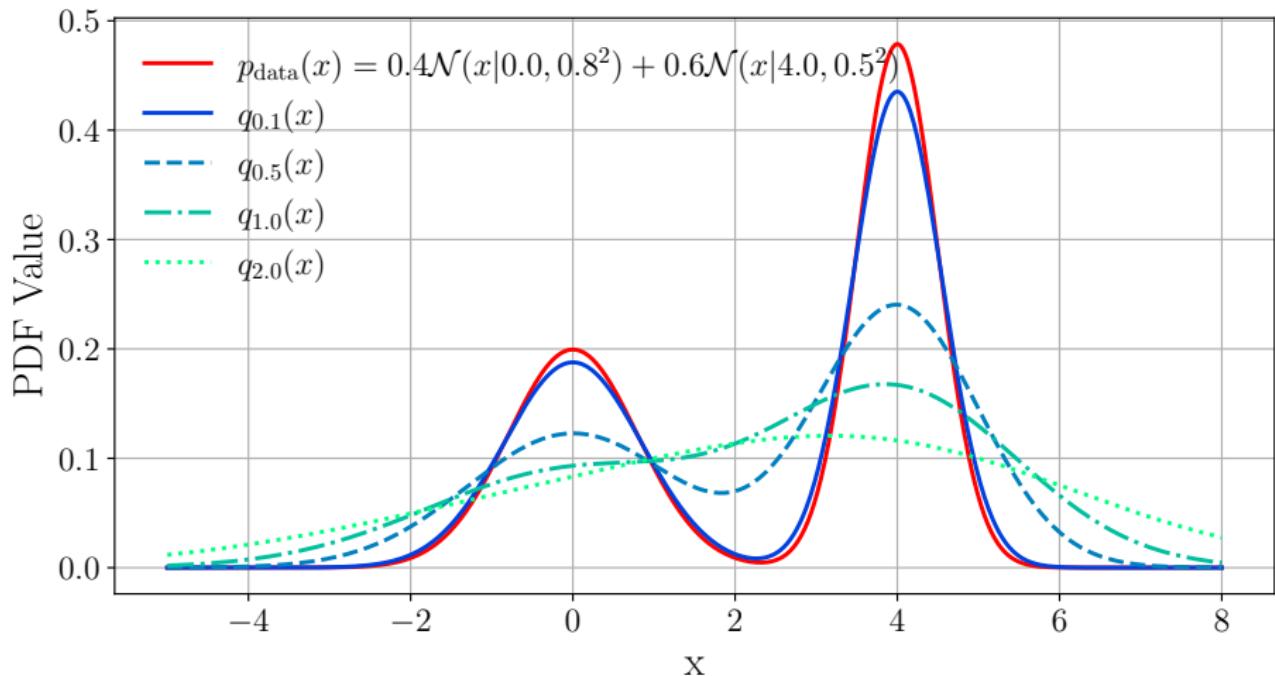
Assume data distribution is a mixture of Gaussian as:

$$p_{\text{data}}(\mathbb{X}) = \sum_m \pi_m \mathcal{N}(\mathbb{X} | \mu_m, \sigma_m^2 \mathbf{I}), \quad \begin{cases} 0 \leq \pi_m \leq 1 \\ \sum_m \pi_m = 1 \end{cases}$$

and  $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ . Then using the Bayes rule for Gaussians, we can show:

$$q_\sigma(\tilde{\mathbb{X}}) = \sum_m \pi_m \mathcal{N}(\mathbb{X} | \mu_m, (\sigma_m^2 + \sigma^2) \mathbf{I})$$

# Noisy Dataset



**Figure:** The marginal distribution  $q_\sigma(x)$  for different value of  $\sigma$  (as  $\sigma$  approaches zero, the similarity of  $p_{\text{data}}(x)$  and  $q_\sigma(x)$  increases)

# Denoising Score Matching

## ISM for Noisy Data

As we see before, when  $\sigma \rightarrow 0$ , the marginal distribution  $q_\sigma(\tilde{\mathbb{X}})$  approaches the data distribution  $p_{\text{data}}$ .

# Denoising Score Matching

## ISM for Noisy Data

As we see before, when  $\sigma \rightarrow 0$ , the marginal distribution  $q_\sigma(\tilde{\mathbb{X}})$  approaches the data distribution  $p_{\text{data}}$ .

- We can think of matching the model score function  $\mathbf{s}_\theta(\mathbf{x})$  to score function  $\mathbf{s}_\sigma(\mathbf{x})$  associated with  $q_\sigma$  distribution.

# Denoising Score Matching

## ISM for Noisy Data

As we see before, when  $\sigma \rightarrow 0$ , the marginal distribution  $q_\sigma(\tilde{\mathbb{X}})$  approaches the data distribution  $p_{\text{data}}$ .

- We can think of matching the model score function  $\mathbf{s}_\theta(\mathbf{x})$  to score function  $\mathbf{s}_\sigma(\mathbf{x})$  associated with  $q_\sigma$  distribution. The resulting ESM loss is:

$$L_{ESM,\sigma}(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbb{X}})} [\|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \mathbf{s}_\sigma(\tilde{\mathbf{x}})\|_2^2]$$

# Denoising Score Matching

## ISM for Noisy Data

As we see before, when  $\sigma \rightarrow 0$ , the marginal distribution  $q_\sigma(\tilde{\mathbb{X}})$  approaches the data distribution  $p_{\text{data}}$ .

- We can think of matching the model score function  $\mathbf{s}_\theta(\mathbf{x})$  to score function  $\mathbf{s}_\sigma(\mathbf{x})$  associated with  $q_\sigma$  distribution. The resulting ESM loss is:

$$L_{ESM,\sigma}(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbb{X}})} [\|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \mathbf{s}_\sigma(\tilde{\mathbf{x}})\|_2^2]$$

## Denoising Score Matching

Vincent defines *Denoising Score Matching* (DSM) loss as [8]:

$$L_{DSM,\sigma}(\boldsymbol{\theta}) \triangleq \mathbb{E}_{(\mathbf{x}, \tilde{\mathbf{x}}) \sim q_\sigma(\mathbb{X}, \tilde{\mathbb{X}})} \left[ \frac{1}{2} \|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} | \mathbf{x})\|_2^2 \right]$$

# Denoising Score Matching

ESM over Noisy Data and DSM are Equivalent

Vincent proved that  $L_{ESM,\sigma}(\theta)$  and  $L_{DSM,\sigma}(\theta)$  are equivalent optimization objectives with respect to  $\theta$ .

# Denoising Score Matching

ESM over Noisy Data and DSM are Equivalent

Vincent proved that  $L_{ESM,\sigma}(\theta)$  and  $L_{DSM,\sigma}(\theta)$  are equivalent optimization objectives with respect to  $\theta$ . In other words:

$$L_{ESM,\sigma}(\theta) \stackrel{\theta}{\equiv} L_{DSM,\sigma}(\theta)$$

where:

$$L_{ESM,\sigma}(\theta) = \frac{1}{2} \mathbb{E}_{\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbb{X}})} [\|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \mathbf{s}_\sigma(\tilde{\mathbf{x}})\|_2^2]$$

$$L_{DSM,\sigma}(\theta) = \mathbb{E}_{(\mathbf{x}, \tilde{\mathbf{x}}) \sim q_\sigma(\mathbb{X}, \tilde{\mathbb{X}})} \left[ \frac{1}{2} \|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} | \mathbf{x})\|^2 \right]$$

# Denoising Score Matching

## Revisiting DSM

Remember smoothing kernel:

# Denoising Score Matching

## Revisiting DSM

Remember smoothing kernel:

$$q_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}\sigma^d} \exp\left\{-\frac{1}{2\sigma^2}\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|^2\right\}$$

# Denoising Score Matching

## Revisiting DSM

Remember smoothing kernel:

$$q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}\sigma^d} \exp\left\{-\frac{1}{2\sigma^2}\|\tilde{\mathbf{x}} - \mathbf{x}\|^2\right\}$$
$$\Rightarrow \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})$$

# Denoising Score Matching

## Revisiting DSM

Remember smoothing kernel:

$$q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}\sigma^d} \exp\left\{-\frac{1}{2\sigma^2}\|\tilde{\mathbf{x}} - \mathbf{x}\|^2\right\}$$
$$\Rightarrow \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})$$

So we can revisit DSM loss as:

# Denoising Score Matching

## Revisiting DSM

Remember smoothing kernel:

$$q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}\sigma^d} \exp\left\{-\frac{1}{2\sigma^2}\|\tilde{\mathbf{x}} - \mathbf{x}\|^2\right\}$$
$$\Rightarrow \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})$$

So we can revisit DSM loss as:

$$L_{DSM,\sigma}(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{x}, \tilde{\mathbf{x}}) \sim q_\sigma(\mathbb{X}, \tilde{\mathbb{X}})} \left[ \frac{1}{2} \|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})\|^2 \right]$$

# Denoising Score Matching

## Revisiting DSM

Remember smoothing kernel:

$$q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}\sigma^d} \exp\left\{-\frac{1}{2\sigma^2}\|\tilde{\mathbf{x}} - \mathbf{x}\|^2\right\}$$
$$\Rightarrow \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})$$

So we can revisit DSM loss as:

$$L_{DSM,\sigma}(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{x}, \tilde{\mathbf{x}}) \sim q_\sigma(\mathbb{X}, \tilde{\mathbb{X}})} \left[ \frac{1}{2} \|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})\|^2 \right]$$
$$\stackrel{(a)}{=} \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \mathbb{E}_{\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbb{X}}|\mathbf{x})} \left[ \|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})\|^2 \right]$$

# Denoising Score Matching

## Revisiting DSM

Remember smoothing kernel:

$$q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}\sigma^d} \exp\left\{-\frac{1}{2\sigma^2}\|\tilde{\mathbf{x}} - \mathbf{x}\|^2\right\}$$
$$\Rightarrow \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})$$

So we can revisit DSM loss as:

$$L_{DSM,\sigma}(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{x}, \tilde{\mathbf{x}}) \sim q_\sigma(\mathbb{X}, \tilde{\mathbb{X}})} \left[ \frac{1}{2} \|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})\|^2 \right]$$
$$\stackrel{(a)}{=} \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \mathbb{E}_{\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbb{X}}|\mathbf{x})} \left[ \|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})\|^2 \right]$$

where in (a) we use the equation  $q_\sigma(\mathbf{x}, \tilde{\mathbf{x}}) = p_{\text{data}}(\mathbf{x})q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})$ .

# Denoising Score Matching

## Intuition

$$L_{DSM, \sigma}(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \mathbb{E}_{\tilde{\mathbf{x}} \sim q_{\sigma}(\tilde{\mathbb{X}} | \mathbf{x})} \left[ \|\mathbf{s}_{\theta}(\tilde{\mathbf{x}}) - \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})\|^2 \right]$$

Interestingly:

# Denoising Score Matching

## Intuition

$$L_{DSM,\sigma}(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \mathbb{E}_{\tilde{\mathbf{x}} \sim q_{\sigma}(\tilde{\mathbb{X}}|\mathbf{x})} \left[ \|\mathbf{s}_{\theta}(\tilde{\mathbf{x}}) - \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})\|^2 \right]$$

Interestingly:

- *No element from the Jacobian of score function:* The computational complexity is reduced.

# Denoising Score Matching

## Intuition

$$L_{DSM,\sigma}(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \mathbb{E}_{\tilde{\mathbf{x}} \sim q_{\sigma}(\tilde{\mathbb{X}}|\mathbf{x})} \left[ \|\mathbf{s}_{\theta}(\tilde{\mathbf{x}}) - \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})\|^2 \right]$$

Interestingly:

- *No element from the Jacobian of score function:* The computational complexity is reduced.
- *No score function related to noisy data:* We have no  $\mathbf{s}_{\sigma}(\cdot)$  in the new loss function.

# Denoising Score Matching

## Intuition

$$L_{DSM,\sigma}(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \mathbb{E}_{\tilde{\mathbf{x}} \sim q_{\sigma}(\tilde{\mathbb{X}}|\mathbf{x})} \left[ \|\mathbf{s}_{\theta}(\tilde{\mathbf{x}}) - \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})\|^2 \right]$$

Interestingly:

- *No element from the Jacobian of score function:* The computational complexity is reduced.
- *No score function related to noisy data:* We have no  $\mathbf{s}_{\sigma}(\cdot)$  in the new loss function.

# Denoising Score Matching

## Intuition

$$L_{DSM, \sigma}(\theta) = \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \mathbb{E}_{\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbb{X}} | \mathbf{x})} \left[ \| \mathbf{s}_\theta(\tilde{\mathbf{x}}) - \frac{1}{\sigma^2} (\mathbf{x} - \tilde{\mathbf{x}}) \|^2 \right]$$

Interestingly:

# Denoising Score Matching

## Intuition

$$L_{DSM, \sigma}(\theta) = \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \mathbb{E}_{\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbb{X}} | \mathbf{x})} \left[ \| \mathbf{s}_\theta(\tilde{\mathbf{x}}) - \frac{1}{\sigma^2} (\mathbf{x} - \tilde{\mathbf{x}}) \|^2 \right]$$

Interestingly:

- The ideal model score matching based on DSM is:  $\mathbf{s}_\theta(\tilde{\mathbf{x}}) \simeq \frac{1}{\sigma^2} (\mathbf{x} - \tilde{\mathbf{x}})$

# Denoising Score Matching

## Intuition

$$L_{DSM,\sigma}(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \mathbb{E}_{\tilde{\mathbf{x}} \sim q_{\sigma}(\tilde{\mathbb{X}}|\mathbf{x})} \left[ \|\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}) - \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})\|^2 \right]$$

Interestingly:

- The ideal model score matching based on DSM is:  $\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}) \simeq \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})$
- On the other hand we know:  $L_{ESM,\sigma}(\boldsymbol{\theta}) \stackrel{\boldsymbol{\theta}}{\equiv} L_{DSM,\sigma}(\boldsymbol{\theta})$

# Denoising Score Matching

## Intuition

$$L_{DSM,\sigma}(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \mathbb{E}_{\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbb{X}} | \mathbf{x})} \left[ \|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})\|^2 \right]$$

Interestingly:

- The ideal model score matching based on DSM is:  $\mathbf{s}_\theta(\tilde{\mathbf{x}}) \simeq \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})$
- On the other hand we know:  $L_{ESM,\sigma}(\boldsymbol{\theta}) \stackrel{\boldsymbol{\theta}}{\equiv} L_{DSM,\sigma}(\boldsymbol{\theta})$

Thus the ideal  $\mathbf{s}_\theta(\tilde{\mathbf{x}}) \simeq \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})$  is also the ideal model score function approximation to noisy data marginal distribution  $q_\sigma(\tilde{\mathbb{X}})$

# Denoising Score Matching

---

**Algorithm 24** Denoising Score Matching

---

1: **procedure** DSM( $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N, \sigma, q_\sigma(\cdot|\cdot)$ )

# Denoising Score Matching

---

**Algorithm 25** Denoising Score Matching

---

1: **procedure** DSM( $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N, \sigma, q_\sigma(\cdot|\cdot)$ )  
2:     **Initialization:**  $\theta$

# Denoising Score Matching

---

**Algorithm 26** Denoising Score Matching

---

```
1: procedure DSM( $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N, \sigma, q_\sigma(\cdot|\cdot)$ )
2:   Initialization:  $\theta$ 
3:   for  $k = 1 : K$  do
4:     Sample  $\mathbf{x} \sim p_{\text{data}}(\mathbb{X})$ 
```

---

# Denoising Score Matching

---

**Algorithm 27** Denoising Score Matching

---

```
1: procedure DSM( $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N, \sigma, q_\sigma(\cdot|\cdot)$ )
2:   Initialization:  $\theta$ 
3:   for  $k = 1 : K$  do
4:     Sample  $\mathbf{x} \sim p_{\text{data}}(\mathbb{X})$ 
5:     Sample  $\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbb{X}}|\mathbf{x})$ 
```

---

# Denoising Score Matching

---

**Algorithm 28** Denoising Score Matching

---

```
1: procedure DSM( $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N, \sigma, q_\sigma(\cdot|\cdot)$ )
2:   Initialization:  $\theta$ 
3:   for  $k = 1 : K$  do
4:     Sample  $\mathbf{x} \sim p_{\text{data}}(\mathbb{X})$ 
5:     Sample  $\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbb{X}}|\mathbf{x})$ 
6:     Update model score function as:
```

$$\theta \leftarrow \theta - \mu_k \nabla_\theta \left[ \| \mathbf{s}_\theta(\tilde{\mathbf{x}}) - \frac{1}{\sigma^2} (\mathbf{x} - \tilde{\mathbf{x}}) \|^2 \right]$$

# Denoising Score Matching

---

**Algorithm 29** Denoising Score Matching

---

```
1: procedure DSM( $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N, \sigma, q_\sigma(\cdot|\cdot)$ )
2:   Initialization:  $\theta$ 
3:   for  $k = 1 : K$  do
4:     Sample  $\mathbf{x} \sim p_{\text{data}}(\mathbb{X})$ 
5:     Sample  $\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbb{X}}|\mathbf{x})$ 
6:     Update model score function as:
```

$$\theta \leftarrow \theta - \mu_k \nabla_\theta \left[ \|s_\theta(\tilde{\mathbf{x}}) - \frac{1}{\sigma^2} (\mathbf{x} - \tilde{\mathbf{x}})\|^2 \right]$$

```
7:   end for
8: end procedure
```

---

# Denoising Score Matching

## Challenge

DSM has the following main challenges:

- In DSM, we are estimating the score function of the noisy dataset  $\mathbf{s}_\sigma(\tilde{\mathbf{x}})$ , which is not exactly the score function of data distribution  $\mathbf{s}_{\text{data}}(\mathbf{x})$ .

# Denoising Score Matching

## Challenge

DSM has the following main challenges:

- In DSM, we are estimating the score function of the noisy dataset  $\mathbf{s}_\sigma(\tilde{\mathbf{x}})$ , which is not exactly the score function of data distribution  $\mathbf{s}_{\text{data}}(\mathbf{x})$ .
- Selecting  $\sigma$  is tricky in DSM as:
  - Small  $\sigma$ : Although lead to better approximation of  $\mathbf{s}_{\text{data}}(\mathbf{x})$  using  $\mathbf{s}_\sigma(\tilde{\mathbf{x}})$ , but it leads to numerical problems as we have  $\frac{1}{\sigma^2}$  in the objective function.

# Denoising Score Matching

## Challenge

DSM has the following main challenges:

- In DSM, we are estimating the score function of the noisy dataset  $\mathbf{s}_\sigma(\tilde{\mathbf{x}})$ , which is not exactly the score function of data distribution  $\mathbf{s}_{\text{data}}(\mathbf{x})$ .
- Selecting  $\sigma$  is tricky in DSM as:
  - Small  $\sigma$ : Although lead to better approximation of  $\mathbf{s}_{\text{data}}(\mathbf{x})$  using  $\mathbf{s}_\sigma(\tilde{\mathbf{x}})$ , but it lead to numerical problems as we have  $\frac{1}{\sigma^2}$  in the objective function.
  - Large  $\sigma$ : This option leads to high discrepancy between data and noisy data distributions (and equivalently score functions)

## Subsection 2

### Sliced Score Matching

# Sliced Score Matching

## Intuition

Remember the ISM loss:

$$L_{ISM}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \left[ \frac{1}{2} \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})\|^2 + \text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})) \right]$$

# Sliced Score Matching

## Intuition

Remember the ISM loss:

$$L_{ISM}(\theta) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \left[ \frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x})\|^2 + \text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x})) \right]$$

The main problem was  $\text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x}))$  part which needs  $D$  backpropagation to calculate the loss.

# Sliced Score Matching

## Intuition

Remember the ISM loss:

$$L_{ISM}(\theta) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \left[ \frac{1}{2} \|\mathbf{s}_\theta(\mathbf{x})\|^2 + \text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x})) \right]$$

The main problem was  $\text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_\theta(\mathbf{x}))$  part which needs  $D$  backpropagation to calculate the loss.

- ☞ This difficulty arises because we are comparing two vectors of dimension  $D$ . What if we just compare  $\mathbf{s}_\theta(\mathbf{x})$  and  $\mathbf{s}_{\text{data}}(\mathbf{x})$  along one direction (In this case, we are comparing two scalars)?

# Sliced Score Matching

## Explicit Sliced Score Matching

Song *et al.* proposed to use the expected Fisher divergence along a random direction  $\mathbf{v}$  as the *Sliced Score Matching* (SSM) loss, defined as [9]:

$$L_{ESSM}(\theta) = \frac{1}{2} \mathbb{E}_{\mathbf{v} \sim p_v(\mathbb{V})} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \left[ \left( \mathbf{v}^T \mathbf{s}_\theta(\mathbf{x}) - \mathbf{v}^T \mathbf{s}_{\text{data}}(\mathbf{x}) \right)^2 \right]$$

where  $\mathbb{V}$  and  $\mathbb{X}$  are independent random variables.

# Sliced Score Matching

## Explicit Sliced Score Matching

Song *et al.* proposed to use the expected Fisher divergence along a random direction  $\mathbf{v}$  as the *Sliced Score Matching* (SSM) loss, defined as [9]:

$$L_{ESSM}(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{\mathbf{v} \sim p_v(\mathbb{V})} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \left[ \left( \mathbf{v}^T \mathbf{s}_\theta(\mathbf{x}) - \mathbf{v}^T \mathbf{s}_{\text{data}}(\mathbf{x}) \right)^2 \right]$$

where  $\mathbb{V}$  and  $\mathbb{X}$  are independent random variables.

## Implicit Sliced Score Matching

Song *et al.* define the *Implicit Sliced Score Matching* (ISSM) loss as [9]:

$$L_{ISSM}(\boldsymbol{\theta}) \triangleq \mathbb{E}_{\mathbf{v} \sim p_v(\mathbb{V})} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \left[ \frac{1}{2} \left( \mathbf{v}^T \mathbf{s}_\theta(\mathbf{x}) \right)^2 + \mathbf{v}^T (\nabla_x \mathbf{s}_\theta(\mathbf{x})) \mathbf{v} \right]$$

# Sliced Score Matching

## ESSM and ISSM are Equivalent

Song *et al.* proved that  $L_{ESSM}(\theta)$  and  $L_{ISSM}(\theta)$  are equivalent optimization objectives with respect to  $\theta$  if some weak regularity conditions are met [9].

# Sliced Score Matching

## ESSM and ISSM are Equivalent

Song *et al.* proved that  $L_{ESSM}(\theta)$  and  $L_{ISSM}(\theta)$  are equivalent optimization objectives with respect to  $\theta$  if some weak regularity conditions are met [9]. In other words:

$$L_{ESSM}(\theta) \stackrel{\theta}{\equiv} L_{ISSM}(\theta)$$

# Sliced Score Matching

## ESSM and ISSM are Equivalent

Song *et al.* proved that  $L_{ESSM}(\theta)$  and  $L_{ISSM}(\theta)$  are equivalent optimization objectives with respect to  $\theta$  if some weak regularity conditions are met [9]. In other words:

$$L_{ESSM}(\theta) \stackrel{\theta}{\equiv} L_{ESSM}(\theta)$$

where:

$$L_{ESSM}(\theta) = \frac{1}{2} \mathbb{E}_{v \sim p_v(\mathbb{V})} \mathbb{E}_{x \sim p_{\text{data}}(\mathbb{X})} \left[ \left( v^T \mathbf{s}_\theta(x) - v^T \mathbf{s}_{\text{data}}(x) \right)^2 \right]$$

$$L_{ISSM}(\theta) = \mathbb{E}_{v \sim p_v(\mathbb{V})} \mathbb{E}_{x \sim p_{\text{data}}(\mathbb{X})} \left[ \frac{1}{2} \left( v^T \mathbf{s}_\theta(x) \right)^2 + v^T (\nabla_x \mathbf{s}_\theta(x)) v \right]$$

# Sliced Score Matching

## Implicit Objective

The ISSM objective is:

$$L_{ISSM}(\theta) = \mathbb{E}_{v \sim p_v(\mathbb{V})} \mathbb{E}_{x \sim p_{\text{data}}(\mathbb{X})} \left[ \frac{1}{2} \left( v^T \mathbf{s}_\theta(x) \right)^2 + v^T (\nabla_x \mathbf{s}_\theta(x)) v \right]$$

- The first term can be calculated easily by the inner product of score function output vector  $\mathbf{s}_\theta(x)$  and sampled direction vector  $v$ .

# Sliced Score Matching

## Implicit Objective

The ISSM objective is:

$$L_{ISSM}(\theta) = \mathbb{E}_{v \sim p_v(\mathbb{V})} \mathbb{E}_{x \sim p_{\text{data}}(\mathbb{X})} \left[ \frac{1}{2} \left( v^T \mathbf{s}_\theta(x) \right)^2 + v^T (\nabla_x \mathbf{s}_\theta(x)) v \right]$$

- The first term can be calculated easily by the inner product of score function output vector  $\mathbf{s}_\theta(x)$  and sampled direction vector  $v$ .
- The second term needs a simple reformulation based on gradient properties:

$$v^T \nabla_x \mathbf{s}_\theta(x) v = v^T \nabla_x (v^T \mathbf{s}_\theta(x))$$

now we will show how this reformulation simplifies the calculations.

# SSM Objective Calculations

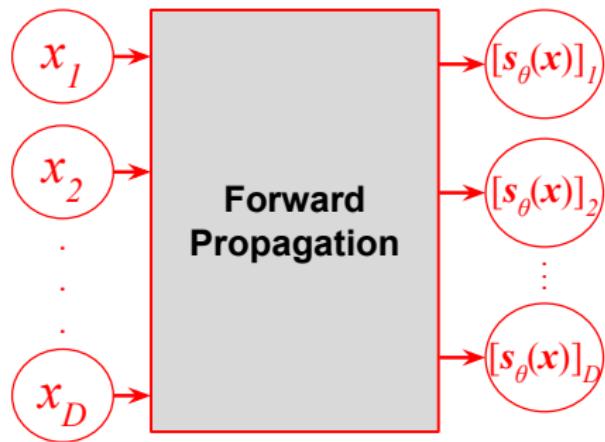


Figure: Forward propagating the model

# SSM Objective Calculations

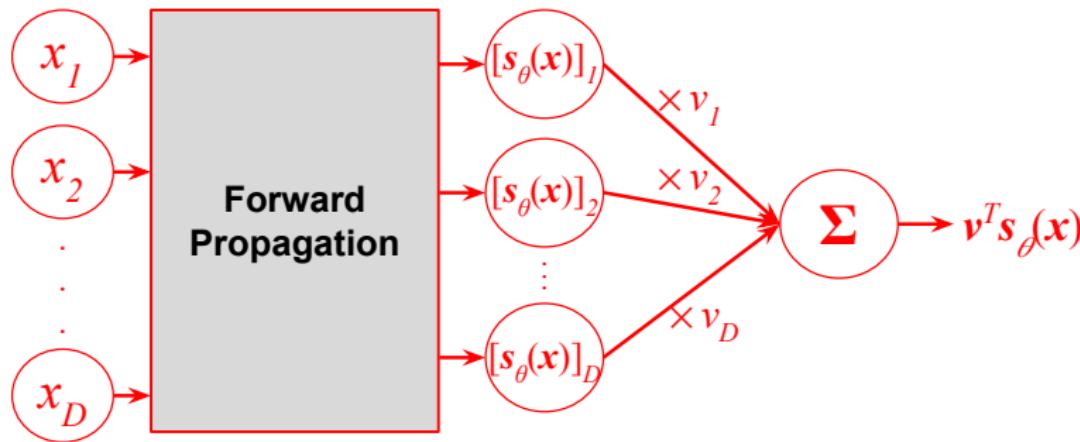


Figure: Calculating the inner product between score function vector  $s_\theta(\mathbf{x})$  and  $\mathbf{v}$

# SSM Objective Calculations

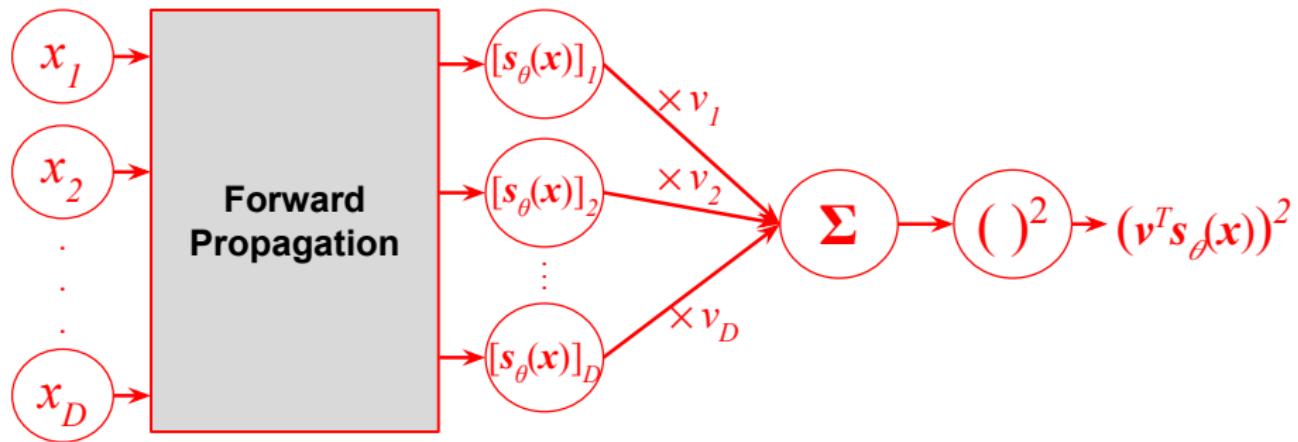
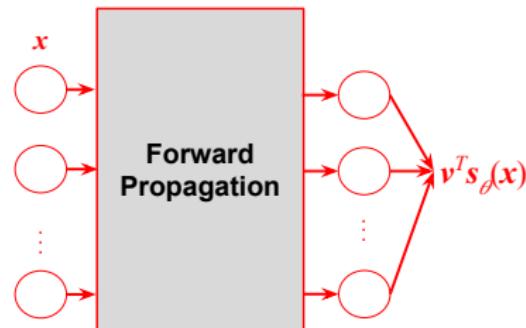


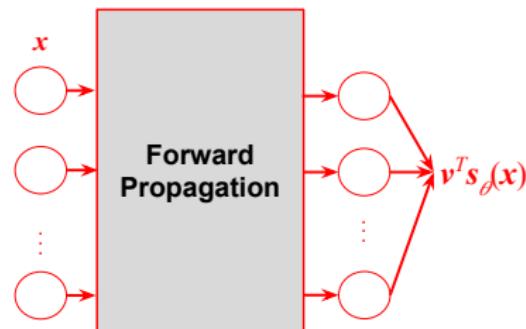
Figure: Calculating the second term index  $D$

# SSM Objective Calculations

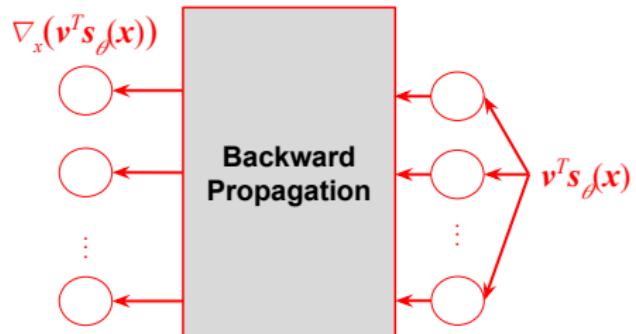


(a) Step 1: Calculating  $v^T \mathbf{s}_\theta(\mathbf{x})$

# SSM Objective Calculations

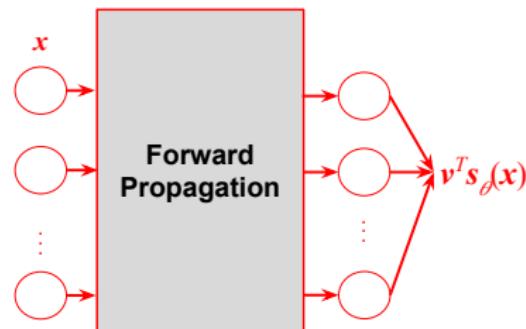


(a) Step 1: Calculating  $v^T s_\theta(\mathbf{x})$

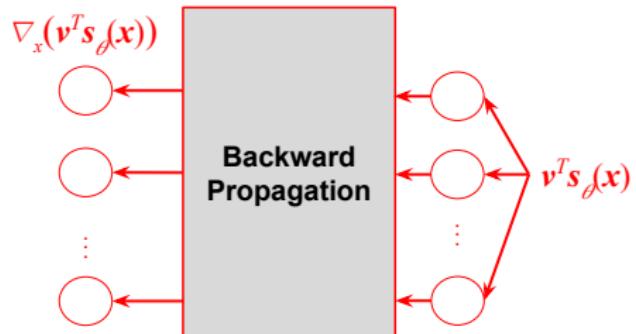


(b) Step 2: Calculating  $\nabla_x v^T s_\theta(\mathbf{x})$

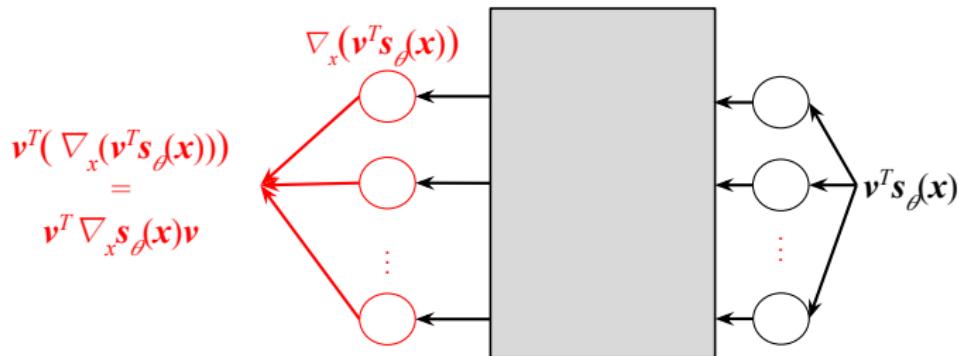
# SSM Objective Calculations



(a) Step 1: Calculating  $v^T s_\theta(x)$



(b) Step 2: Calculating  $\nabla_x v^T s_\theta(x)$



(c) Step 3: Calculating second term in SSM Objective  $v^T (\nabla_x s_\theta(x)) v$

# Sliced Score Matching

---

**Algorithm 30** Sliced Score Matching

---

1: **procedure** SSM( $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N, p_v(\mathbb{V})$ )

# Sliced Score Matching

---

**Algorithm 31** Sliced Score Matching

---

```
1: procedure SSM( $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N, p_v(\mathbb{V})$ )
2:   Initialization:  $\theta$ 
```

# Sliced Score Matching

---

**Algorithm 32** Sliced Score Matching

---

```
1: procedure SSM( $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N, p_v(\mathbb{V})$ )
2:   Initialization:  $\theta$ 
3:   for  $k = 1 : K$  do
4:     Sample  $\mathbf{x} \sim p_{\text{data}}(\mathbb{X})$ 
```

---

# Sliced Score Matching

---

**Algorithm 33** Sliced Score Matching

---

```
1: procedure SSM( $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N, p_v(\mathbb{V})$ )
2:   Initialization:  $\theta$ 
3:   for  $k = 1 : K$  do
4:     Sample  $\mathbf{x} \sim p_{\text{data}}(\mathbb{X})$ 
5:     Sample  $\mathbf{v} \sim p_v(\mathbb{V})$ 
```

# Sliced Score Matching

---

**Algorithm 34** Sliced Score Matching

---

```
1: procedure SSM( $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N, p_v(\mathbb{V})$ )
2:   Initialization:  $\theta$ 
3:   for  $k = 1 : K$  do
4:     Sample  $\mathbf{x} \sim p_{\text{data}}(\mathbb{X})$ 
5:     Sample  $\mathbf{v} \sim p_v(\mathbb{V})$ 
6:     Update model score function as:
```

$$\theta \leftarrow \theta - \mu_k \nabla_\theta \left[ \frac{1}{2} \left( \mathbf{v}^T \mathbf{s}_\theta(\mathbf{x}) \right)^2 + \mathbf{v}^T (\nabla_x \mathbf{s}_\theta(\mathbf{x})) \mathbf{v} \right]$$

# Sliced Score Matching

---

**Algorithm 35** Sliced Score Matching

---

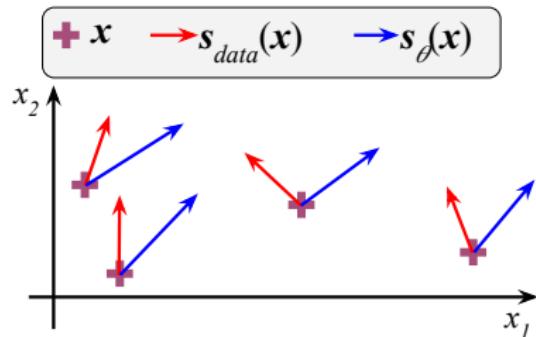
```
1: procedure SSM( $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N, p_v(\mathbb{V})$ )
2:   Initialization:  $\theta$ 
3:   for  $k = 1 : K$  do
4:     Sample  $\mathbf{x} \sim p_{\text{data}}(\mathbb{X})$ 
5:     Sample  $\mathbf{v} \sim p_v(\mathbb{V})$ 
6:     Update model score function as:
```

$$\theta \leftarrow \theta - \mu_k \nabla_\theta \left[ \frac{1}{2} \left( \mathbf{v}^T \mathbf{s}_\theta(\mathbf{x}) \right)^2 + \mathbf{v}^T (\nabla_x \mathbf{s}_\theta(\mathbf{x})) \mathbf{v} \right]$$

```
7:   end for
8: end procedure
```

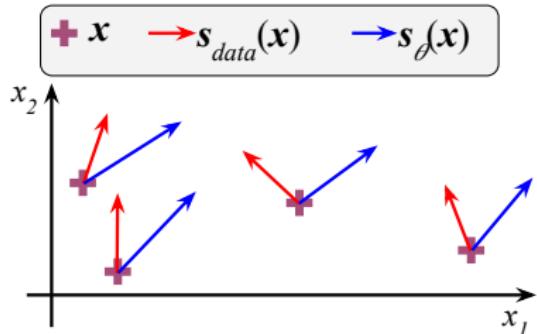
---

# SSM Visualization

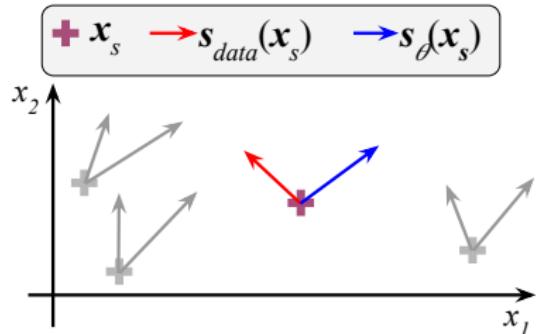


(a) Samples and score functions

# SSM Visualization

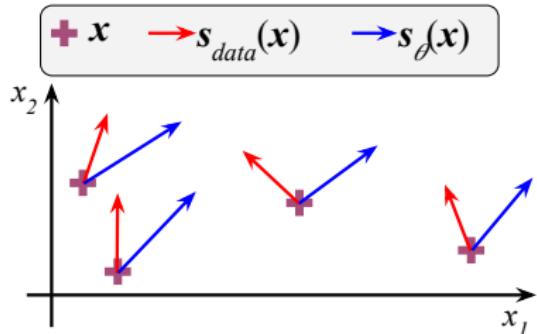


(a) Samples and score functions

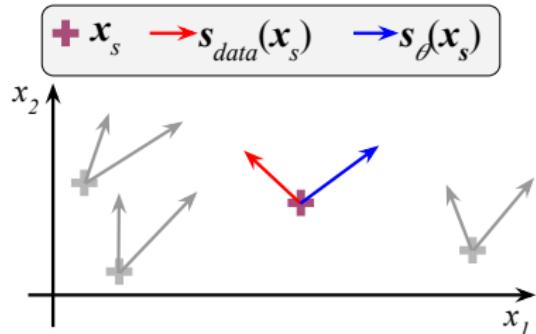


(b) Selecting a sample randomly

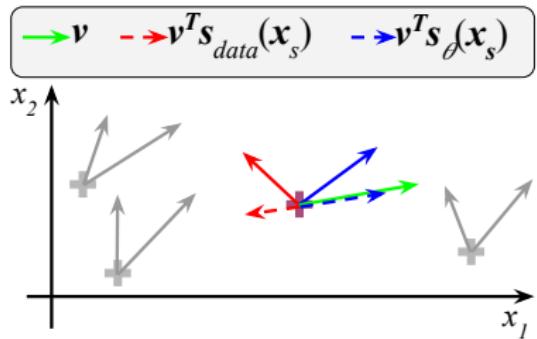
# SSM Visualization



(a) Samples and score functions

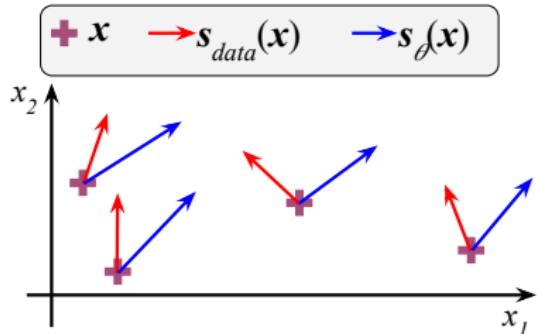


(b) Selecting a sample randomly

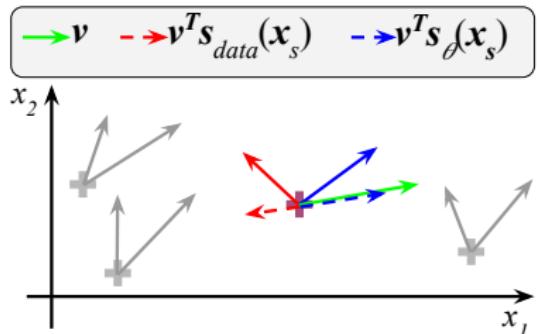


(c) Selecting a direction randomly

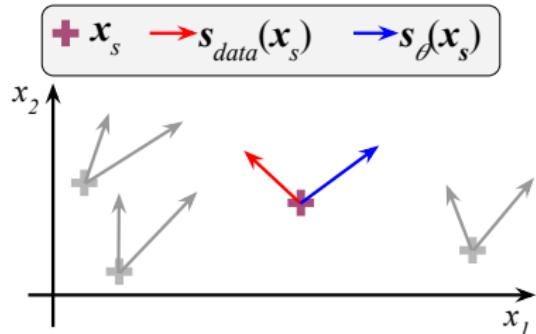
# SSM Visualization



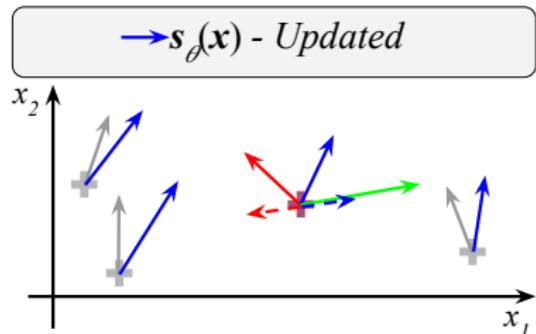
(a) Samples and score functions



(c) Selecting a direction randomly



(b) Selecting a sample randomly



(d) Update ISSM loss

## Section 6

### Noise Conditional Score Networks

# Naïve Score Matching

## Challenegs

Song and Ermon showed three major obstacles preventing a naïve application of Score Matching [10].

- Manifold hypothesis

# Naïve Score Matching

## Challenges

Song and Ermon showed three major obstacles preventing a naïve application of Score Matching [10].

- Manifold hypothesis
- Inaccurate score estimation with score matching

# Naïve Score Matching

## Challenges

Song and Ermon showed three major obstacles preventing a naïve application of Score Matching [10].

- Manifold hypothesis
- Inaccurate score estimation with score matching
- Slow mixing of Langevin dynamics

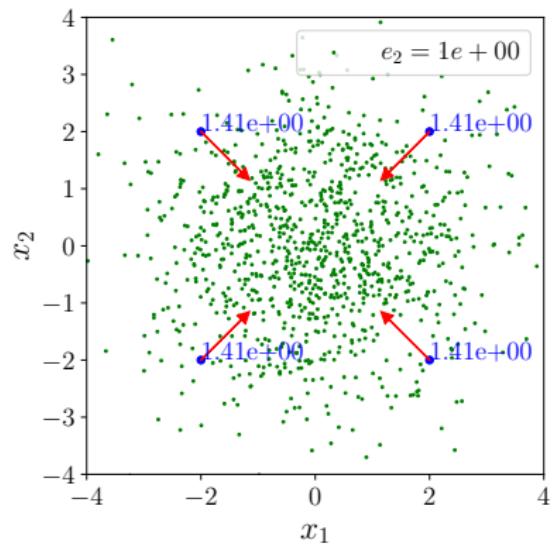
# Naïve Score Matching

## Challenges

Song and Ermon showed three major obstacles preventing a naïve application of Score Matching [10].

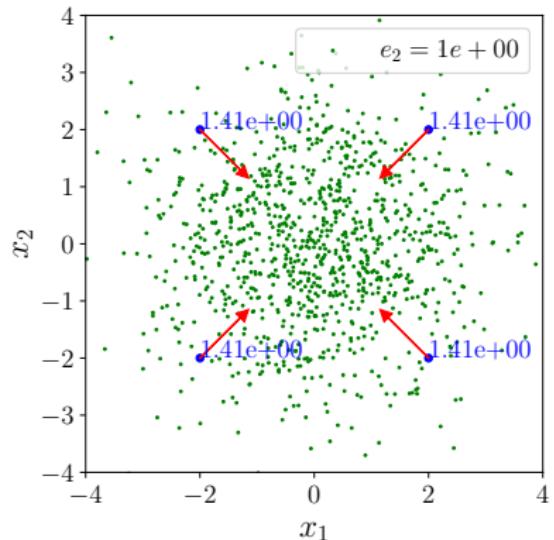
- Manifold hypothesis
- Inaccurate score estimation with score matching
- Slow mixing of Langevin dynamics

# Manifold hypothesis

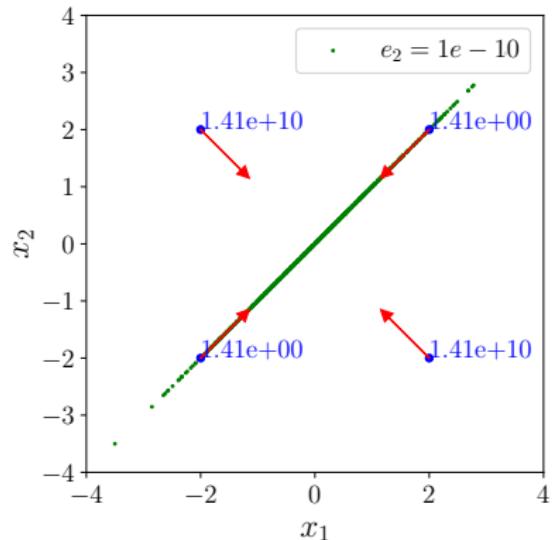


(a) Dataset in  $\mathbb{R}^2$  space

# Manifold hypothesis



(a) Dataset in  $\mathbb{R}^2$  space



(b) Dataset on one-dimensional manifold embedded in  $\mathbb{R}^2$  space (ambient space)

**Figure:** Samples from  $\boldsymbol{x} \sim \mathcal{N}\left(\begin{bmatrix} 0, 0 \end{bmatrix}^T, \boldsymbol{\Sigma}_{e_2}\right)$ ,  $\boldsymbol{\Sigma} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} [1, 1] + e_2 \begin{bmatrix} 1 \\ -1 \end{bmatrix} [1, -1]$ . The red arrows show the direction of the score function at the start of the arrow. The length of the score function is written in blue near each arrow.

# Manifold Hypothesis

## Higher Dimension Datasets [10]

Assume the MNIST dataset where each vectorized image  $\mathbf{x}$  is in  $\mathbb{R}^{784}$ .

# Manifold Hypothesis

## Higher Dimension Datasets [10]

Assume the MNIST dataset where each vectorized image  $\mathbf{x}$  is in  $\mathbb{R}^{784}$ .

- $\mathbb{R}^{784}$  is known as the ambient space of the dataset.

# Manifold Hypothesis

## Higher Dimension Datasets [10]

Assume the MNIST dataset where each vectorized image  $\mathbf{x}$  is in  $\mathbb{R}^{784}$ .

- $\mathbb{R}^{784}$  is known as the ambient space of the dataset.
- In practice, data samples lie on a lower dimensional manifold which shows the real dimension of data.

# Manifold Hypothesis

## Higher Dimension Datasets [10]

Assume the MNIST dataset where each vectorized image  $\mathbf{x}$  is in  $\mathbb{R}^{784}$ .

- $\mathbb{R}^{784}$  is known as the ambient space of the dataset.
- In practice, data samples lie on a lower dimensional manifold which shows the real dimension of data.
- There are locations in the ambient space (outside the manifold) where you have almost no sample and score function vector length tends to infinity.

# Manifold Hypothesis

## Higher Dimension Datasets [10]

Assume the MNIST dataset where each vectorized image  $\mathbf{x}$  is in  $\mathbb{R}^{784}$ .

- $\mathbb{R}^{784}$  is known as the ambient space of the dataset.
- In practice, data samples lie on a lower dimensional manifold which shows the real dimension of data.
- There are locations in the ambient space (outside the manifold) where you have almost no sample and score function vector length tends to infinity.
- Score matching objective is consistent estimator only when the support of the data distribution is the whole space [7].

# Manifold Hypothesis

## Higher Dimension Datasets [10]

Assume the MNIST dataset where each vectorized image  $\mathbf{x}$  is in  $\mathbb{R}^{784}$ .

- $\mathbb{R}^{784}$  is known as the ambient space of the dataset.
  - In practice, data samples lie on a lower dimensional manifold which shows the real dimension of data.
  - There are locations in the ambient space (outside the manifold) where you have almost no sample and score function vector length tends to infinity.
  - Score matching objective is consistent estimator only when the support of the data distribution is the whole space [7].
- ☞ The resulting obstacle is known as is known as *Manifold Hypothesis*.

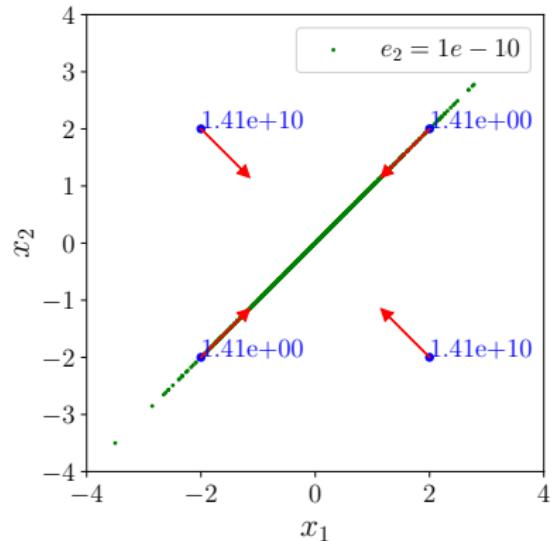
# Manifold Hypothesis

## Higher Dimension Datasets [10]

Assume the MNIST dataset where each vectorized image  $\mathbf{x}$  is in  $\mathbb{R}^{784}$ .

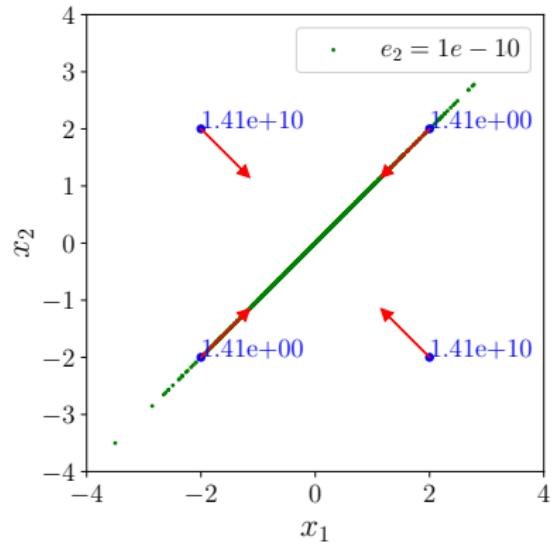
- $\mathbb{R}^{784}$  is known as the ambient space of the dataset.
  - In practice, data samples lie on a lower dimensional manifold which shows the real dimension of data.
  - There are locations in the ambient space (outside the manifold) where you have almost no sample and score function vector length tends to infinity.
  - Score matching objective is consistent estimator only when the support of the data distribution is the whole space [7].
- ☞ The resulting obstacle is known as is known as *Manifold Hypothesis*.
- ☞ Song and Ermon showed that adding noise to the original dataset can solve manifold hypothesis obstacle [10].

# Noisy Dataset

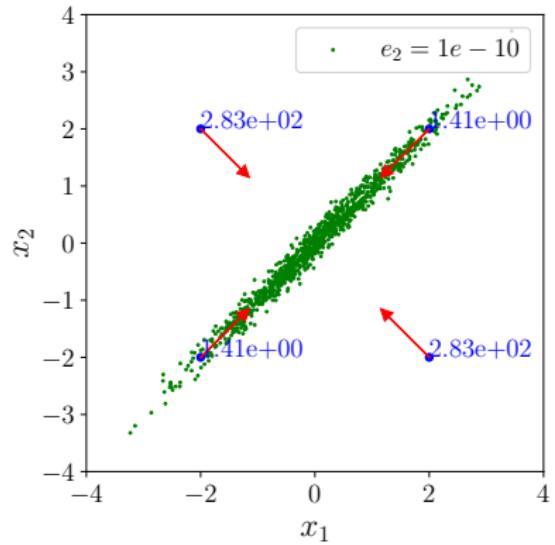


(a)  $\mathcal{D}$

# Noisy Dataset



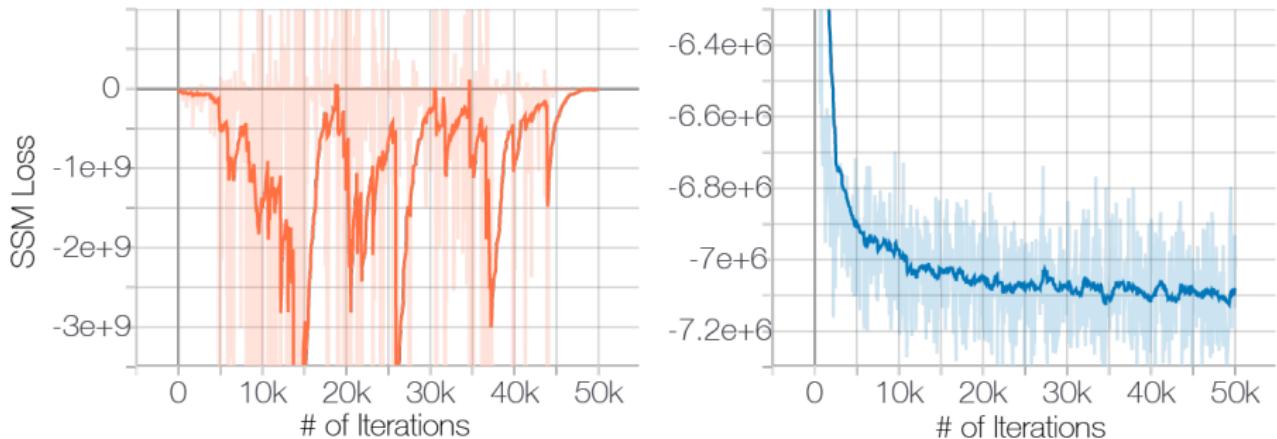
(a)  $\mathcal{D}$



(b)  $\mathcal{D}_{\text{noisy}}, q_{\sigma}(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, 10^{-4} \mathbf{I})$

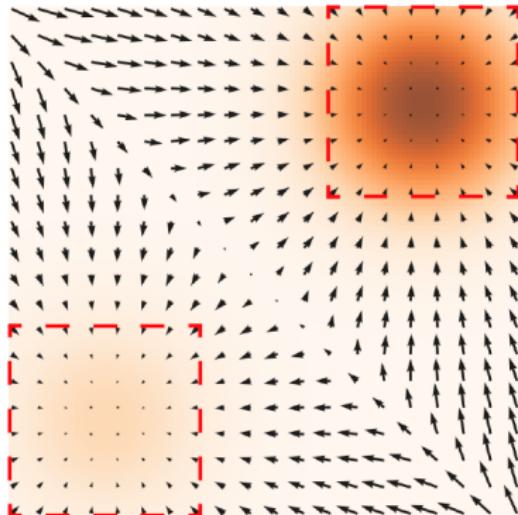
**Figure:** Samples and score function output for two datasets. Adding noise with low variance reduces the score function vector length significantly

# SM Training with Noisy Dataset



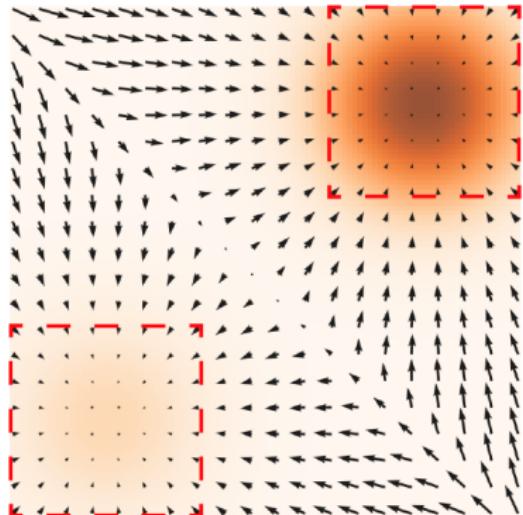
**Figure:** Left: Sliced score matching (SSM) loss with  $\mathcal{D}$  training dataset. Right: Sliced score matching (SSM) loss with  $\mathcal{D}_{\text{noisy}}$  training dataset and  $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\hat{\mathbf{x}}|\mathbf{x}, 10^{-4})$  (source: [10])

# Inaccurate score estimation with score matching

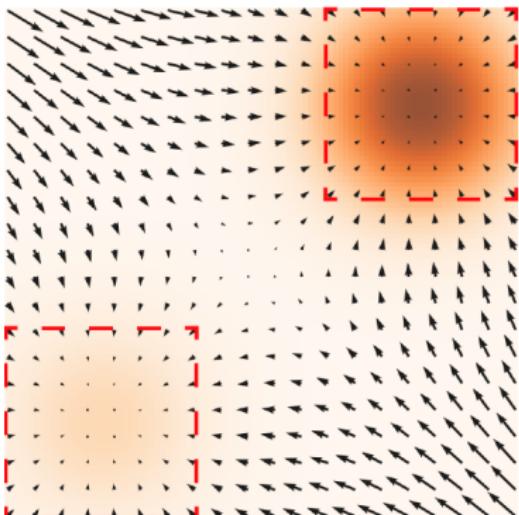


(a)  $\mathbf{s}_{\text{data}}(\theta)$  (source: [10])

# Inaccurate score estimation with score matching

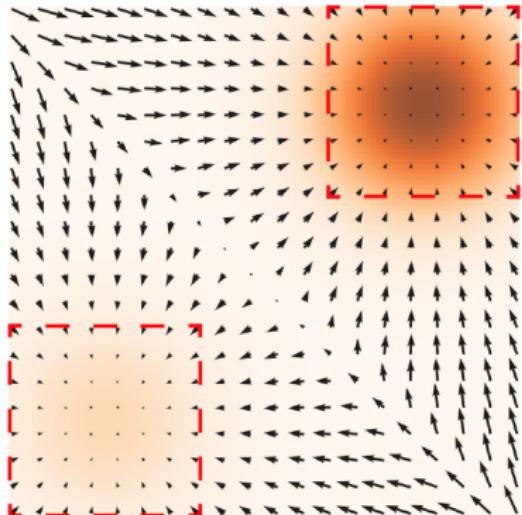


(a)  $\mathbf{s}_{\text{data}}(\theta)$  (source: [10])

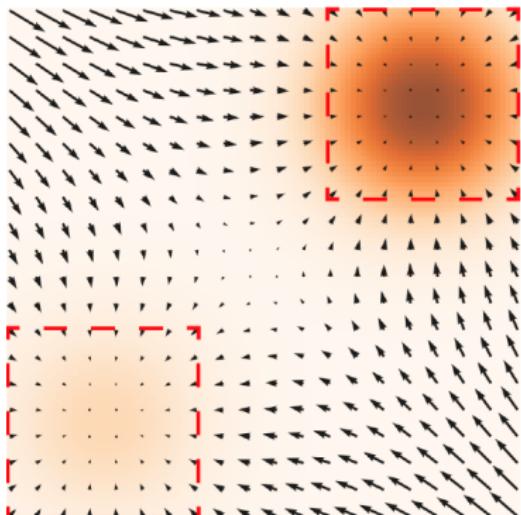


(b)  $\mathbf{s}_\theta(\mathbf{x})$  after training (source: [10])

# Inaccurate score estimation with score matching



(a)  $\mathbf{s}_{\text{data}}(\theta)$  (source: [10])



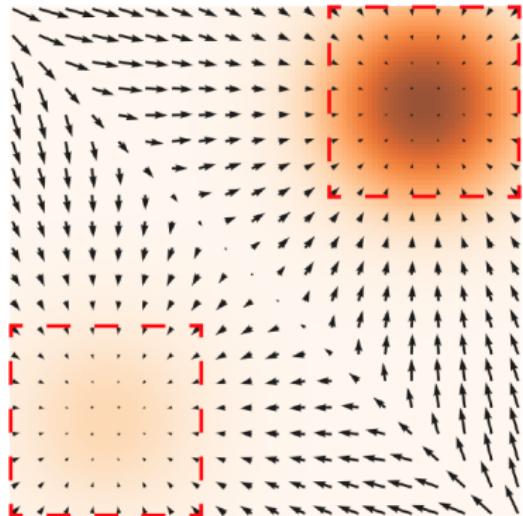
(b)  $\mathbf{s}_\theta(\mathbf{x})$  after training (source: [10])

## Intuition

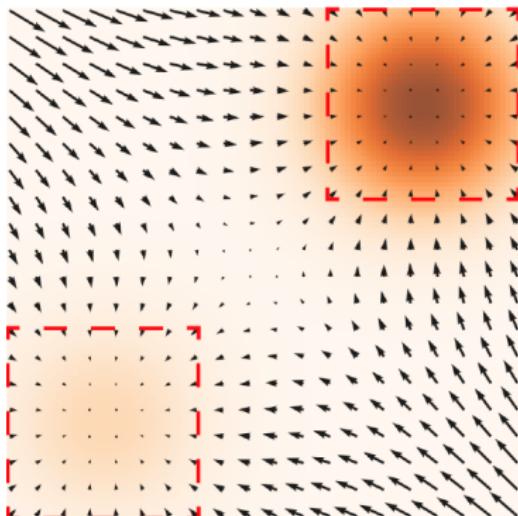
While both ambient and dataset space is  $\mathbb{R}^2$ , but we have:

- High estimation accuracy in high probability regions (Red rectangles)

# Inaccurate score estimation with score matching



(a)  $\mathbf{s}_{\text{data}}(\theta)$  (source: [10])



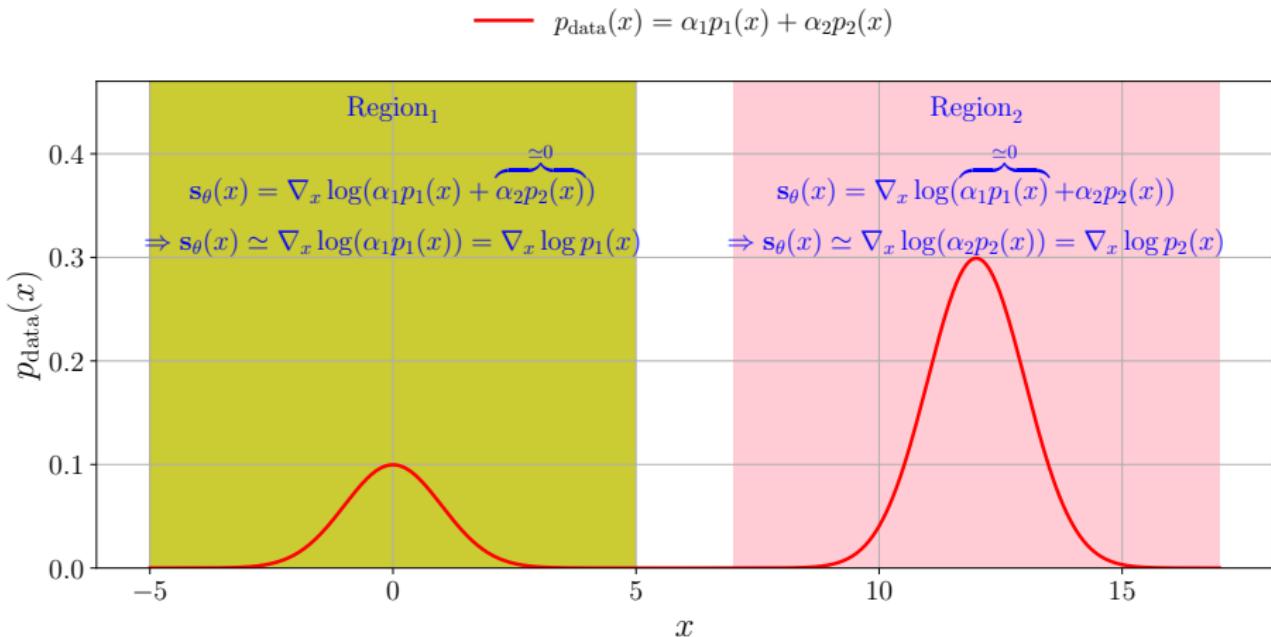
(b)  $\mathbf{s}_\theta(\mathbf{x})$  after training (source: [10])

## Intuition

While both ambient and dataset space is  $\mathbb{R}^2$ , but we have:

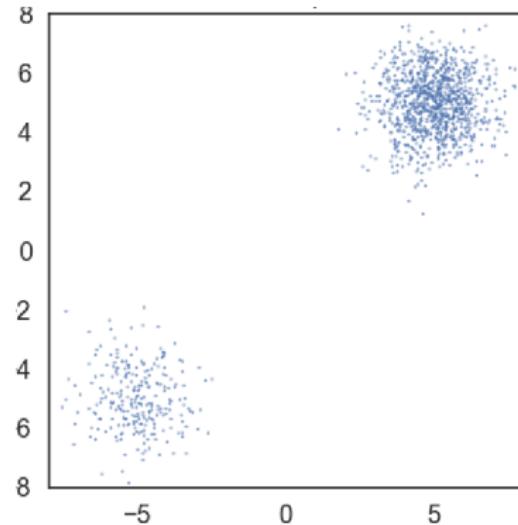
- High estimation accuracy in high probability regions (Red rectangles)
- Low estimation accuracy in low probability regions

# Slow mixing of Langevin Dynamics



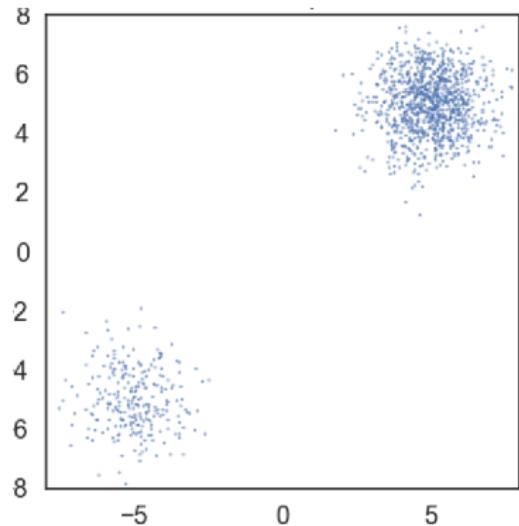
**Figure:** Slow mixing of Langevin dynamics (When the components support is disjoint (Region<sub>1</sub> and Region<sub>2</sub>), the score function is almost independent of mixing ratios  $\pi_1$  and  $\pi_2$ ).

# Slow Mixing of Langevin Dynamics [10]

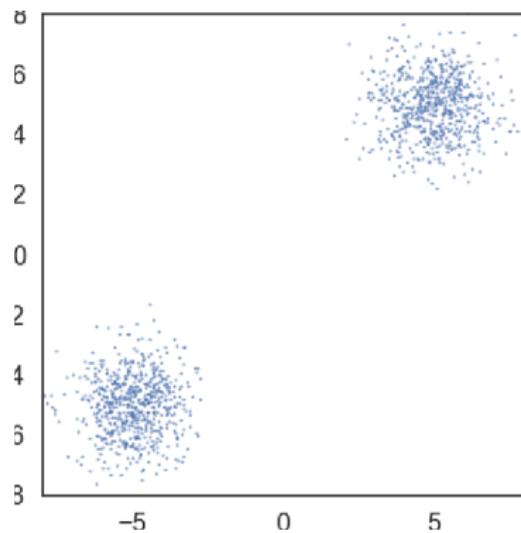


(a) iid samples from  $p_{\text{data}}(\mathbf{X})$

# Slow Mixing of Langevin Dynamics [10]



(a) iid samples from  $p_{\text{data}}(\mathbf{X})$



(b) Samples generated by LMC

**Figure:** The effect of Slow mixing of Langevin Dynamics when sampling using LMC without MH correction. we can see the resulting samples are equally assigned to each of the distribution modes because the score function is almost independent of  $\pi_1$  and  $\pi_2$  (source: [10])

# Noise Conditional Score Networks

## Intuition

Song and Ermon observe that perturbing data with random Gaussian noise removes three obstacles [10]:

- Perturbed data will not be confined to a low-dimensional manifold.

# Noise Conditional Score Networks

## Intuition

Song and Ermon observe that perturbing data with random Gaussian noise removes three obstacles [10]:

- Perturbed data will not be confined to a low-dimensional manifold.
- Gaussian noise can fill the low-density region and can provide more training signals for training.

## Intuition

Song and Ermon observe that perturbing data with random Gaussian noise removes three obstacles [10]:

- Perturbed data will not be confined to a low-dimensional manifold.
- Gaussian noise can fill the low-density region and can provide more training signals for training.
- Sampling from the score-based generative model can overcome the Slow mixing of Langevin dynamics by sampling from a set of models with decreasing noise perturbation level (*Annealed Langevin dynamics* algorithm).

# Noise Conditional Score Networks

## Prerequisites

- $\{\sigma_i\}_{i=1}^L$ : Positive geometric sequence such that  $\frac{\sigma_1}{\sigma_2} = \dots = \frac{\sigma_{L-1}}{\sigma_L} > 1$ .

# Noise Conditional Score Networks

## Prerequisites

- $\{\sigma_i\}_{i=1}^L$ : Positive geometric sequence such that  $\frac{\sigma_1}{\sigma_2} = \dots = \frac{\sigma_{L-1}}{\sigma_L} > 1$ .
- $\begin{cases} \sigma_1 : \text{Large enough to remove the obstacle} \\ \sigma_L : \text{Small enough such that } q_\sigma \text{ and } p_{\text{data}} \text{ are almost equal} \end{cases}$

## Prerequisites

- $\{\sigma_i\}_{i=1}^L$ : Positive geometric sequence such that  $\frac{\sigma_1}{\sigma_2} = \dots = \frac{\sigma_{L-1}}{\sigma_L} > 1$ .
- $\begin{cases} \sigma_1 : \text{Large enough to remove the obstacle} \\ \sigma_L : \text{Small enough such that } q_\sigma \text{ and } p_{\text{data}} \text{ are almost equal} \end{cases}$
- Perturbed dataset:  $q_\sigma(\tilde{\mathbf{x}}) \triangleq \int p_{\text{data}}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) d\mathbf{x}, \quad q_\sigma(\tilde{\mathbb{X}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbb{X}}|\mathbf{x}, \sigma^2 \mathbf{I})$

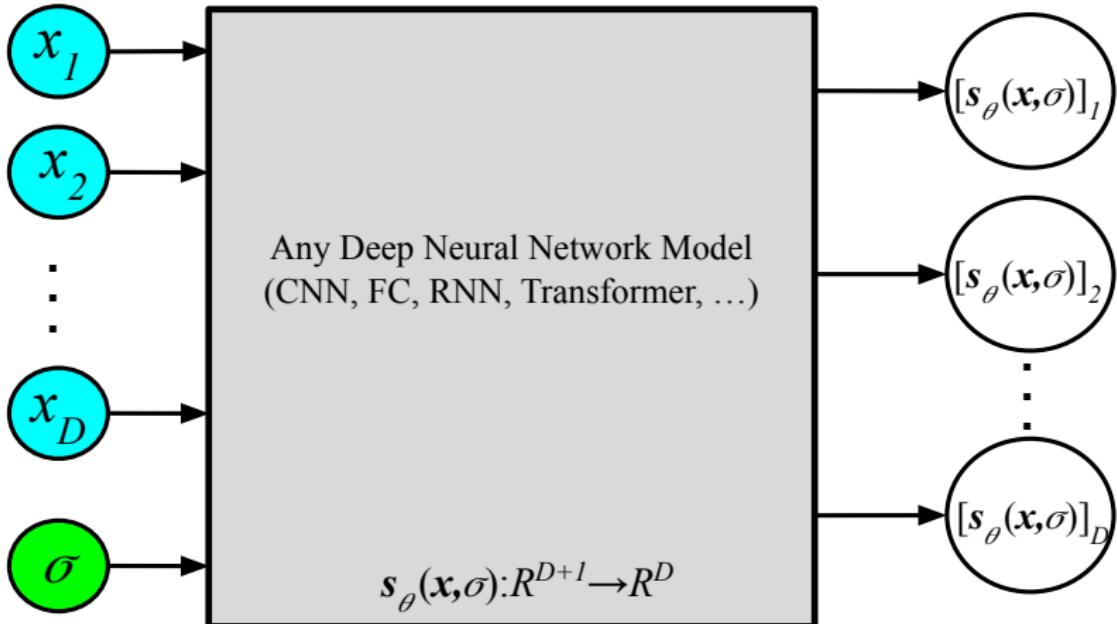
# Noise Conditional Score Networks

## Prerequisites

- $\{\sigma_i\}_{i=1}^L$ : Positive geometric sequence such that  $\frac{\sigma_1}{\sigma_2} = \dots = \frac{\sigma_{L-1}}{\sigma_L} > 1$ .
- $\begin{cases} \sigma_1 : \text{Large enough to remove the obstacle} \\ \sigma_L : \text{Small enough such that } q_\sigma \text{ and } p_{\text{data}} \text{ are almost equal} \end{cases}$
- Perturbed dataset:  $q_\sigma(\tilde{\mathbf{x}}) \triangleq \int p_{\text{data}}(\mathbf{x}) q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) d\mathbf{x}$ ,  $q_\sigma(\tilde{\mathbb{X}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbb{X}}|\mathbf{x}, \sigma^2 \mathbf{I})$
- ☞ Objective: Train conditional score network  $\mathbf{s}_\theta(\mathbf{x}, \sigma)$ , known as *Noise Conditional Score Network* (NCSN), that can estimate score function for all perturbed data as:

$$\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma) \simeq \nabla_{\mathbf{x}} \log q_\sigma(\tilde{\mathbf{x}}), \quad \forall \sigma \in \{\sigma_i\}_{i=1}^L$$

# Noise Conditional Score Network



**Figure:** The model score function used for training based on Noise Conditional Score Network

# Noise Conditional Score Network

## Training

Consider  $\{\sigma_i\}_{i=1}^L$  and assume  $L = 1$ . Then we have ordinary DSM objectives:

$$L_{DSM, \sigma_1}(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, \sigma_1^2 \mathbf{I})} \left[ \|\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma_1) - \frac{1}{\sigma_1^2}(\mathbf{x} - \tilde{\mathbf{x}})\|^2 \right]$$

# Noise Conditional Score Network

## Training

Consider  $\{\sigma_i\}_{i=1}^L$  and assume  $L = 1$ . Then we have ordinary DSM objectives:

$$L_{DSM, \sigma_1}(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, \sigma_1^2 \mathbf{I})} \left[ \|\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma_1) - \frac{1}{\sigma_1^2}(\mathbf{x} - \tilde{\mathbf{x}})\|^2 \right]$$

Note:

- Nothing changed in comparison to DSM because  $\sigma_1$  is fixed.

# Noise Conditional Score Network

## Training

Consider  $\{\sigma_i\}_{i=1}^L$  and assume  $L = 1$ . Then we have ordinary DSM objectives:

$$L_{DSM, \sigma_1}(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, \sigma_1^2 \mathbf{I})} \left[ \|\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma_1) - \frac{1}{\sigma_1^2} (\mathbf{x} - \tilde{\mathbf{x}})\|^2 \right]$$

Note:

- Nothing changed in comparison to DSM because  $\sigma_1$  is fixed.
- We know the resulting network  $\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma_1) = \nabla_{\mathbf{x}} \log q_{\sigma_1}(\tilde{\mathbf{x}})$

# Noise Conditional Score Network

## Training

Consider  $\{\sigma_i\}_{i=1}^L$  and assume  $L = 1$ . Then we have ordinary DSM objectives:

$$L_{DSM, \sigma_1}(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbb{X})} \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, \sigma_1^2 \mathbf{I})} \left[ \|\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma_1) - \frac{1}{\sigma_1^2}(\mathbf{x} - \tilde{\mathbf{x}})\|^2 \right]$$

Note:

- Nothing changed in comparison to DSM because  $\sigma_1$  is fixed.
- We know the resulting network  $\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma_1) = \nabla_{\mathbf{x}} \log q_{\sigma_1}(\tilde{\mathbf{x}})$

For  $L \geq 1$  Noise Conditional Score Network objective is defined as:

$$L_{NCSM, \{\sigma_i\}_{i=1}^L}(\boldsymbol{\theta}) \triangleq \frac{1}{L} \sum_{i=1}^L \lambda(\sigma_i) L_{DSM, \sigma_i}(\boldsymbol{\theta})$$

where  $\lambda(\sigma_i) > 0$  is a coefficient depending on  $\sigma_i$  (example:  $\lambda(\sigma) = \sigma^2$ )

# Noise Conditional Score Network

## Inference

After training  $\mathbf{s}_\theta(\mathbf{x}, \sigma), \sigma \in \{\sigma_i\}_{i=1}^L$ :

- We have a pool of  $L$  score functions  $\{\mathbf{s}_\theta(\mathbf{x}, \sigma_i)\}_{i=1}^L$ .

# Noise Conditional Score Network

## Inference

After training  $\mathbf{s}_\theta(\mathbf{x}, \sigma), \sigma \in \{\sigma_i\}_{i=1}^L$ :

- We have a pool of  $L$  score functions  $\{\mathbf{s}_\theta(\mathbf{x}, \sigma_i)\}_{i=1}^L$ .
- $\mathbf{s}_\theta(\mathbf{x}, \sigma_L)$  is the score function corresponding to nearest random variable to  $\mathbb{X}$ , because  $\sigma_L < \sigma_i, i \in \{1, 2, \dots, L - 1\}$

# Noise Conditional Score Network

## Inference

After training  $\mathbf{s}_\theta(\mathbf{x}, \sigma), \sigma \in \{\sigma_i\}_{i=1}^L$ :

- We have a pool of  $L$  score functions  $\{\mathbf{s}_\theta(\mathbf{x}, \sigma_i)\}_{i=1}^L$ .
- $\mathbf{s}_\theta(\mathbf{x}, \sigma_L)$  is the score function corresponding to nearest random variable to  $\mathbb{X}$ , because  $\sigma_L < \sigma_i, i \in \{1, 2, \dots, L-1\}$
- If you use  $\mathbf{s}_\theta(\mathbf{x}, \sigma_L)$  to generate samples using LMC, then you have Slow Mixing of Langevin Dynamics.

# Noise Conditional Score Network

## Inference

After training  $\mathbf{s}_\theta(\mathbf{x}, \sigma), \sigma \in \{\sigma_i\}_{i=1}^L$ :

- We have a pool of  $L$  score functions  $\{\mathbf{s}_\theta(\mathbf{x}, \sigma_i)\}_{i=1}^L$ .
  - $\mathbf{s}_\theta(\mathbf{x}, \sigma_L)$  is the score function corresponding to nearest random variable to  $\mathbb{X}$ , because  $\sigma_L < \sigma_i, i \in \{1, 2, \dots, L-1\}$
  - If you use  $\mathbf{s}_\theta(\mathbf{x}, \sigma_L)$  to generate samples using LMC, then you have Slow Mixing of Langevin Dynamics.
- ☞ Song and Ermon proposed *Annealed Langevin Dynamics* (ALD) to make the use of all  $\sigma_L < \sigma_i, i \in \{1, 2, \dots, L-1\}$  and solve Slow Mixing of Langevin Dynamics.

# Annealed Langevin Dynamics

## General Idea

Annealed Langevin Dynamics draw samples from  $p_\theta(\mathbb{X})$  using the following steps:

# Annealed Langevin Dynamics

## General Idea

Annealed Langevin Dynamics draw samples from  $p_\theta(\mathbb{X})$  using the following steps:

- Sample  $\tilde{\mathbf{x}}$  from  $q_{\sigma_1}(\tilde{\mathbf{x}})$  using LMC based on  $\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma_1)$

# Annealed Langevin Dynamics

## General Idea

Annealed Langevin Dynamics draw samples from  $p_\theta(\mathbb{X})$  using the following steps:

- Sample  $\tilde{\mathbf{x}}$  from  $q_{\sigma_1}(\tilde{\mathbf{x}})$  using LMC based on  $\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma_1)$
- Use  $\tilde{\mathbf{x}}$  as the initialization for sampling  $q_{\sigma_2}(\tilde{\mathbf{x}})$  based on  $\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma_2)$

# Annealed Langevin Dynamics

## General Idea

Annealed Langevin Dynamics draw samples from  $p_\theta(\mathbb{X})$  using the following steps:

- Sample  $\tilde{\mathbf{x}}$  from  $q_{\sigma_1}(\tilde{\mathbf{x}})$  using LMC based on  $\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma_1)$
- Use  $\tilde{\mathbf{x}}$  as the initialization for sampling  $q_{\sigma_2}(\tilde{\mathbf{x}})$  based on  $\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma_2)$
- Repeat the above procedure until sampling  $q_{\sigma_L}(\tilde{\mathbf{x}})$  based on  $\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma_L)$

# Annealed Langevin Dynamics

## General Idea

Annealed Langevin Dynamics draw samples from  $p_\theta(\mathbb{X})$  using the following steps:

- Sample  $\tilde{\mathbf{x}}$  from  $q_{\sigma_1}(\tilde{\mathbf{x}})$  using LMC based on  $\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma_1)$
- Use  $\tilde{\mathbf{x}}$  as the initialization for sampling  $q_{\sigma_2}(\tilde{\mathbf{x}})$  based on  $\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma_2)$
- Repeat the above procedure until sampling  $q_{\sigma_L}(\tilde{\mathbf{x}})$  based on  $\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma_L)$

The above procedure can solve the problem of Slow Mixing of Langevin Dynamics.

# Annealed Langevin Dynamics Algorithm

---

**Algorithm 36** Sampling from  $p_\theta(\mathbb{X})$  using Annealed Langevin Dynamics Algorithm

---

1: **procedure** ANNEALED LANGEVIN DYNAMICS( $\{\sigma_i\}_{i=1}^L, \{\mathbf{s}_\theta(\mathbf{x}, \sigma_i)\}_{i=1}^L, T, \epsilon$ )

# Annealed Langevin Dynamics Algorithm

---

**Algorithm 37** Sampling from  $p_\theta(\mathbb{X})$  using Annealed Langevin Dynamics Algorithm

---

```
1: procedure ANNEALED LANGEVIN DYNAMICS( $\{\sigma_i\}_{i=1}^L, \{\mathbf{s}_\theta(\mathbf{x}, \sigma_i)\}_{i=1}^L, T, \epsilon$ )
2:   Initialization:  $\tilde{\mathbf{x}}^0$ 
```

# Annealed Langevin Dynamics Algorithm

---

**Algorithm 38** Sampling from  $p_\theta(\mathbb{X})$  using Annealed Langevin Dynamics Algorithm

---

```
1: procedure ANNEALED LANGEVIN DYNAMICS( $\{\sigma_i\}_{i=1}^L, \{\mathbf{s}_\theta(\mathbf{x}, \sigma_i)\}_{i=1}^L, T, \epsilon$ )
2:   Initialization:  $\tilde{\mathbf{x}}^0$ 
3:   for  $i = 1 : L$  do
4:     Calculate the step size:  $\alpha_i = \epsilon \frac{\sigma_i^2}{\sigma_L^2}$ 
```

# Annealed Langevin Dynamics Algorithm

---

**Algorithm 39** Sampling from  $p_\theta(\mathbb{X})$  using Annealed Langevin Dynamics Algorithm

---

```
1: procedure ANNEALED LANGEVIN DYNAMICS( $\{\sigma_i\}_{i=1}^L, \{\mathbf{s}_\theta(\mathbf{x}, \sigma_i)\}_{i=1}^L, T, \epsilon$ )
2:   Initialization:  $\tilde{\mathbf{x}}^0$ 
3:   for  $i = 1 : L$  do
4:     Calculate the step size:  $\alpha_i = \epsilon \frac{\sigma_i^2}{\sigma_L^2}$ 
5:     for  $t = 1 : T$  do
6:       Sample  $\mathbf{v}_t \sim \mathcal{N}(0, \mathbf{I})$ 
```

# Annealed Langevin Dynamics Algorithm

---

**Algorithm 40** Sampling from  $p_\theta(\mathbb{X})$  using Annealed Langevin Dynamics Algorithm

---

```
1: procedure ANNEALED LANGEVIN DYNAMICS( $\{\sigma_i\}_{i=1}^L, \{\mathbf{s}_\theta(\mathbf{x}, \sigma_i)\}_{i=1}^L, T, \epsilon$ )
2:   Initialization:  $\tilde{\mathbf{x}}^0$ 
3:   for  $i = 1 : L$  do
4:     Calculate the step size:  $\alpha_i = \epsilon \frac{\sigma_i^2}{\sigma_L^2}$ 
5:     for  $t = 1 : T$  do
6:       Sample  $\mathbf{v}_t \sim \mathcal{N}(0, \mathbf{I})$ 
7:       Update sample as:
           
$$\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} \mathbf{s}_\theta(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i} \mathbf{v}_t$$

8:   end for
```

# Annealed Langevin Dynamics Algorithm

---

**Algorithm 41** Sampling from  $p_\theta(\mathbb{X})$  using Annealed Langevin Dynamics Algorithm

---

```
1: procedure ANNEALED LANGEVIN DYNAMICS( $\{\sigma_i\}_{i=1}^L, \{\mathbf{s}_\theta(\mathbf{x}, \sigma_i)\}_{i=1}^L, T, \epsilon$ )
2:   Initialization:  $\tilde{\mathbf{x}}^0$ 
3:   for  $i = 1 : L$  do
4:     Calculate the step size:  $\alpha_i = \epsilon \frac{\sigma_i^2}{\sigma_L^2}$ 
5:     for  $t = 1 : T$  do
6:       Sample  $\mathbf{v}_t \sim \mathcal{N}(0, \mathbf{I})$ 
7:       Update sample as:
           
$$\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} \mathbf{s}_\theta(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i} \mathbf{v}_t$$

8:     end for
9:    $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$ 
```

# Annealed Langevin Dynamics Algorithm

---

**Algorithm 42** Sampling from  $p_\theta(\mathbb{X})$  using Annealed Langevin Dynamics Algorithm

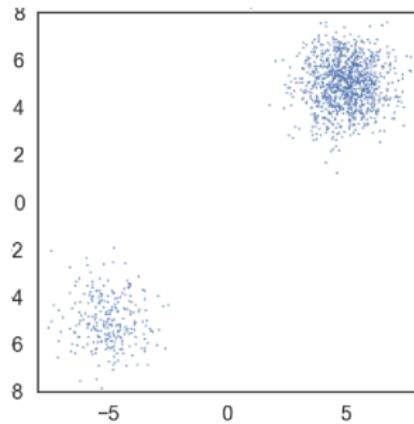
---

```
1: procedure ANNEALED LANGEVIN DYNAMICS( $\{\sigma_i\}_{i=1}^L, \{\mathbf{s}_\theta(\mathbf{x}, \sigma_i)\}_{i=1}^L, T, \epsilon$ )
2:   Initialization:  $\tilde{\mathbf{x}}^0$ 
3:   for  $i = 1 : L$  do
4:     Calculate the step size:  $\alpha_i = \epsilon \frac{\sigma_i^2}{\sigma_L^2}$ 
5:     for  $t = 1 : T$  do
6:       Sample  $\mathbf{v}_t \sim \mathcal{N}(0, \mathbf{I})$ 
7:       Update sample as:
           
$$\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} \mathbf{s}_\theta(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i} \mathbf{v}_t$$

8:     end for
9:      $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$ 
10:    end for
11: end procedure
```

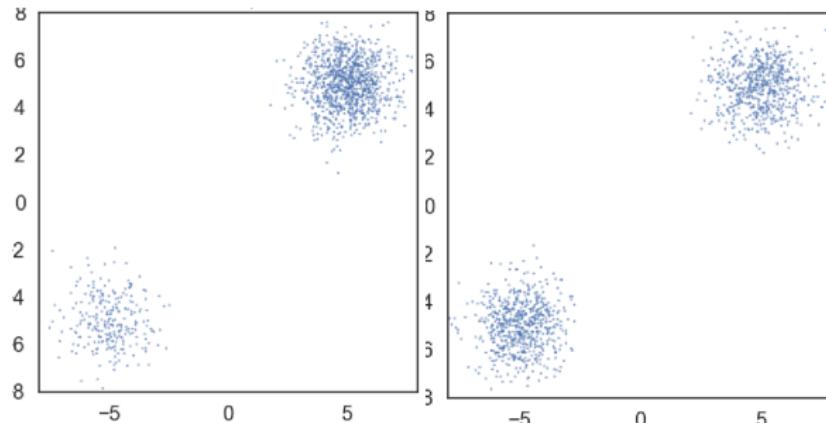
---

# Annealed Langevin Dynamics Algorithm [10]



(a) Samples from  $p_{\text{data}}(\mathbb{X})$

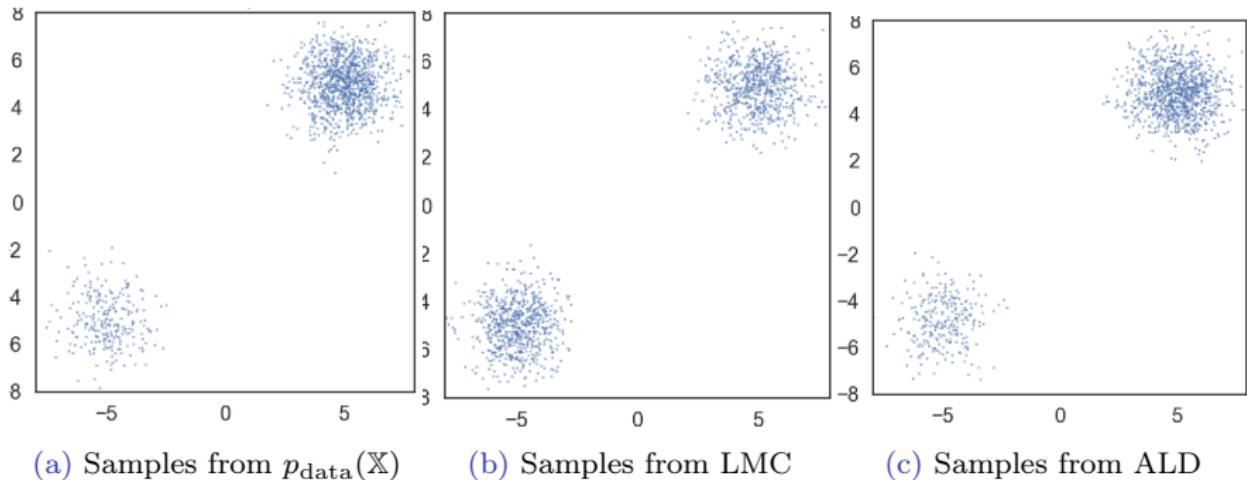
# Annealed Langevin Dynamics Algorithm [10]



(a) Samples from  $p_{\text{data}}(\mathbb{X})$

(b) Samples from LMC

# Annealed Langevin Dynamics Algorithm [10]



**Figure:** Annealed Langevin Dynamics Algorithm results. We can see the resulting samples almost preserve the ratio of each mode in the data distribution (source: [10]).

# Annealed Langevin Dynamics Algorithm



Figure: ALD middle and final samples over CelebA dataset (source: [10])

# Annealed Langevin Dynamics Algorithm



Figure: ALD middle and final samples over Cifar10 dataset (source: [10])

# Annealed Langevin Dynamics Algorithm

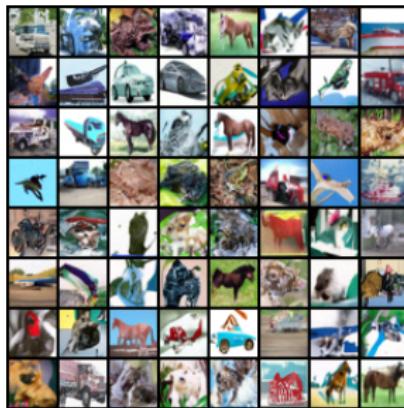


(a) MNIST dataset

# Annealed Langevin Dynamics Algorithm



(a) MNIST dataset

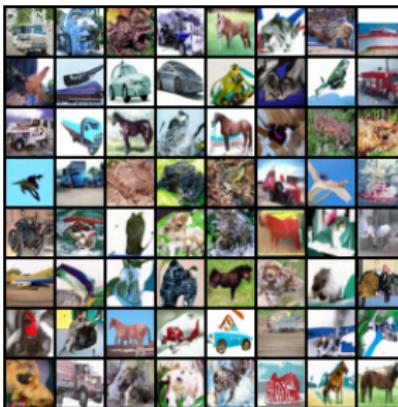


(b) Cifar10 dataset

# Annealed Langevin Dynamics Algorithm



(a) MNIST dataset



(b) Cifar10 dataset



(c) CelebA dataset

Figure: ALD sample over different datasets (source: [10])

# References I

-  Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang,  
“A tutorial on energy-based learning,”  
*Predicting structured data*, vol. 1, no. 0, 2006.
-  Stratis Markou and James Allingham,  
“Energy-based models.” .
-  Abdul Fatir Ansari,  
“Metropolis hastings,” <https://github.com/abdlutfatir/sampling-methods-numpy/blob/master/Metropolis-Hastings.ipynb>,  
2018.
-  Ali Siahkoohi,  
“Sampling with gradient-based markov chain monte carlo approaches,”  
<https://github.com/alisiahkoohi/Langevin-dynamics>, 2023.
-  Geoffrey E Hinton,  
“Training products of experts by minimizing contrastive divergence,”  
*Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.
-  Phillip Lippe,  
“UvA Deep Learning Tutorials,” <https://uvadlc-notebooks.readthedocs.io/en/latest/>, 2023.
-  Aapo Hyvärinen and Peter Dayan,  
“Estimation of non-normalized statistical models by score matching.,”  
*Journal of Machine Learning Research*, vol. 6, no. 4, 2005.

## References II

-  Pascal Vincent,  
“A connection between score matching and denoising autoencoders,”  
*Neural computation*, vol. 23, no. 7, pp. 1661–1674, 2011.
-  Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon,  
“Sliced score matching: A scalable approach to density and score estimation,”  
in *Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 574–584.
-  Yang Song and Stefano Ermon,  
“Generative modeling by estimating gradients of the data distribution,”  
*Advances in neural information processing systems*, vol. 32, 2019.