CSCI 699 - ProbGen

Probabilistic and Generative Models

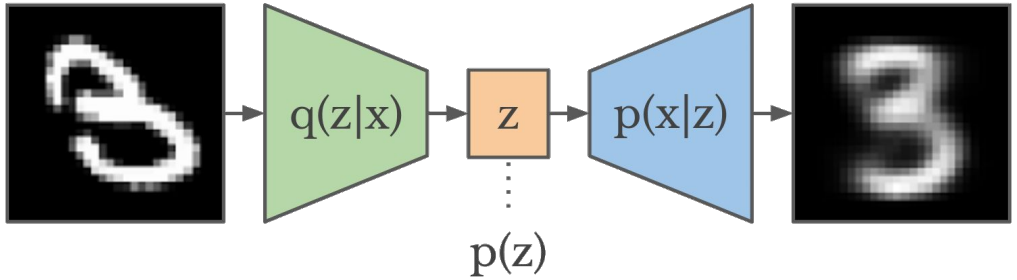
Willie Neiswanger

Lecture 7 - VAEs (continued) and Intro to Score-based Generative Models

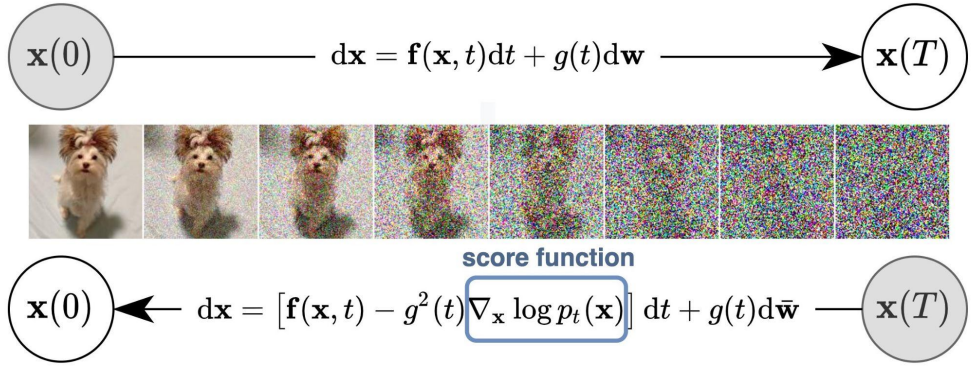
Today

Today

Lecture: Continued explanation of the variational autoencoder (VAE), and then intro to score-based generative models.



"Building Variational Auto-Encoders in TensorFlow", Danijar Hafner

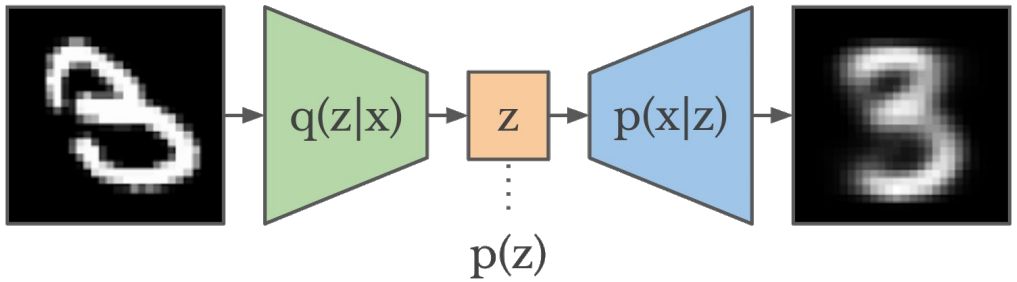


Source: Yang Song, "Generative Modeling by Estimating Gradients of the Data Distribution"

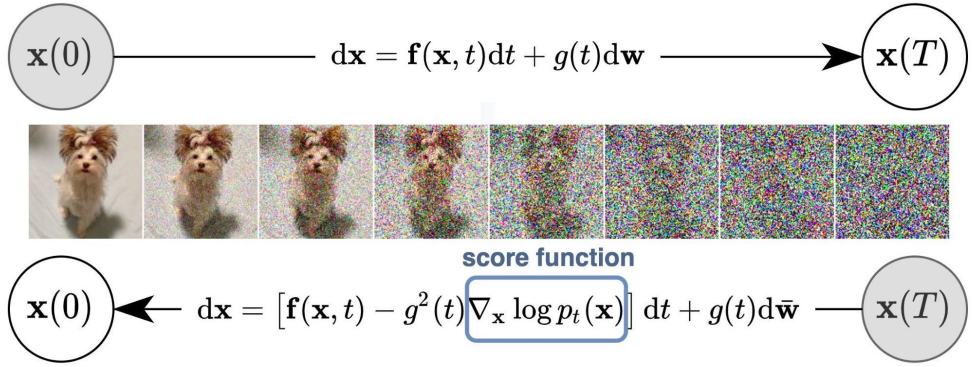
Today

Lecture: Continued explanation of the variational autoencoder (VAE), and then intro to score-based generative models.

- Review: Variational inference (VI) methods
- Variational autoencoders (VAEs)
  - Reparameterization trick, SGVB.
  - Relation to (classic) autoencoders.



"Building Variational Auto-Encoders in TensorFlow", Danijar Hafner

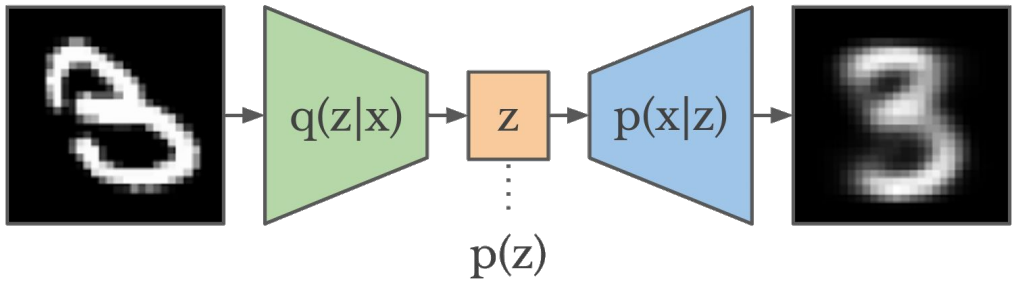


Source: Yang Song, "Generative Modeling by Estimating Gradients of the Data Distribution"

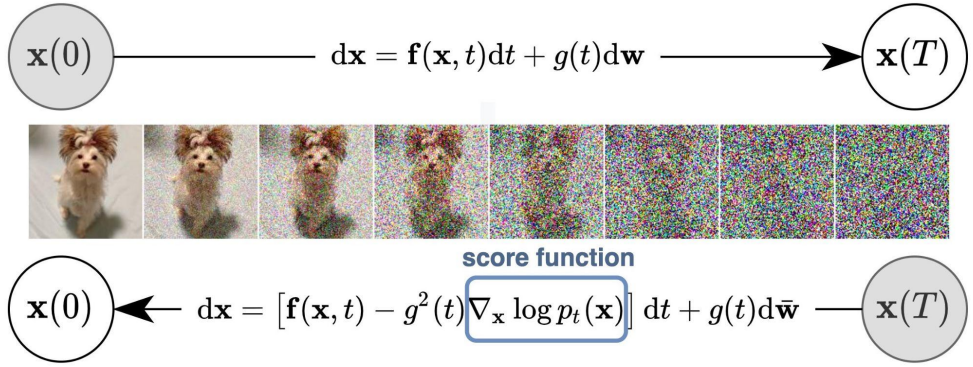
Today

Lecture: Continued explanation of the variational autoencoder (VAE), and then intro to score-based generative models.

- Review: Variational inference (VI) methods
- Variational autoencoders (VAEs)
  - Reparameterization trick, SGVB.
  - Relation to (classic) autoencoders.
- Intro to score-based generative models



"Building Variational Auto-Encoders in TensorFlow", Danijar Hafner

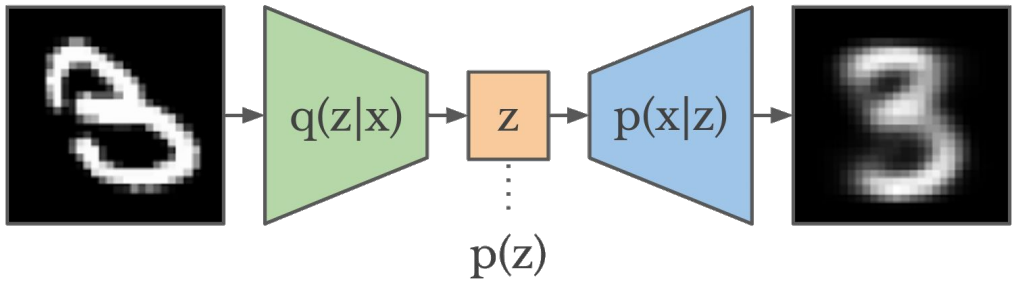


Source: Yang Song, "Generative Modeling by Estimating Gradients of the Data Distribution"

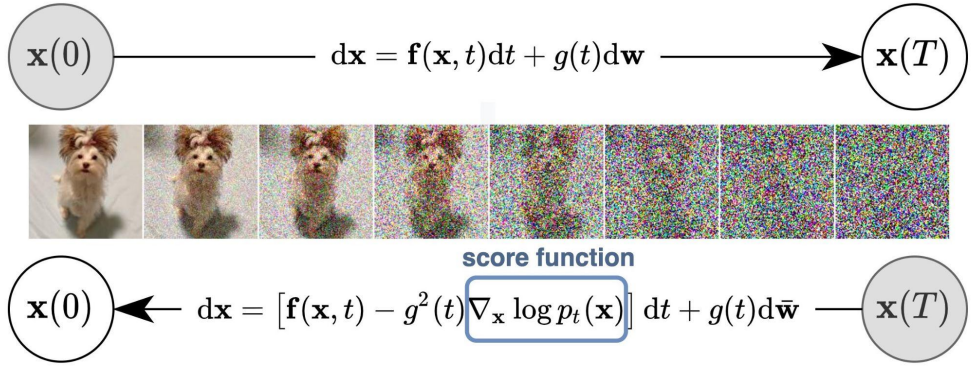
Today

Lecture: Continued explanation of the variational autoencoder (VAE), and then intro to score-based generative models.

- Review: Variational inference (VI) methods
- Variational autoencoders (VAEs)
  - Reparameterization trick, SGVB.
  - Relation to (classic) autoencoders.
- Intro to score-based generative models



"Building Variational Auto-Encoders in TensorFlow", Danijar Hafner



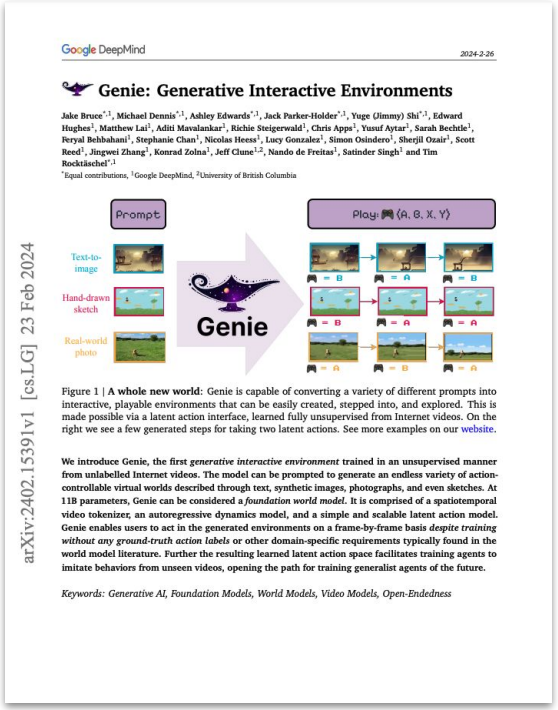
Source: Yang Song, "Generative Modeling by Estimating Gradients of the Data Distribution"

After:

- First groups of four **paper presentations**.

Today

After: First groups of four **paper presentations**.



Google DeepMind 2024-2-26

Genie: Generative Interactive Environments

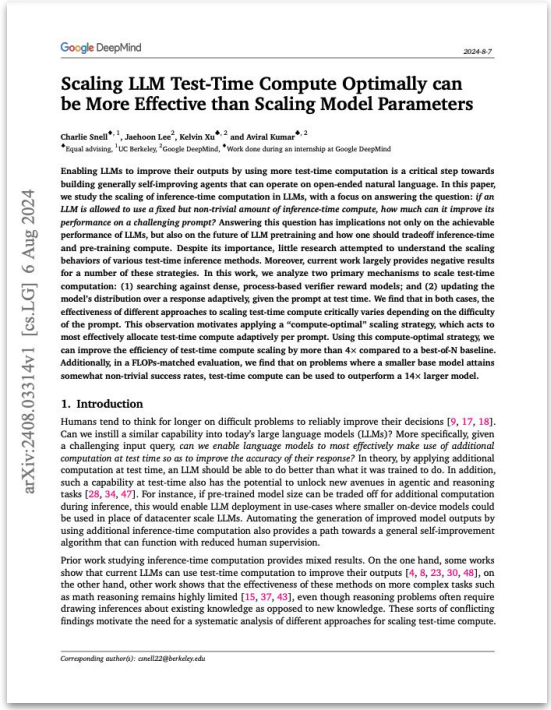
Jake Bruce*,1, Michael Dennis*,1, Ashley Edwards*,1, Jack Parker-Holder*,1, Yuge (Jimmy) Shi*,1, Edward Hughes1, Matthew Lai1, Aditi Mavalankar1, Richie Steigerwald1, Chris Apps1, Yusuf Aytar1, Sarah Bechtle1, Feryal Behbahani1, Stephanie Chan1, Nicolas Heess1, Lucy Gonzalez1, Simon Osindero1, Sherjil Ozair1, Scott Reed1, Jingwei Zhang1, Konrad Zolna1, Jeff Clune1,2, Nando de Freitas1, Satinder Singh1 and Tim Rocktäschel*,1

*Equal contributions, 1Google DeepMind, 2University of British Columbia

Figure 1 | **A whole new world**: Genie is capable of converting a variety of different prompts into interactive, playable environments that can be easily created, stepped into, and explored. This is made possible via a latent action interface, learned fully unsupervised from Internet videos. On the right we see a few generated steps for taking two latent actions. See more examples on our website.

We introduce Genie, the first *generative interactive environment* trained in an unsupervised manner from unlabelled Internet videos. The model can be prompted to generate an endless variety of action-controllable virtual worlds described through text, synthetic images, photographs, and even sketches. At 11B parameters, Genie can be considered a *foundation world model*. It is comprised of a spatiotemporal video tokenizer, an autoregressive dynamics model, and a simple and scalable latent action model. Genie enables users to act in the generated environments on a frame-by-frame basis *despite training without any ground-truth action labels* or other domain-specific requirements typically found in the world model literature. Further the resulting learned latent action space facilitates training agents to imitate behaviors from unseen videos, opening the path for training generalist agents of the future.

Keywords: Generative AI, Foundation Models, World Models, Video Models, Open-Endedness



Google DeepMind 2024-8-7

Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters

Charlie Snell♦,1, Jaehoon Lee2, Kelvin Xu♦,2 and Aviral Kumar♦,2

♦Equal advising, 1UC Berkeley, 2Google DeepMind, *Work done during an internship at Google DeepMind

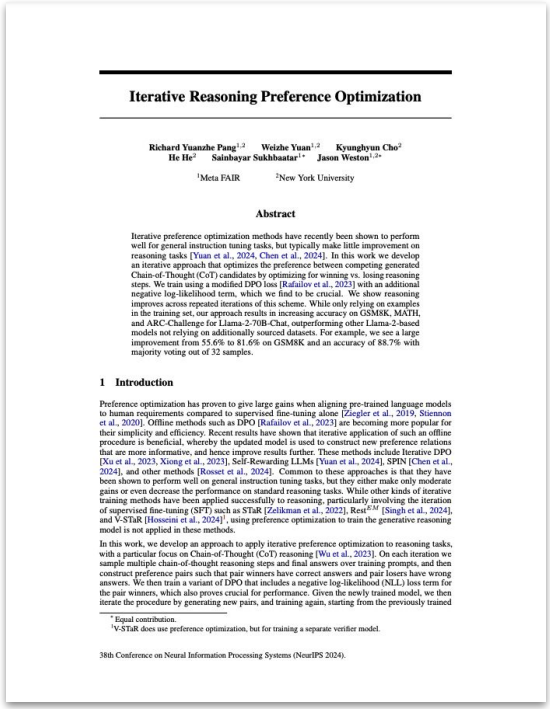
Enabling LLMs to improve their outputs by using more test-time computation is a critical step towards building generally self-improving agents that can operate on open-ended natural language. In this paper, we study the scaling of inference-time computation in LLMs, with a focus on answering the question: *if an LLM is allowed to use a fixed but non-trivial amount of inference-time compute, how much can it improve its performance on a challenging prompt?* Answering this question has implications not only on the achievable performance of LLMs, but also on the future of LLM pretraining and how one should tradeoff inference-time and pre-training compute. Despite its importance, little research attempted to understand the scaling behaviors of various test-time inference methods. Moreover, current work largely provides negative results for a number of these strategies. In this work, we analyze two primary mechanisms to scale test-time computation: (1) searching against dense, process-based verifier reward models; and (2) updating the model's distribution over a response adaptively, given the prompt at test time. We find that in both cases, the effectiveness of different approaches to scaling test-time compute critically varies depending on the difficulty of the prompt. This observation motivates applying a "compute-optimal" scaling strategy, which acts to most effectively allocate test-time compute adaptively per prompt. Using this compute-optimal strategy, we can improve the efficiency of test-time compute scaling by more than 4× compared to a best-of-N baseline. Additionally, in a FLOPs-matched evaluation, we find that on problems where a smaller base model attains somewhat non-trivial success rates, test-time compute can be used to outperform a 14× larger model.

1. Introduction

Humans tend to think for longer on difficult problems to reliably improve their decisions [9, 17, 18]. Can we instill a similar capability into today's large language models (LLMs)? More specifically, given a challenging input query, *can we enable language models to most effectively make use of additional computation at test time so as to improve the accuracy of their response?* In theory, by applying additional computation at test time, an LLM should be able to do better than what it was trained to do. In addition, such a capability at test-time also has the potential to unlock new avenues in agentic and reasoning tasks [28, 34, 47]. For instance, if pre-trained model size can be traded off for additional computation during inference, this would enable LLM deployment in use-cases where smaller on-device models could be used in place of datacenter scale LLMs. Automating the generation of improved model outputs by using additional inference-time computation also provides a path towards a general self-improvement algorithm that can function with reduced human supervision.

Prior work studying inference-time computation provides mixed results. On the one hand, some works show that current LLMs can use test-time computation to improve their outputs [4, 8, 23, 30, 48], on the other hand, other work shows that the effectiveness of these methods on more complex tasks such as math reasoning remains highly limited [15, 37, 43], even though reasoning problems often require drawing inferences about existing knowledge as opposed to new knowledge. These sorts of conflicting findings motivate the need for a systematic analysis of different approaches for scaling test-time compute.

Corresponding author(s): canell22@berkeley.edu



Iterative Reasoning Preference Optimization

Richard Yuanzhe Pang1,2 Weizhe Yuan1,2 Kyunghyun Cho2
He He2 Sainbayar Sukhbaatar1* Jason Weston1,2*

1Meta FAIR 2New York University

Abstract

Iterative preference optimization methods have recently been shown to perform well for general instruction tuning tasks, but typically make little improvement on reasoning tasks [Yuan et al., 2024, Chen et al., 2024]. In this work we develop an iterative approach that optimizes the preference between competing generated Chain-of-Thought (CoT) candidates by optimizing for winning vs. losing reasoning steps. We train using a modified DPO loss [Rafailov et al., 2023] with an additional negative log-likelihood term, which we find to be crucial. We show reasoning improves across repeated iterations of this scheme. While only relying on examples in the training set, our approach results in increasing accuracy on GSM8K, MATH, and ARC-Challenge for Llama-2-70B-Chat, outperforming other Llama-2-based models not relying on additionally sourced datasets. For example, we see a large improvement from 55.6% to 81.6% on GSM8K and an accuracy of 88.7% with majority voting out of 32 samples.

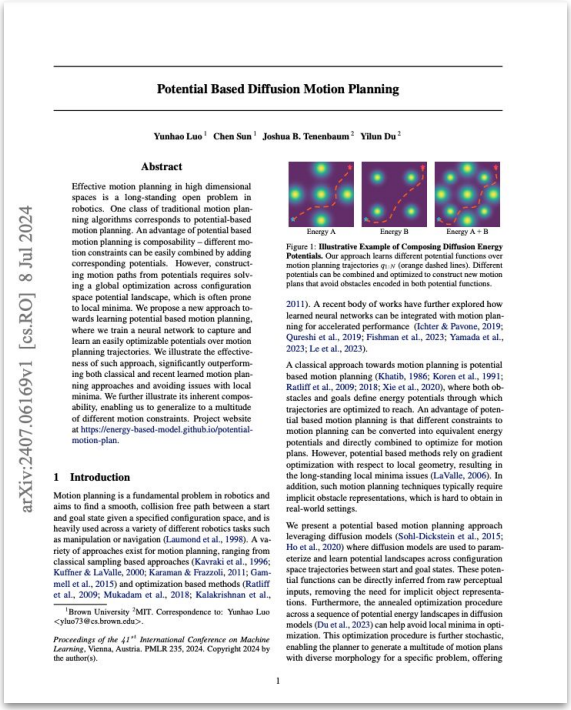
1 Introduction

Preference optimization has proven to give large gains when aligning pre-trained language models to human requirements compared to supervised fine-tuning alone [Ziegler et al., 2019, Stiennon et al., 2020]. Offline methods such as DPO [Rafailov et al., 2023] are becoming more popular for their simplicity and efficiency. Recent results have shown that iterative application of such an offline procedure is beneficial, whereby the updated model is used to construct new preference relations that are more informative, and hence improve results further. These methods include Iterative DPO [Xu et al., 2023, Xiong et al., 2023], Self-Rewarding LLMs [Yuan et al., 2024], SPIN [Chen et al., 2024], and other methods [Rosset et al., 2024]. Common to these approaches is that they have been shown to perform well on general instruction tuning tasks, but they either make only moderate gains or even decrease the performance on standard reasoning tasks. While other kinds of iterative training methods have been applied successfully to reasoning, particularly involving the iteration of supervised fine-tuning (SFT) such as STaR [Zelikman et al., 2022], ReST^{EM} [Singh et al., 2024], and V-STaR [Hosseini et al., 2024][1], using preference optimization to train the generative reasoning model is not applied in these methods.

In this work, we develop an approach to apply iterative preference optimization to reasoning tasks, with a particular focus on Chain-of-Thought (CoT) reasoning [Wu et al., 2023]. On each iteration we sample multiple chain-of-thought reasoning steps and final answers over training prompts, and then construct preference pairs such that pair winners have correct answers and pair losers have wrong answers. We then train a variant of DPO that includes a negative log-likelihood (NLL) loss term for the pair winners, which also proves crucial for performance. Given the newly trained model, we then iterate the procedure by generating new pairs, and training again, starting from the previously trained

* Equal contribution.
[1]V-STaR does use preference optimization, but for training a separate verifier model.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).



Potential Based Diffusion Motion Planning

Yunhao Luo [1] Chen Sun [1] Joshua B. Tenenbaum [2] Yilun Du [2]

Abstract

Effective motion planning in high dimensional spaces is a long-standing open problem in robotics. One class of traditional motion planning algorithms corresponds to potential-based motion planning. An advantage of potential based motion planning is composability – different motion constraints can be easily combined by adding corresponding potentials. However, constructing motion paths from potentials requires solving a global optimization across configuration space potential landscape, which is often prone to local minima. We propose a new approach towards learning potential based motion planning, where we train a neural network to capture and learn an easily optimizable potentials over motion planning trajectories. We illustrate the effectiveness of such approach, significantly outperforming both classical and recent learned motion planning approaches and avoiding issues with local minima. We further illustrate its inherent composability, enabling us to generalize to a multitude of different motion constraints. Project website at https://energy-based-model.github.io/potential-motion-plan.

1 Introduction

Motion planning is a fundamental problem in robotics and aims to find a smooth, collision free path between a start and goal state given a specified configuration space, and is heavily used across a variety of different robotics tasks such as manipulation or navigation (Laumond et al., 1998). A variety of approaches exist for motion planning, ranging from classical sampling based approaches (Kavraki et al., 1996; Kuffner & LaValle, 2000; Karaman & Frazzoli, 2011; Gammell et al., 2015) and optimization based methods (Ratliff et al., 2009; Mukadam et al., 2018; Kalakrishnan et al.,

[1]Brown University [2]MIT. Correspondence to: Yunhao Luo <yluo73@cs.brown.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Figure 1: **Illustrative Example of Composing Diffusion Energy Potentials.** Our approach learns different potential functions over motion planning trajectories $q_0 : H$ (orange dashed lines). Different potentials can be combined and optimized to construct new motion plans that avoid obstacles encoded in both potential functions.

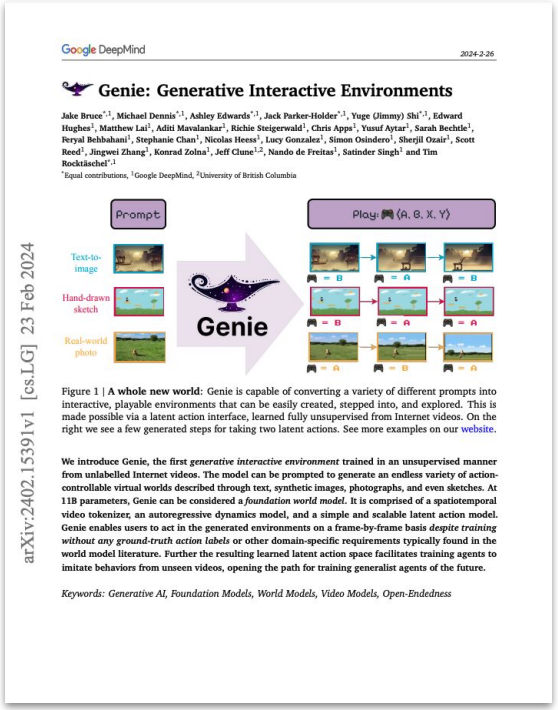
2011). A recent body of works have further explored how learned neural networks can be integrated with motion planning for accelerated performance (Ichter & Pavone, 2019; Qureshi et al., 2019; Fishman et al., 2023; Yamada et al., 2023; Le et al., 2023).

A classical approach towards motion planning is potential based motion planning (Khatib, 1986; Koren et al., 1991; Ratliff et al., 2009; 2018; Xie et al., 2020), where both obstacles and goals define energy potentials through which trajectories are optimized to reach. An advantage of potential based motion planning is that different constraints to motion planning can be converted into equivalent energy potentials and directly combined to optimize for motion plans. However, potential based methods rely on gradient optimization with respect to local geometry, resulting in the long-standing local minima issues (LaValle, 2006). In addition, such motion planning techniques typically require implicit obstacle representations, which is hard to obtain in real-world settings.

We present a potential based motion planning approach leveraging diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) where diffusion models are used to parameterize and learn potential landscapes across configuration space trajectories between start and goal states. These potential functions can be directly inferred from raw perceptual inputs, removing the need for implicit object representations. Furthermore, the annealed optimization procedure across a sequence of potential energy landscapes in diffusion models (Du et al., 2023) can help avoid local minima in optimization. This optimization procedure is further stochastic, enabling the planner to generate a multitude of motion plans with diverse morphology for a specific problem, offering

Today

After: First groups of four **paper presentations**.



Google DeepMind 2024-2-26

Genie: Generative Interactive Environments

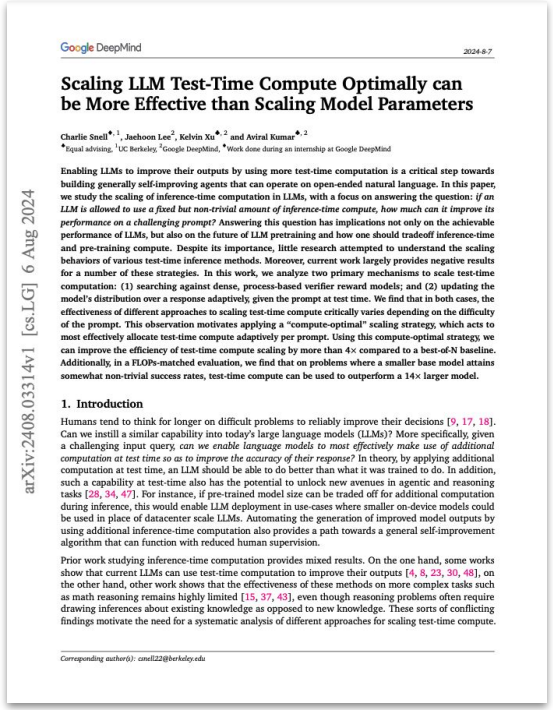
Jake Bruce*,1, Michael Dennis*,1, Ashley Edwards*,1, Jack Parker-Holder*,1, Yuge (Jimmy) Shi*,1, Edward Hughes1, Matthew Lai1, Aditi Mavalankar1, Richie Steigerwald1, Chris Apps1, Yusuf Aytar1, Sarah Bechtle1, Feryal Behbahani1, Stephanie Chan1, Nicolas Heess1, Lucy Gonzalez1, Simon Osindero1, Sherjil Ozair1, Scott Reed1, Jingwei Zhang1, Konrad Zolna1, Jeff Clune1,2, Nando de Freitas1, Satinder Singh1 and Tim Rocktäschel*,1

*Equal contributions, 1Google DeepMind, 2University of British Columbia

Figure 1 | **A whole new world**: Genie is capable of converting a variety of different prompts into interactive, playable environments that can be easily created, stepped into, and explored. This is made possible via a latent action interface, learned fully unsupervised from Internet videos. On the right we see a few generated steps for taking two latent actions. See more examples on our website.

We introduce Genie, the first *generative interactive environment* trained in an unsupervised manner from unlabelled Internet videos. The model can be prompted to generate an endless variety of action-controllable virtual worlds described through text, synthetic images, photographs, and even sketches. At 11B parameters, Genie can be considered a *foundation world model*. It is comprised of a spatiotemporal video tokenizer, an autoregressive dynamics model, and a simple and scalable latent action model. Genie enables users to act in the generated environments on a frame-by-frame basis *despite training without any ground-truth action labels* or other domain-specific requirements typically found in the world model literature. Further the resulting learned latent action space facilitates training agents to imitate behaviors from unseen videos, opening the path for training generalist agents of the future.

Keywords: Generative AI, Foundation Models, World Models, Video Models, Open-Endedness



Google DeepMind 2024-8-7

Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters

Charlie Snell♦,1, Jaehoon Lee2, Kelvin Xu♣,2 and Aviral Kumar♣,2

♦Equal advising, 1UC Berkeley, 2Google DeepMind, ♣Work done during an internship at Google DeepMind

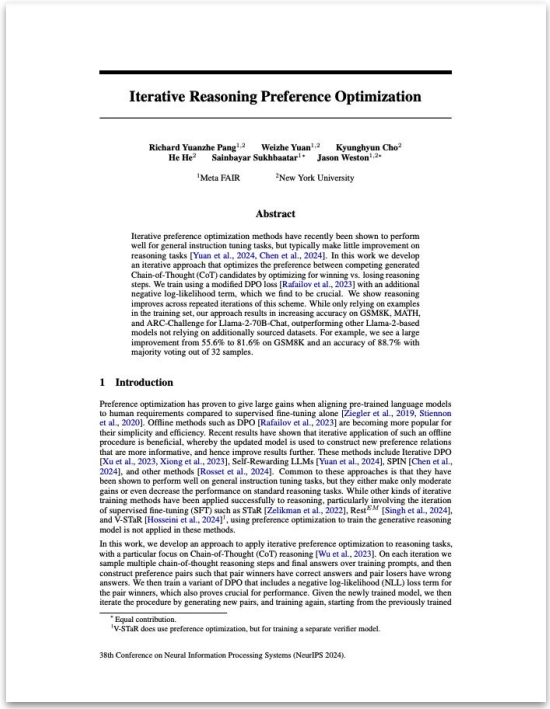
Enabling LLMs to improve their outputs by using more test-time computation is a critical step towards building generally self-improving agents that can operate on open-ended natural language. In this paper, we study the scaling of inference-time computation in LLMs, with a focus on answering the question: *if an LLM is allowed to use a fixed but non-trivial amount of inference-time compute, how much can it improve its performance on a challenging prompt?* Answering this question has implications not only on the achievable performance of LLMs, but also on the future of LLM pretraining and how one should tradeoff inference-time and pre-training compute. Despite its importance, little research attempted to understand the scaling behaviors of various test-time inference methods. Moreover, current work largely provides negative results for a number of these strategies. In this work, we analyze two primary mechanisms to scale test-time computation: (1) searching against dense, process-based verifier reward models; and (2) updating the model's distribution over a response adaptively, given the prompt at test time. We find that in both cases, the effectiveness of different approaches to scaling test-time compute critically varies depending on the difficulty of the prompt. This observation motivates applying a "compute-optimal" scaling strategy, which acts to most effectively allocate test-time compute adaptively per prompt. Using this compute-optimal strategy, we can improve the efficiency of test-time compute scaling by more than 4× compared to a best-of-N baseline. Additionally, in a FLOPs-matched evaluation, we find that on problems where a smaller base model attains somewhat non-trivial success rates, test-time compute can be used to outperform a 14× larger model.

1. Introduction

Humans tend to think for longer on difficult problems to reliably improve their decisions [9, 17, 18]. Can we instill a similar capability into today's large language models (LLMs)? More specifically, given a challenging input query, *can we enable language models to most effectively make use of additional computation at test time so as to improve the accuracy of their response?* In theory, by applying additional computation at test time, an LLM should be able to do better than what it was trained to do. In addition, such a capability at test-time also has the potential to unlock new avenues in agentic and reasoning tasks [28, 34, 47]. For instance, if pre-trained model size can be traded off for additional computation during inference, this would enable LLM deployment in use-cases where smaller on-device models could be used in place of datacenter scale LLMs. Automating the generation of improved model outputs by using additional inference-time computation also provides a path towards a general self-improvement algorithm that can function with reduced human supervision.

Prior work studying inference-time computation provides mixed results. On the one hand, some works show that current LLMs can use test-time computation to improve their outputs [4, 8, 23, 30, 48], on the other hand, other work shows that the effectiveness of these methods on more complex tasks such as math reasoning remains highly limited [15, 37, 43], even though reasoning problems often require drawing inferences about existing knowledge as opposed to new knowledge. These sorts of conflicting findings motivate the need for a systematic analysis of different approaches for scaling test-time compute.

Corresponding author(s): canell22@berkeley.edu



Iterative Reasoning Preference Optimization

Richard Yuanzhe Pang1,2 Weizhe Yuan1,2 Kyunghyun Cho2
He He2 Sainbayar Sukhbaatar1* Jason Weston1,2*

1Meta FAIR 2New York University

Abstract

Iterative preference optimization methods have recently been shown to perform well for general instruction tuning tasks, but typically make little improvement on reasoning tasks [Yuan et al., 2024, Chen et al., 2024]. In this work we develop an iterative approach that optimizes the preference between competing generated Chain-of-Thought (CoT) candidates by optimizing for winning vs. losing reasoning steps. We train using a modified DPO loss [Rafailov et al., 2023] with an additional negative log-likelihood term, which we find to be crucial. We show reasoning improves across repeated iterations of this scheme. While only relying on examples in the training set, our approach results in increasing accuracy on GSM8K, MATH, and ARC-Challenge for Llama-2-70B-Chat, outperforming other Llama-2-based models not relying on additionally sourced datasets. For example, we see a large improvement from 55.6% to 81.6% on GSM8K and an accuracy of 88.7% with majority voting out of 32 samples.

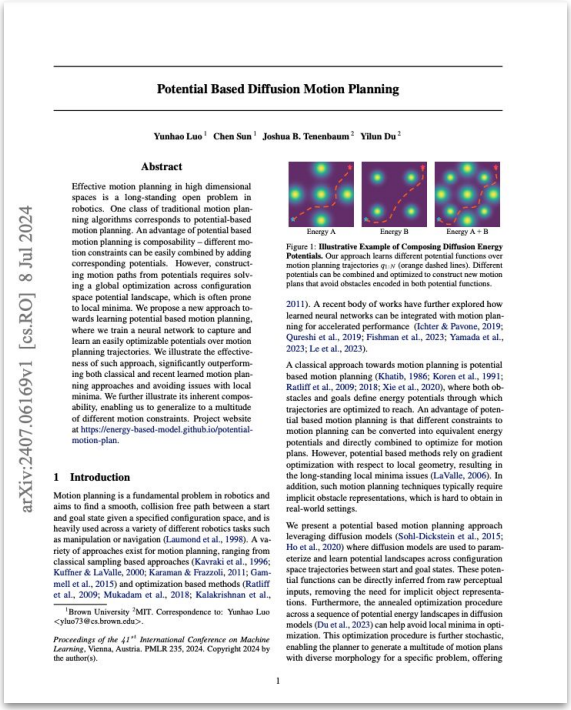
1 Introduction

Preference optimization has proven to give large gains when aligning pre-trained language models to human requirements compared to supervised fine-tuning alone [Ziegler et al., 2019, Stiennon et al., 2020]. Offline methods such as DPO [Rafailov et al., 2023] are becoming more popular for their simplicity and efficiency. Recent results have shown that iterative application of such an offline procedure is beneficial, whereby the updated model is used to construct new preference relations that are more informative, and hence improve results further. These methods include Iterative DPO [Xu et al., 2023, Xiong et al., 2023], Self-Rewarding LLMs [Yuan et al., 2024], SPIN [Chen et al., 2024], and other methods [Rosset et al., 2024]. Common to these approaches is that they have been shown to perform well on general instruction tuning tasks, but they either make only moderate gains or even decrease the performance on standard reasoning tasks. While other kinds of iterative training methods have been applied successfully to reasoning, particularly involving the iteration of supervised fine-tuning (SFT) such as STaR [Zelikman et al., 2022], ReST^{EM} [Singh et al., 2024], and V-STaR [Hosseini et al., 2024][1], using preference optimization to train the generative reasoning model is not applied in these methods.

In this work, we develop an approach to apply iterative preference optimization to reasoning tasks, with a particular focus on Chain-of-Thought (CoT) reasoning [Wu et al., 2023]. On each iteration we sample multiple chain-of-thought reasoning steps and final answers over training prompts, and then construct preference pairs such that pair winners have correct answers and pair losers have wrong answers. We then train a variant of DPO that includes a negative log-likelihood (NLL) loss term for the pair winners, which also proves crucial for performance. Given the newly trained model, we then iterate the procedure by generating new pairs, and training again, starting from the previously trained

* Equal contribution.
[1]V-STaR does use preference optimization, but for training a separate verifier model.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).



Potential Based Diffusion Motion Planning

Yunhao Luo [1] Chen Sun [1] Joshua B. Tenenbaum [2] Yilun Du [2]

Abstract

Effective motion planning in high dimensional spaces is a long-standing open problem in robotics. One class of traditional motion planning algorithms corresponds to potential-based motion planning. An advantage of potential based motion planning is composability – different motion constraints can be easily combined by adding corresponding potentials. However, constructing motion paths from potentials requires solving a global optimization across configuration space potential landscape, which is often prone to local minima. We propose a new approach towards learning potential based motion planning, where we train a neural network to capture and learn an easily optimizable potentials over motion planning trajectories. We illustrate the effectiveness of such approach, significantly outperforming both classical and recent learned motion planning approaches and avoiding issues with local minima. We further illustrate its inherent composability, enabling us to generalize to a multitude of different motion constraints. Project website at https://energy-based-model.github.io/potential-motion-plan.

1 Introduction

Motion planning is a fundamental problem in robotics and aims to find a smooth, collision free path between a start and goal state given a specified configuration space, and is heavily used across a variety of different robotics tasks such as manipulation or navigation (Laumond et al., 1998). A variety of approaches exist for motion planning, ranging from classical sampling based approaches (Kavraki et al., 1996; Kuffner & LaValle, 2000; Karaman & Frazzoli, 2011; Gammell et al., 2015) and optimization based methods (Ratliff et al., 2009; Mukadam et al., 2018; Kalakrishnan et al., 2011). A recent body of works have further explored how learned neural networks can be integrated with motion planning for accelerated performance (Ichter & Pavone, 2019; Qureshi et al., 2019; Fishman et al., 2023; Yamada et al., 2023; Le et al., 2023).

Figure 1: **Illustrative Example of Composing Diffusion Energy Potentials.** Our approach learns different potential functions over motion planning trajectories $q_0 : H$ (orange dashed lines). Different potentials can be combined and optimized to construct new motion plans that avoid obstacles encoded in both potential functions.

A classical approach towards motion planning is potential based motion planning (Khatib, 1986; Koren et al., 1991; Ratliff et al., 2009; 2018; Xie et al., 2020), where both obstacles and goals define energy potentials through which trajectories are optimized to reach. An advantage of potential based motion planning is that different constraints to motion planning can be converted into equivalent energy potentials and directly combined to optimize for motion plans. However, potential based methods rely on gradient optimization with respect to local geometry, resulting in the long-standing local minima issues (LaValle, 2006). In addition, such motion planning techniques typically require implicit obstacle representations, which is hard to obtain in real-world settings.

We present a potential based motion planning approach leveraging diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) where diffusion models are used to parameterize and learn potential landscapes across configuration space trajectories between start and goal states. These potential functions can be directly inferred from raw perceptual inputs, removing the need for implicit object representations. Furthermore, the annealed optimization procedure across a sequence of potential energy landscapes in diffusion models (Du et al., 2023) can help avoid local minima in optimization. This optimization procedure is further stochastic, enabling the planner to generate a multitude of motion plans with diverse morphology for a specific problem, offering

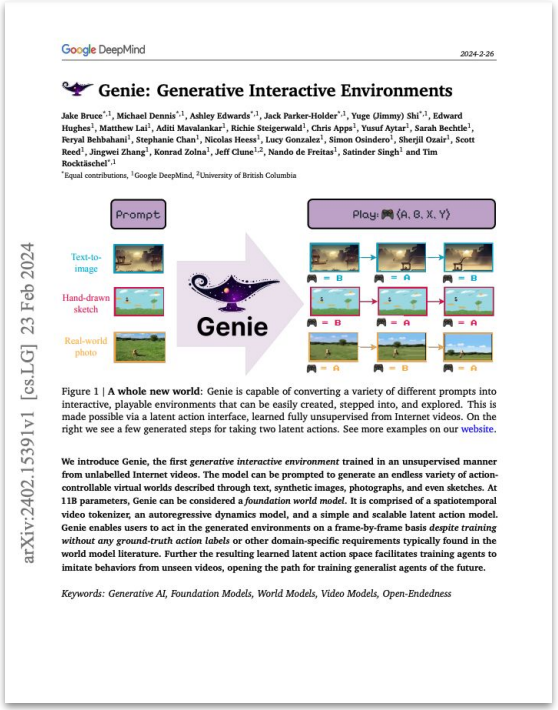
[1]Brown University [2]MIT. Correspondence to: Yunhao Luo <yluo73@cs.brown.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Generative Interactive Environments

Today

After: First groups of four **paper presentations**.



Google DeepMind 2024-2-26

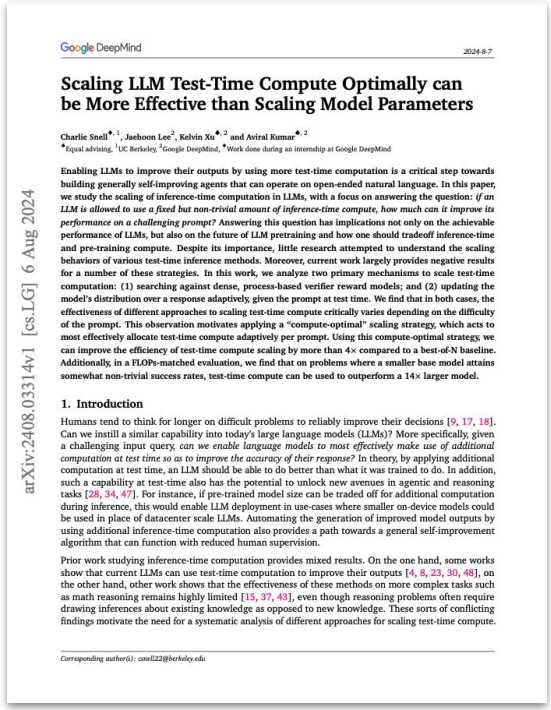
Genie: Generative Interactive Environments

Jake Bruce*,1, Michael Dennis*,1, Ashley Edwards*,1, Jack Parker-Holder*,1, Yuge (Jimmy) Shi*,1, Edward Hughes1, Matthew Lai1, Aditi Mavalankar1, Richie Steigerwald1, Chris Apps1, Yusuf Aytar1, Sarah Bechtle1, Feryal Behbahani1, Stephanie Chan1, Nicolas Heess1, Lucy Gonzalez1, Simon Osindero1, Sherjil Ozair1, Scott Reed1, Jingwei Zhang1, Konrad Zolna1, Jeff Clune1,2, Nando de Freitas1, Satinder Singh1 and Tim Rocktäschel*,1
*Equal contributions, 1Google DeepMind, 2University of British Columbia

Figure 1 | **A whole new world**: Genie is capable of converting a variety of different prompts into interactive, playable environments that can be easily created, stepped into, and explored. This is made possible via a latent action interface, learned fully unsupervised from Internet videos. On the right we see a few generated steps for taking two latent actions. See more examples on our website.

We introduce Genie, the first *generative interactive environment* trained in an unsupervised manner from unlabelled Internet videos. The model can be prompted to generate an endless variety of action-controllable virtual worlds described through text, synthetic images, photographs, and even sketches. At 11B parameters, Genie can be considered a *foundation world model*. It is comprised of a spatiotemporal video tokenizer, an autoregressive dynamics model, and a simple and scalable latent action model. Genie enables users to act in the generated environments on a frame-by-frame basis *despite training without any ground-truth action labels* or other domain-specific requirements typically found in the world model literature. Further the resulting learned latent action space facilitates training agents to imitate behaviors from unseen videos, opening the path for training generalist agents of the future.

Keywords: Generative AI, Foundation Models, World Models, Video Models, Open-Endedness



Google DeepMind 2024-8-7

Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters

Charlie Snell*,1, Jaehoon Lee2, Kelvin Xu*,2 and Aviral Kumar*,2
*Equal advising, 1UC Berkeley, 2Google DeepMind, *Work done during an internship at Google DeepMind

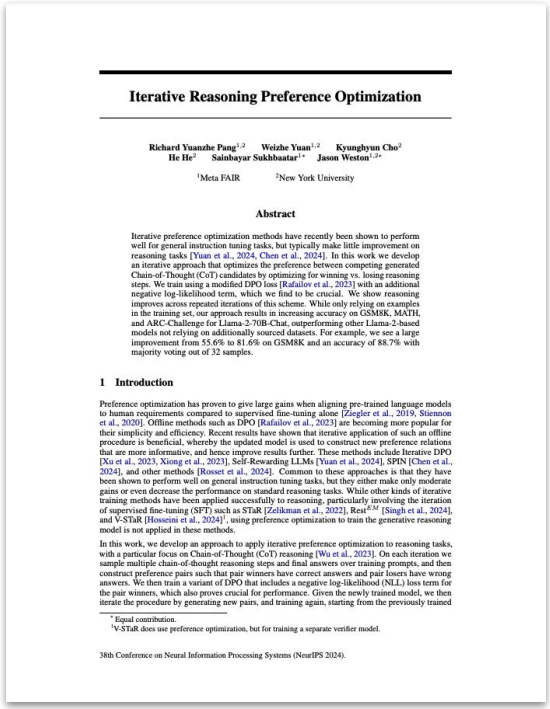
Enabling LLMs to improve their outputs by using more test-time computation is a critical step towards building generally self-improving agents that can operate on open-ended natural language. In this paper, we study the scaling of inference-time computation in LLMs, with a focus on answering the question: *if an LLM is allowed to use a fixed but non-trivial amount of inference-time compute, how much can it improve its performance on a challenging prompt?* Answering this question has implications not only on the achievable performance of LLMs, but also on the future of LLM pretraining and how one should tradeoff inference-time and pre-training compute. Despite its importance, little research attempted to understand the scaling behaviors of various test-time inference methods. Moreover, current work largely provides negative results for a number of these strategies. In this work, we analyze two primary mechanisms to scale test-time computation: (1) searching against dense, process-based verifier reward models; and (2) updating the model's distribution over a response adaptively, given the prompt at test time. We find that in both cases, the effectiveness of different approaches to scaling test-time compute critically varies depending on the difficulty of the prompt. This observation motivates applying a "compute-optimal" scaling strategy, which acts to most effectively allocate test-time compute adaptively per prompt. Using this compute-optimal strategy, we can improve the efficiency of test-time compute scaling by more than 4× compared to a best-of-N baseline. Additionally, in a FLOPs-matched evaluation, we find that on problems where a smaller base model attains somewhat non-trivial success rates, test-time compute can be used to outperform a 14× larger model.

1. Introduction

Humans tend to think for longer on difficult problems to reliably improve their decisions [9, 17, 18]. Can we instill a similar capability into today's large language models (LLMs)? More specifically, given a challenging input query, *can we enable language models to most effectively make use of additional computation at test time so as to improve the accuracy of their response?* In theory, by applying additional computation at test time, an LLM should be able to do better than what it was trained to do. In addition, such a capability at test-time also has the potential to unlock new avenues in agentic and reasoning tasks [28, 34, 47]. For instance, if pre-trained model size can be traded off for additional computation during inference, this would enable LLM deployment in use-cases where smaller on-device models could be used in place of datacenter scale LLMs. Automating the generation of improved model outputs by using additional inference-time computation also provides a path towards a general self-improvement algorithm that can function with reduced human supervision.

Prior work studying inference-time computation provides mixed results. On the one hand, some works show that current LLMs can use test-time computation to improve their outputs [4, 8, 23, 30, 48], on the other hand, other work shows that the effectiveness of these methods on more complex tasks such as math reasoning remains highly limited [15, 37, 43], even though reasoning problems often require drawing inferences about existing knowledge as opposed to new knowledge. These sorts of conflicting findings motivate the need for a systematic analysis of different approaches for scaling test-time compute.

Corresponding author(s): cansell22@berkeley.edu



Iterative Reasoning Preference Optimization

Richard Yuanzhe Pang1,2 Weizhe Yuan1,2 Kyunghyun Cho2
He He2 Sainbayar Sukhbaatar1* Jason Weston1,2*
1Meta FAIR 2New York University

Abstract

Iterative preference optimization methods have recently been shown to perform well for general instruction tuning tasks, but typically make little improvement on reasoning tasks [Yuan et al., 2024, Chen et al., 2024]. In this work we develop an iterative approach that optimizes the preference between competing generated Chain-of-Thought (CoT) candidates by optimizing for winning vs. losing reasoning steps. We train using a modified DPO loss [Rafailov et al., 2023] with an additional negative log-likelihood term, which we find to be crucial. We show reasoning improves across repeated iterations of this scheme. While only relying on examples in the training set, our approach results in increasing accuracy on GSM8K, MATH, and ARC-Challenge for Llama-2-70B-Chat, outperforming other Llama-2-based models not relying on additionally sourced datasets. For example, we see a large improvement from 55.6% to 81.6% on GSM8K and an accuracy of 88.7% with majority voting out of 32 samples.

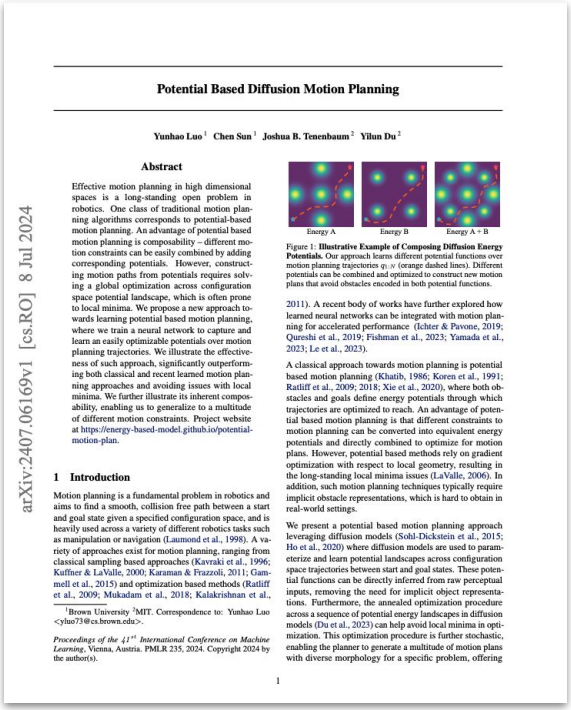
1 Introduction

Preference optimization has proven to give large gains when aligning pre-trained language models to human requirements compared to supervised fine-tuning alone [Ziegler et al., 2019, Stiennon et al., 2020]. Offline methods such as DPO [Rafailov et al., 2023] are becoming more popular for their simplicity and efficiency. Recent results have shown that iterative application of such an offline procedure is beneficial, whereby the updated model is used to construct new preference relations that are more informative, and hence improve results further. These methods include Iterative DPO [Xu et al., 2023, Xiong et al., 2023], Self-Rewarding LLMs [Yuan et al., 2024], SPIN [Chen et al., 2024], and other methods [Rosset et al., 2024]. Common to these approaches is that they have been shown to perform well on general instruction tuning tasks, but they either make only moderate gains or even decrease the performance on standard reasoning tasks. While other kinds of iterative training methods have been applied successfully to reasoning, particularly involving the iteration of supervised fine-tuning (SFT) such as STaR [Zelikman et al., 2022], ReST^{EM} [Singh et al., 2024], and V-STaR [Hosseini et al., 2024][1], using preference optimization to train the generative reasoning model is not applied in these methods.

In this work, we develop an approach to apply iterative preference optimization to reasoning tasks, with a particular focus on Chain-of-Thought (CoT) reasoning [Wu et al., 2023]. On each iteration we sample multiple chain-of-thought reasoning steps and final answers over training prompts, and then construct preference pairs such that pair winners have correct answers and pair losers have wrong answers. We then train a variant of DPO that includes a negative log-likelihood (NLL) loss term for the pair winners, which also proves crucial for performance. Given the newly trained model, we then iterate the procedure by generating new pairs, and training again, starting from the previously trained

* Equal contribution.
[1]V-STaR does use preference optimization, but for training a separate verifier model.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).



Potential Based Diffusion Motion Planning

Yunhao Luo [1] Chen Sun [1] Joshua B. Tenenbaum [2] Yilun Du [2]

Abstract

Effective motion planning in high dimensional spaces is a long-standing open problem in robotics. One class of traditional motion planning algorithms corresponds to potential-based motion planning. An advantage of potential based motion planning is composability – different motion constraints can be easily combined by adding corresponding potentials. However, constructing motion paths from potentials requires solving a global optimization across configuration space potential landscape, which is often prone to local minima. We propose a new approach towards learning potential based motion planning, where we train a neural network to capture and learn an easily optimizable potentials over motion planning trajectories. We illustrate the effectiveness of such approach, significantly outperforming classical and recent learned motion planning approaches and avoiding issues with local minima. We further illustrate its inherent composability, enabling us to generalize to a multitude of different motion constraints. Project website at https://energy-based-model.github.io/potential-motion-plan.

1 Introduction

Motion planning is a fundamental problem in robotics and aims to find a smooth, collision free path between a start and goal state given a specified configuration space, and is heavily used across a variety of different robotics tasks such as manipulation or navigation (Laumond et al., 1998). A variety of approaches exist for motion planning, ranging from classical sampling based approaches (Kavraki et al., 1996; Kuffner & LaValle, 2000; Karaman & Frazzoli, 2011; Gammell et al., 2015) and optimization based methods (Ratliff et al., 2009; Mukadam et al., 2018; Kalakrishnan et al., 2011). A recent body of works have further explored how learned neural networks can be integrated with motion planning for accelerated performance (Ichter & Pavone, 2019; Qureshi et al., 2019; Fishman et al., 2023; Yamada et al., 2023; Le et al., 2023).

Figure 1: **Illustrative Example of Composing Diffusion Energy Potentials.** Our approach learns different potential functions over motion planning trajectories $q_0 : H$ (orange dashed lines). Different potentials can be combined and optimized to construct new motion plans that avoid obstacles encoded in both potential functions.

A classical approach towards motion planning is potential based motion planning (Khatib, 1986; Koren et al., 1991; Ratliff et al., 2009; 2018; Xie et al., 2020), where both obstacles and goals define energy potentials through which trajectories are optimized to reach. An advantage of potential based motion planning is that different constraints to motion planning can be converted into equivalent energy potentials and directly combined to optimize for motion plans. However, potential based methods rely on gradient optimization with respect to local geometry, resulting in the long-standing local minima issues (LaValle, 2006). In addition, such motion planning techniques typically require implicit obstacle representations, which is hard to obtain in real-world settings.

We present a potential based motion planning approach leveraging diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) where diffusion models are used to parameterize and learn potential landscapes across configuration space trajectories between start and goal states. These potential functions can be directly inferred from raw perceptual inputs, removing the need for implicit object representations. Furthermore, the annealed optimization procedure across a sequence of potential energy landscapes in diffusion models (Du et al., 2023) can help avoid local minima in optimization. This optimization procedure is further stochastic, enabling the planner to generate a multitude of motion plans with diverse morphology for a specific problem, offering

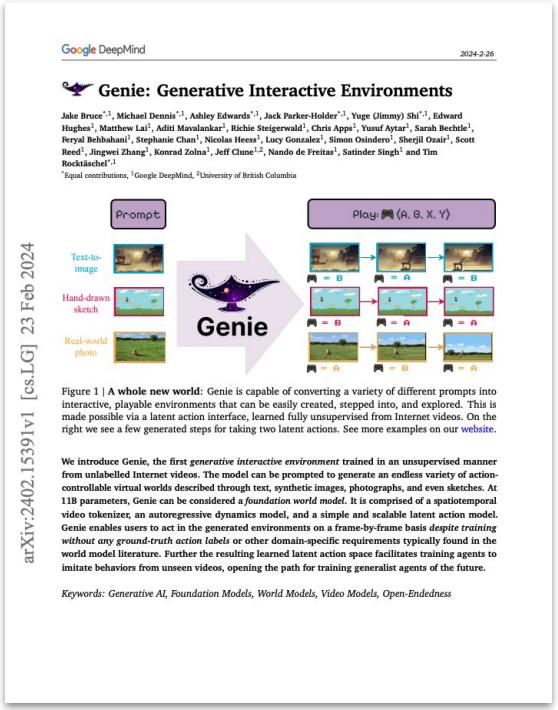
[1]Brown University [2]MIT. Correspondence to: Yunhao Luo <yluo73@cs.brown.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Scaling LLM Test-Time Compute

Today

After: First groups of four **paper presentations**.



Google DeepMind 2024-2-26

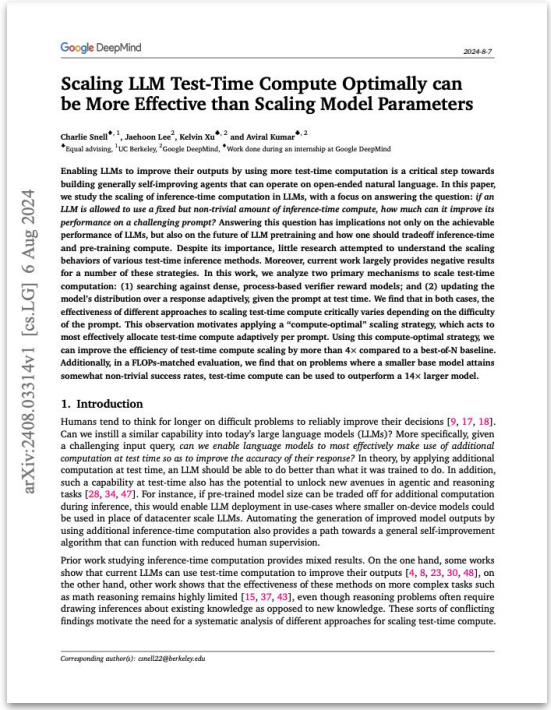
Genie: Generative Interactive Environments

Jake Bruce*,1, Michael Dennis*,1, Ashley Edwards*,1, Jack Parker-Holder*,1, Yuge (Jimmy) Shi*,1, Edward Hughes1, Matthew Lai1, Aditi Mavalankar1, Richie Steigerwald1, Chris Apps1, Yusuf Aytar1, Sarah Bechtle1, Feryal Behbahani1, Stephanie Chan1, Nicolas Heess1, Lucy Gonzalez1, Simon Osindero1, Sherjil Ozair1, Scott Reed1, Jingwei Zhang1, Konrad Zolna1, Jeff Clune1,2, Nando de Freitas1, Satinder Singh1 and Tim Rocktäschel*,1
*Equal contributions, 1Google DeepMind, 2University of British Columbia

Figure 1 | **A whole new world**: Genie is capable of converting a variety of different prompts into interactive, playable environments that can be easily created, stepped into, and explored. This is made possible via a latent action interface, learned fully unsupervised from Internet videos. On the right we see a few generated steps for taking two latent actions. See more examples on our website.

We introduce Genie, the first *generative interactive environment* trained in an unsupervised manner from unlabelled Internet videos. The model can be prompted to generate an endless variety of action-controllable virtual worlds described through text, synthetic images, photographs, and even sketches. At 11B parameters, Genie can be considered a *foundation world model*. It is comprised of a spatiotemporal video tokenizer, an autoregressive dynamics model, and a simple and scalable latent action model. Genie enables users to act in the generated environments on a frame-by-frame basis *despite training without any ground-truth action labels* or other domain-specific requirements typically found in the world model literature. Further the resulting learned latent action space facilitates training agents to imitate behaviors from unseen videos, opening the path for training generalist agents of the future.

Keywords: Generative AI, Foundation Models, World Models, Video Models, Open-Endedness



Google DeepMind 2024-8-7

Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters

Charlie Snell♦,1, Jaehoon Lee2, Kelvin Xu♦,2 and Aviral Kumar♦,2
♦Equal advising, 1UC Berkeley, 2Google DeepMind, ♦Work done during an internship at Google DeepMind

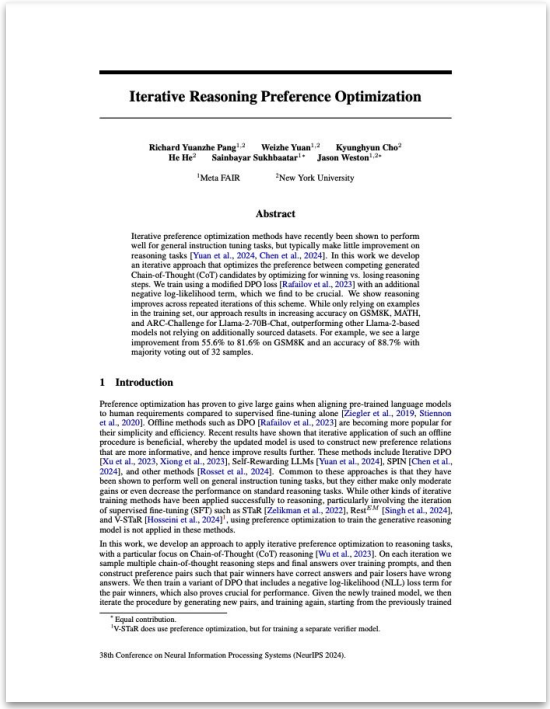
Enabling LLMs to improve their outputs by using more test-time computation is a critical step towards building generally self-improving agents that can operate on open-ended natural language. In this paper, we study the scaling of inference-time computation in LLMs, with a focus on answering the question: *if an LLM is allowed to use a fixed but non-trivial amount of inference-time compute, how much can it improve its performance on a challenging prompt?* Answering this question has implications not only on the achievable performance of LLMs, but also on the future of LLM pretraining and how one should tradeoff inference-time and pre-training compute. Despite its importance, little research attempted to understand the scaling behaviors of various test-time inference methods. Moreover, current work largely provides negative results for a number of these strategies. In this work, we analyze two primary mechanisms to scale test-time computation: (1) searching against dense, process-based verifier reward models; and (2) updating the model's distribution over a response adaptively, given the prompt at test time. We find that in both cases, the effectiveness of different approaches to scaling test-time compute critically varies depending on the difficulty of the prompt. This observation motivates applying a "compute-optimal" scaling strategy, which acts to most effectively allocate test-time compute adaptively per prompt. Using this compute-optimal strategy, we can improve the efficiency of test-time compute scaling by more than 4× compared to a best-of-N baseline. Additionally, in a FLOPs-matched evaluation, we find that on problems where a smaller base model attains somewhat non-trivial success rates, test-time compute can be used to outperform a 14× larger model.

1. Introduction

Humans tend to think for longer on difficult problems to reliably improve their decisions [9, 17, 18]. Can we instill a similar capability into today's large language models (LLMs)? More specifically, given a challenging input query, *can we enable language models to most effectively make use of additional computation at test time so as to improve the accuracy of their response?* In theory, by applying additional computation at test time, an LLM should be able to do better than what it was trained to do. In addition, such a capability at test-time also has the potential to unlock new avenues in agentic and reasoning tasks [28, 34, 47]. For instance, if pre-trained model size can be traded off for additional computation during inference, this would enable LLM deployment in use-cases where smaller on-device models could be used in place of datacenter scale LLMs. Automating the generation of improved model outputs by using additional inference-time computation also provides a path towards a general self-improvement algorithm that can function with reduced human supervision.

Prior work studying inference-time computation provides mixed results. On the one hand, some works show that current LLMs can use test-time computation to improve their outputs [4, 8, 23, 30, 48], on the other hand, other work shows that the effectiveness of these methods on more complex tasks such as math reasoning remains highly limited [15, 37, 43], even though reasoning problems often require drawing inferences about existing knowledge as opposed to new knowledge. These sorts of conflicting findings motivate the need for a systematic analysis of different approaches for scaling test-time compute.

Corresponding author(s): canell22@berkeley.edu



Iterative Reasoning Preference Optimization

Richard Yuanzhe Pang1,2 Weizhe Yuan1,2 Kyunghyun Cho2
He He2 Sainbayar Sukhbaatar1* Jason Weston1,2*
1Meta FAIR 2New York University

Abstract

Iterative preference optimization methods have recently been shown to perform well for general instruction tuning tasks, but typically make little improvement on reasoning tasks [Yuan et al., 2024, Chen et al., 2024]. In this work we develop an iterative approach that optimizes the preference between competing generated Chain-of-Thought (CoT) candidates by optimizing for winning vs. losing reasoning steps. We train using a modified DPO loss [Rafailov et al., 2023] with an additional negative log-likelihood term, which we find to be crucial. We show reasoning improves across repeated iterations of this scheme. While only relying on examples in the training set, our approach results in increasing accuracy on GSM8K, MATH, and ARC-Challenge for Llama-2-70B-Chat, outperforming other Llama-2-based models not relying on additionally sourced datasets. For example, we see a large improvement from 55.6% to 81.6% on GSM8K and an accuracy of 88.7% with majority voting out of 32 samples.

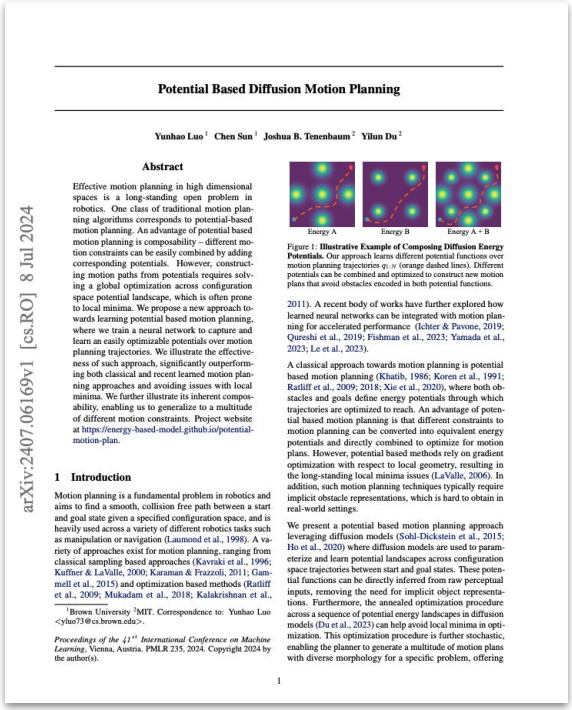
1 Introduction

Preference optimization has proven to give large gains when aligning pre-trained language models to human requirements compared to supervised fine-tuning alone [Ziegler et al., 2019, Stiennon et al., 2020]. Offline methods such as DPO [Rafailov et al., 2023] are becoming more popular for their simplicity and efficiency. Recent results have shown that iterative application of such an offline procedure is beneficial, whereby the updated model is used to construct new preference relations that are more informative, and hence improve results further. These methods include Iterative DPO [Xu et al., 2023, Xiong et al., 2023], Self-Rewarding LLMs [Yuan et al., 2024], SPIN [Chen et al., 2024], and other methods [Rosset et al., 2024]. Common to these approaches is that they have been shown to perform well on general instruction tuning tasks, but they either make only moderate gains or even decrease the performance on standard reasoning tasks. While other kinds of iterative training methods have been applied successfully to reasoning, particularly involving the iteration of supervised fine-tuning (SFT) such as STaR [Zelikman et al., 2022], ReST^{EM} [Singh et al., 2024], and V-STaR [Hosseini et al., 2024][1], using preference optimization to train the generative reasoning model is not applied in these methods.

In this work, we develop an approach to apply iterative preference optimization to reasoning tasks, with a particular focus on Chain-of-Thought (CoT) reasoning [Wu et al., 2023]. On each iteration we sample multiple chain-of-thought reasoning steps and final answers over training prompts, and then construct preference pairs such that pair winners have correct answers and pair losers have wrong answers. We then train a variant of DPO that includes a negative log-likelihood (NLL) loss term for the pair winners, which also proves crucial for performance. Given the newly trained model, we then iterate the procedure by generating new pairs, and training again, starting from the previously trained

* Equal contribution.
[1]V-STaR does use preference optimization, but for training a separate verifier model.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).



Potential Based Diffusion Motion Planning

Yunhao Luo[1] Chen Sun[1] Joshua B. Tenenbaum[2] Yilun Du[2]

Abstract

Effective motion planning in high dimensional spaces is a long-standing open problem in robotics. One class of traditional motion planning algorithms corresponds to potential-based motion planning. An advantage of potential based motion planning is composability – different motion constraints can be easily combined by adding corresponding potentials. However, constructing motion paths from potentials requires solving a global optimization across configuration space potential landscape, which is often prone to local minima. We propose a new approach towards learning potential based motion planning, where we train a neural network to capture and learn an easily optimizable potentials over motion planning trajectories. We illustrate the effectiveness of such approach, significantly outperforming both classical and recent learned motion planning approaches and avoiding issues with local minima. We further illustrate its inherent composability, enabling us to generalize to a multitude of different motion constraints. Project website at https://energy-based-model.github.io/potential-motion-plan.

1 Introduction

Motion planning is a fundamental problem in robotics and aims to find a smooth, collision free path between a start and goal state given a specified configuration space, and is heavily used across a variety of different robotics tasks such as manipulation or navigation (Laumond et al., 1998). A variety of approaches exist for motion planning, ranging from classical sampling based approaches (Kavraki et al., 1996; Kuffner & LaValle, 2000; Karaman & Frazzoli, 2011; Gammell et al., 2015) and optimization based methods (Ratliff et al., 2009; Mukadam et al., 2018; Kalakrishnan et al., 2011). A recent body of works have further explored how learned neural networks can be integrated with motion planning for accelerated performance (Ichter & Pavone, 2019; Qureshi et al., 2019; Fishman et al., 2023; Yamada et al., 2023; Le et al., 2023).

A classical approach towards motion planning is potential based motion planning (Khatib, 1986; Koren et al., 1991; Ratliff et al., 2009; 2018; Xie et al., 2020), where both obstacles and goals define energy potentials through which trajectories are optimized to reach. An advantage of potential based motion planning is that different constraints to motion planning can be converted into equivalent energy potentials and directly combined to optimize for motion plans. However, potential based methods rely on gradient optimization with respect to local geometry, resulting in the long-standing local minima issues (LaValle, 2006). In addition, such motion planning techniques typically require implicit obstacle representations, which is hard to obtain in real-world settings.

We present a potential based motion planning approach leveraging diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) where diffusion models are used to parameterize and learn potential landscapes across configuration space trajectories between start and goal states. These potential functions can be directly inferred from raw perceptual inputs, removing the need for implicit object representations. Furthermore, the annealed optimization procedure across a sequence of potential energy landscapes in diffusion models (Du et al., 2023) can help avoid local minima in optimization. This optimization procedure is further stochastic, enabling the planner to generate a multitude of motion plans with diverse morphology for a specific problem, offering

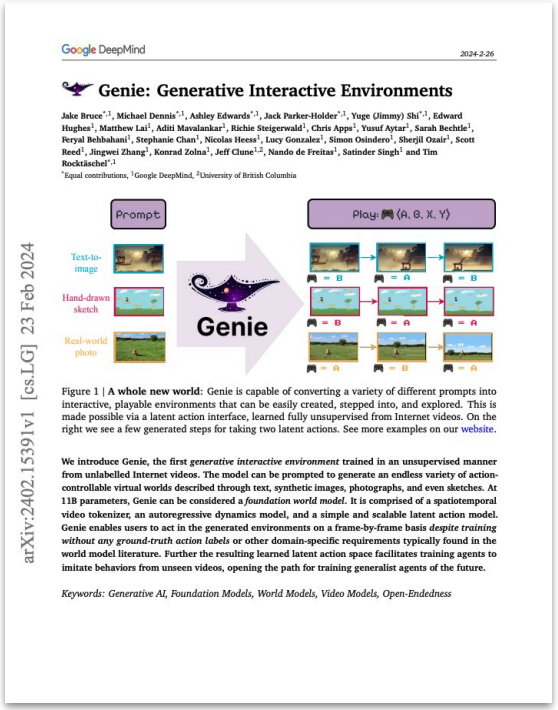
[1]Brown University [2]MIT. Correspondence to: Yunhao Luo <yluo73@cs.brown.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Iterative Preference Optimization

Today

After: First groups of four **paper presentations**.



Google DeepMind 2024-2-26

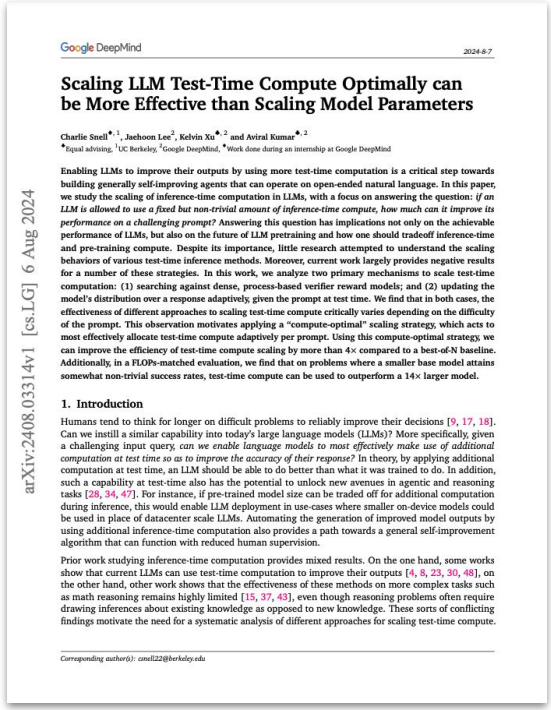
Genie: Generative Interactive Environments

Jake Bruce*,1, Michael Dennis*,1, Ashley Edwards*,1, Jack Parker-Holder*,1, Yuge (Jimmy) Shi*,1, Edward Hughes1, Matthew Lai1, Aditi Mavalankar1, Richie Steigerwald1, Chris Apps1, Yusuf Aytar1, Sarah Bechtle1, Feryal Behbahani1, Stephanie Chan1, Nicolas Heess1, Lucy Gonzalez1, Simon Osindero1, Sherjil Ozair1, Scott Reed1, Jingwei Zhang1, Konrad Zolna1, Jeff Clune1,2, Nando de Freitas1, Satinder Singh1 and Tim Rocktäschel*,1
*Equal contributions, 1Google DeepMind, 2University of British Columbia

Figure 1 | **A whole new world**: Genie is capable of converting a variety of different prompts into interactive, playable environments that can be easily created, stepped into, and explored. This is made possible via a latent action interface, learned fully unsupervised from Internet videos. On the right we see a few generated steps for taking two latent actions. See more examples on our website.

We introduce Genie, the first *generative interactive environment* trained in an unsupervised manner from unlabelled Internet videos. The model can be prompted to generate an endless variety of action-controllable virtual worlds described through text, synthetic images, photographs, and even sketches. At 11B parameters, Genie can be considered a *foundation world model*. It is comprised of a spatiotemporal video tokenizer, an autoregressive dynamics model, and a simple and scalable latent action model. Genie enables users to act in the generated environments on a frame-by-frame basis *despite training without any ground-truth action labels* or other domain-specific requirements typically found in the world model literature. Further the resulting learned latent action space facilitates training agents to imitate behaviors from unseen videos, opening the path for training generalist agents of the future.

Keywords: Generative AI, Foundation Models, World Models, Video Models, Open-Endedness



Google DeepMind 2024-8-7

Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters

Charlie Snell♠,1, Jaehoon Lee2, Kelvin Xu♠,2 and Aviral Kumar♠,2
♠Equal advising, 1UC Berkeley, 2Google DeepMind, ♠Work done during an internship at Google DeepMind

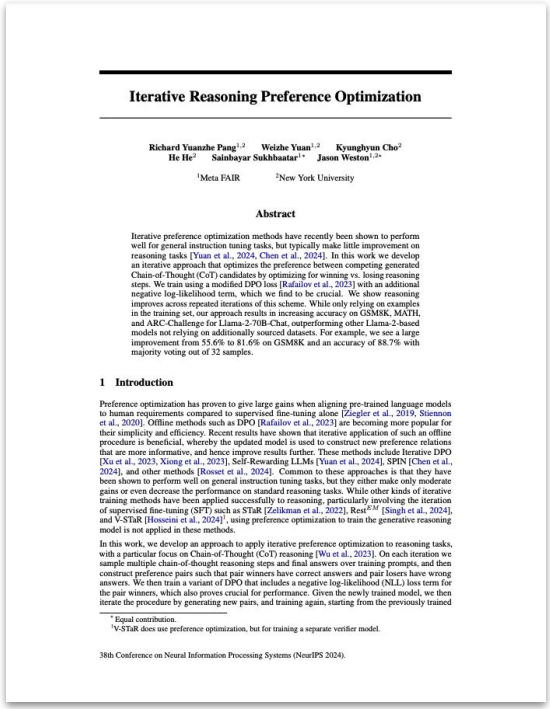
Enabling LLMs to improve their outputs by using more test-time computation is a critical step towards building generally self-improving agents that can operate on open-ended natural language. In this paper, we study the scaling of inference-time computation in LLMs, with a focus on answering the question: *if an LLM is allowed to use a fixed but non-trivial amount of inference-time compute, how much can it improve its performance on a challenging prompt?* Answering this question has implications not only on the achievable performance of LLMs, but also on the future of LLM pretraining and how one should tradeoff inference-time and pre-training compute. Despite its importance, little research attempted to understand the scaling behaviors of various test-time inference methods. Moreover, current work largely provides negative results for a number of these strategies. In this work, we analyze two primary mechanisms to scale test-time computation: (1) searching against dense, process-based verifier reward models; and (2) updating the model's distribution over a response adaptively, given the prompt at test time. We find that in both cases, the effectiveness of different approaches to scaling test-time compute critically varies depending on the difficulty of the prompt. This observation motivates applying a "compute-optimal" scaling strategy, which acts to most effectively allocate test-time compute adaptively per prompt. Using this compute-optimal strategy, we can improve the efficiency of test-time compute scaling by more than 4× compared to a best-of-N baseline. Additionally, in a FLOPs-matched evaluation, we find that on problems where a smaller base model attains somewhat non-trivial success rates, test-time compute can be used to outperform a 14× larger model.

1. Introduction

Humans tend to think for longer on difficult problems to reliably improve their decisions [9, 17, 18]. Can we instill a similar capability into today's large language models (LLMs)? More specifically, given a challenging input query, *can we enable language models to most effectively make use of additional computation at test time so as to improve the accuracy of their response?* In theory, by applying additional computation at test time, an LLM should be able to do better than what it was trained to do. In addition, such a capability at test-time also has the potential to unlock new avenues in agentic and reasoning tasks [28, 34, 47]. For instance, if pre-trained model size can be traded off for additional computation during inference, this would enable LLM deployment in use-cases where smaller on-device models could be used in place of datacenter scale LLMs. Automating the generation of improved model outputs by using additional inference-time computation also provides a path towards a general self-improvement algorithm that can function with reduced human supervision.

Prior work studying inference-time computation provides mixed results. On the one hand, some works show that current LLMs can use test-time computation to improve their outputs [4, 8, 23, 30, 48], on the other hand, other work shows that the effectiveness of these methods on more complex tasks such as math reasoning remains highly limited [15, 37, 43], even though reasoning problems often require drawing inferences about existing knowledge as opposed to new knowledge. These sorts of conflicting findings motivate the need for a systematic analysis of different approaches for scaling test-time compute.

Corresponding author(s): canell22@berkeley.edu



Iterative Reasoning Preference Optimization

Richard Yuanzhe Pang1,2 Weizhe Yuan1,2 Kyunghyun Cho2
He He2 Sainbayar Sukhbaatar1* Jason Weston1,2*
1Meta FAIR 2New York University

Abstract

Iterative preference optimization methods have recently been shown to perform well for general instruction tuning tasks, but typically make little improvement on reasoning tasks [Yuan et al., 2024, Chen et al., 2024]. In this work we develop an iterative approach that optimizes the preference between competing generated Chain-of-Thought (CoT) candidates by optimizing for winning vs. losing reasoning steps. We train using a modified DPO loss [Rafailov et al., 2023] with an additional negative log-likelihood term, which we find to be crucial. We show reasoning improves across repeated iterations of this scheme. While only relying on examples in the training set, our approach results in increasing accuracy on GSM8K, MATH, and ARC-Challenge for Llama-2-70B-Chat, outperforming other Llama-2-based models not relying on additionally sourced datasets. For example, we see a large improvement from 55.6% to 81.6% on GSM8K and an accuracy of 88.7% with majority voting out of 32 samples.

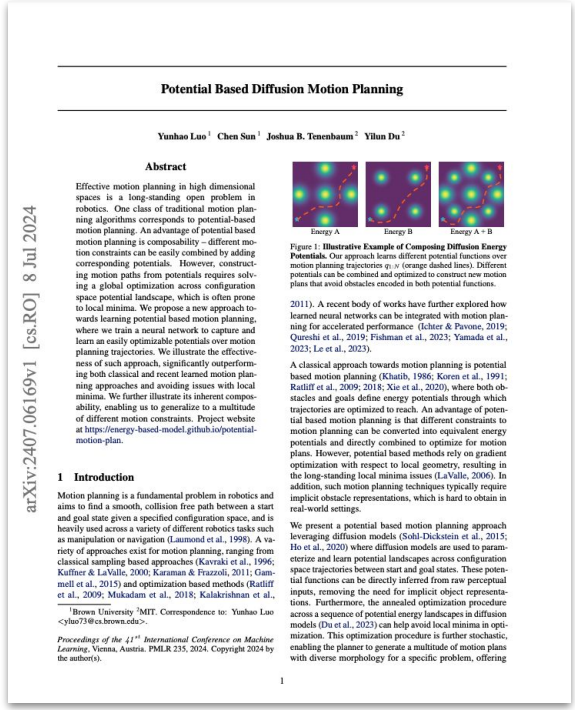
1 Introduction

Preference optimization has proven to give large gains when aligning pre-trained language models to human requirements compared to supervised fine-tuning alone [Ziegler et al., 2019, Stiennon et al., 2020]. Offline methods such as DPO [Rafailov et al., 2023] are becoming more popular for their simplicity and efficiency. Recent results have shown that iterative application of such an offline procedure is beneficial, whereby the updated model is used to construct new preference relations that are more informative, and hence improve results further. These methods include Iterative DPO [Xu et al., 2023, Xiong et al., 2023], Self-Rewarding LLMs [Yuan et al., 2024], SPIN [Chen et al., 2024], and other methods [Rosset et al., 2024]. Common to these approaches is that they have been shown to perform well on general instruction tuning tasks, but they either make only moderate gains or even decrease the performance on standard reasoning tasks. While other kinds of iterative training methods have been applied successfully to reasoning, particularly involving the iteration of supervised fine-tuning (SFT) such as STaR [Zelikman et al., 2022], ReST^{EM} [Singh et al., 2024], and V-STaR [Hosseini et al., 2024][1], using preference optimization to train the generative reasoning model is not applied in these methods.

In this work, we develop an approach to apply iterative preference optimization to reasoning tasks, with a particular focus on Chain-of-Thought (CoT) reasoning [Wu et al., 2023]. On each iteration we sample multiple chain-of-thought reasoning steps and final answers over training prompts, and then construct preference pairs such that pair winners have correct answers and pair losers have wrong answers. We then train a variant of DPO that includes a negative log-likelihood (NLL) loss term for the pair winners, which also proves crucial for performance. Given the newly trained model, we then iterate the procedure by generating new pairs, and training again, starting from the previously trained

* Equal contribution.
[1]V-STaR does use preference optimization, but for training a separate verifier model.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).



Potential Based Diffusion Motion Planning

Yunhao Luo [1] Chen Sun [1] Joshua B. Tenenbaum [2] Yilun Du [2]

Abstract

Effective motion planning in high dimensional spaces is a long-standing open problem in robotics. One class of traditional motion planning algorithms corresponds to potential-based motion planning. An advantage of potential based motion planning is composability – different motion constraints can be easily combined by adding corresponding potentials. However, constructing motion paths from potentials requires solving a global optimization across configuration space potential landscape, which is often prone to local minima. We propose a new approach towards learning potential based motion planning, where we train a neural network to capture and learn an easily optimizable potentials over motion planning trajectories. We illustrate the effectiveness of such approach, significantly outperforming both classical and recent learned motion planning approaches and avoiding issues with local minima. We further illustrate its inherent composability, enabling us to generalize to a multitude of different motion constraints. Project website at https://energy-based-model.github.io/potential-motion-plan.

1 Introduction

Motion planning is a fundamental problem in robotics and aims to find a smooth, collision free path between a start and goal state given a specified configuration space, and is heavily used across a variety of different robotics tasks such as manipulation or navigation (Laumond et al., 1998). A variety of approaches exist for motion planning, ranging from classical sampling based approaches (Kavraki et al., 1996; Kuffner & LaValle, 2000; Karaman & Frazzoli, 2011; Gammell et al., 2015) and optimization based methods (Ratliff et al., 2009; Mukadam et al., 2018; Kalakrishnan et al.,

[1]Brown University [2]MIT. Correspondence to: Yunhao Luo <yluo73@cs.brown.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Figure 1: **Illustrative Example of Composing Diffusion Energy Potentials.** Our approach learns different potential functions over motion planning trajectories $q_0 : H$ (orange dashed lines). Different potentials can be combined and optimized to construct new motion plans that avoid obstacles encoded in both potential functions.

2011). A recent body of works have further explored how learned neural networks can be integrated with motion planning for accelerated performance (Ichter & Pavone, 2019; Qureshi et al., 2019; Fishman et al., 2023; Yamada et al., 2023; Le et al., 2023).

A classical approach towards motion planning is potential based motion planning (Khatib, 1986; Koren et al., 1991; Ratliff et al., 2009; 2018; Xie et al., 2020), where both obstacles and goals define energy potentials through which trajectories are optimized to reach. An advantage of potential based motion planning is that different constraints to motion planning can be converted into equivalent energy potentials and directly combined to optimize for motion plans. However, potential based methods rely on gradient optimization with respect to local geometry, resulting in the long-standing local minima issues (LaValle, 2006). In addition, such motion planning techniques typically require implicit obstacle representations, which is hard to obtain in real-world settings.

We present a potential based motion planning approach leveraging diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) where diffusion models are used to parameterize and learn potential landscapes across configuration space trajectories between start and goal states. These potential functions can be directly inferred from raw perceptual inputs, removing the need for implicit object representations. Furthermore, the annealed optimization procedure across a sequence of potential energy landscapes in diffusion models (Du et al., 2023) can help avoid local minima in optimization. This optimization procedure is further stochastic, enabling the planner to generate a multitude of motion plans with diverse morphology for a specific problem, offering

Diffusion Motion Planning

Paper Presentations

Paper Presentations — Goal

- During the semester, each student gives **one 20-minute presentation** on a paper relevant to probabilistic and generative models — spreadsheet here.
- Ideally from a modern machine learning conference (e.g., NeurIPS, ICML, ICLR, AAAI, etc)
  - Could be a “classic” (older) paper relevant to modern models, as well.
- This will accomplish a few things:
  - Gives the class a chance to see a broad set of interesting probabilistic/generative modeling papers.
  - Gives each student (more) experience with distilling key content from a paper & presenting it.

Paper Presentations — Content of each paper presentation?

Ideally I’d like you to include:

Paper Presentations — Content of each paper presentation?

Ideally I'd like you to include:

1. Motivation (main problem the paper is trying to solve).
2. Brief background on prior work (*i.e.*, related methods, relevant papers).
3. Method and key equations.
    - It'll be important to boil the method down to a concise explanation given ~20 minute presentation.
4. Empirical results — how well does the method work in practice?
5. Your opinion on the paper, its (potential) impact/promise, etc.

Review: Variational Inference

Variational Inference — Overview

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Inference — Overview

We can take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

Variational Inference — Overview

We can take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

The main idea:

Variational Inference — Overview

We can take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

The main idea:

- Suppose we are given an intractable (*complex*) probability distribution $p(x)$.

Variational Inference — Overview

We can take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

The main idea:

- Suppose we are given an intractable (*complex*) probability distribution $p(x)$.

E.g., only known up to a normalization constant

Variational Inference — Overview

We can take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

The main idea:

- Suppose we are given an intractable (*complex*) probability distribution $p(x)$.
- VI aims to solve an optimization problem over a set of tractable distributions Q.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Inference — Overview

We can take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

The main idea:

- Suppose we are given an intractable (*complex*) probability distribution $p(x)$.
- VI aims to solve an optimization problem over a set of tractable distributions Q.
- And, after optimization, return a $q(x) \in Q$ that is *most similar* to $p(x)$.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Inference — Overview

We can take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

The main idea:

- Suppose we are given an intractable (*complex*) probability distribution $p(x)$.
- VI aims to solve an optimization problem over a set of tractable distributions Q.
- And, after optimization, return a $q(x) \in Q$ that is *most similar* to $p(x)$.

Typically: in terms of KL divergence

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Inference — Overview

We can take the strategy/philosophy of *inference as optimization*.

This method of approximate inference is known as **variational inference (VI)**.

The main idea:

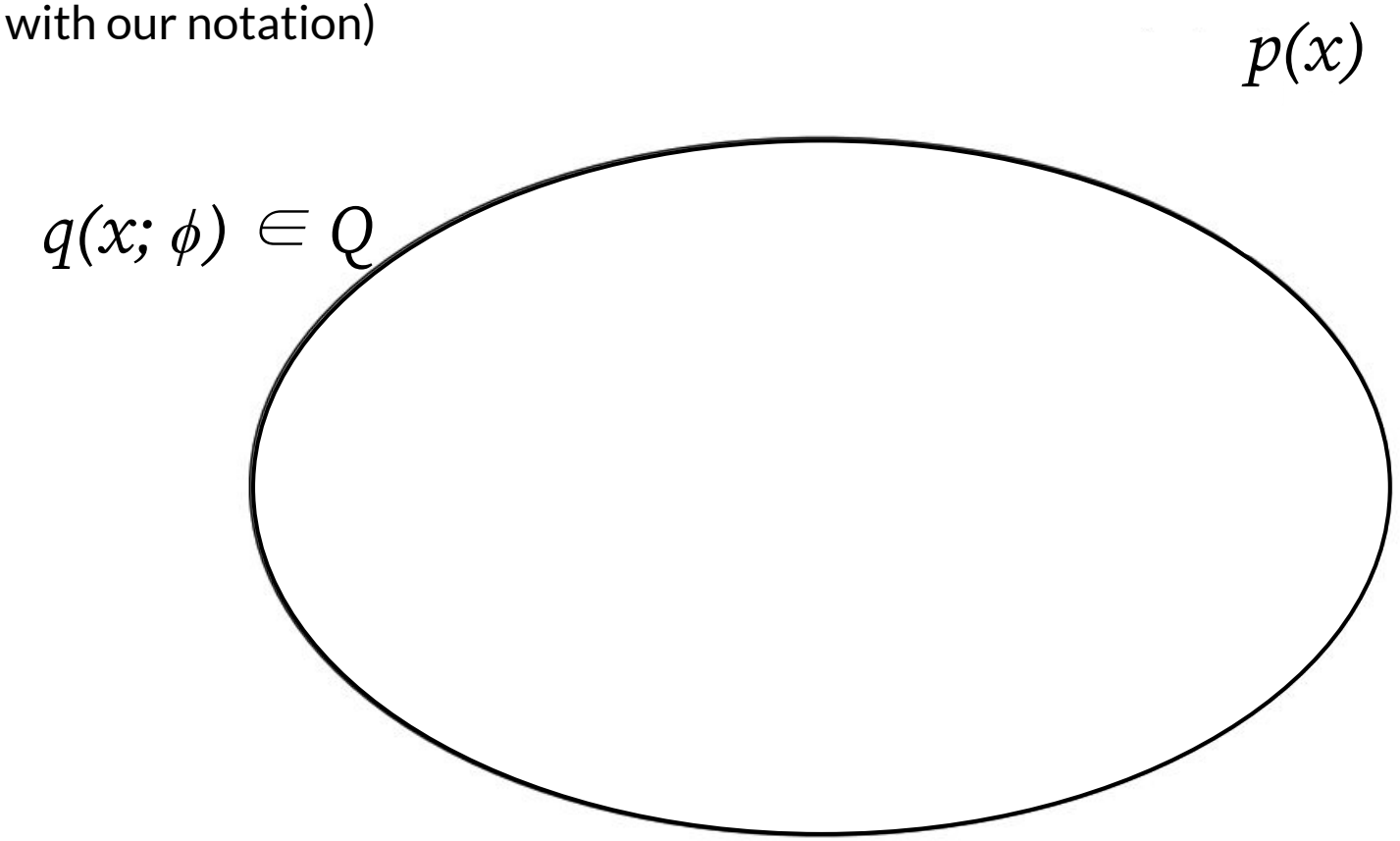
- Suppose we are given an intractable (*complex*) probability distribution $p(x)$.
- VI aims to solve an optimization problem over a set of tractable distributions Q.
- And, after optimization, return a $q(x) \in Q$ that is *most similar* to $p(x)$.
- We can then make an inference query on $q(x)$ instead of $p(x)$.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Inference — Overview

An illustration from Dave Blei's Variational Inference Tutorial:

(replaced with our notation)

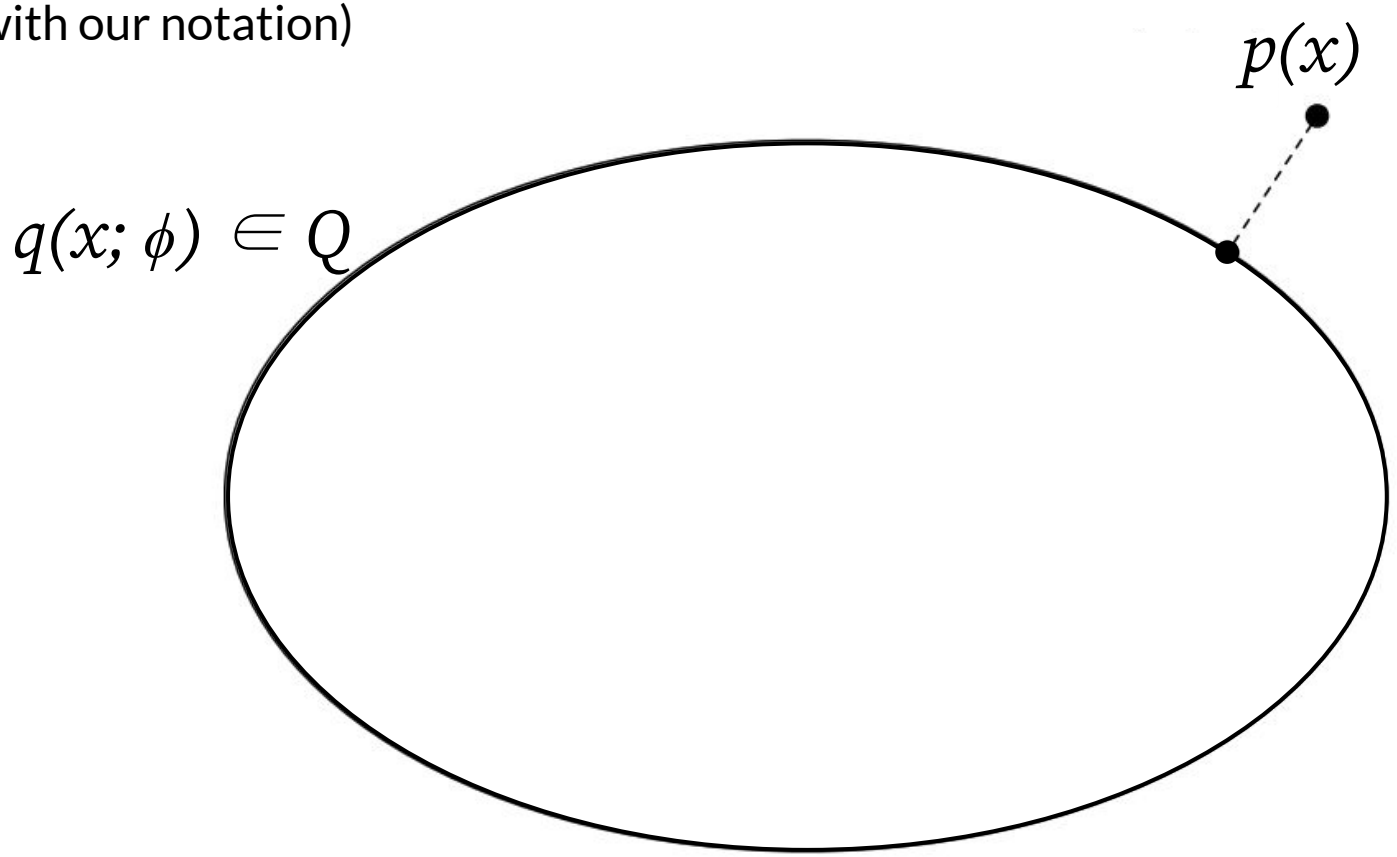


Source: Dave Blei, "Variational Inference"

Variational Inference — Overview

An illustration from Dave Blei's Variational Inference Tutorial:

(replaced with our notation)

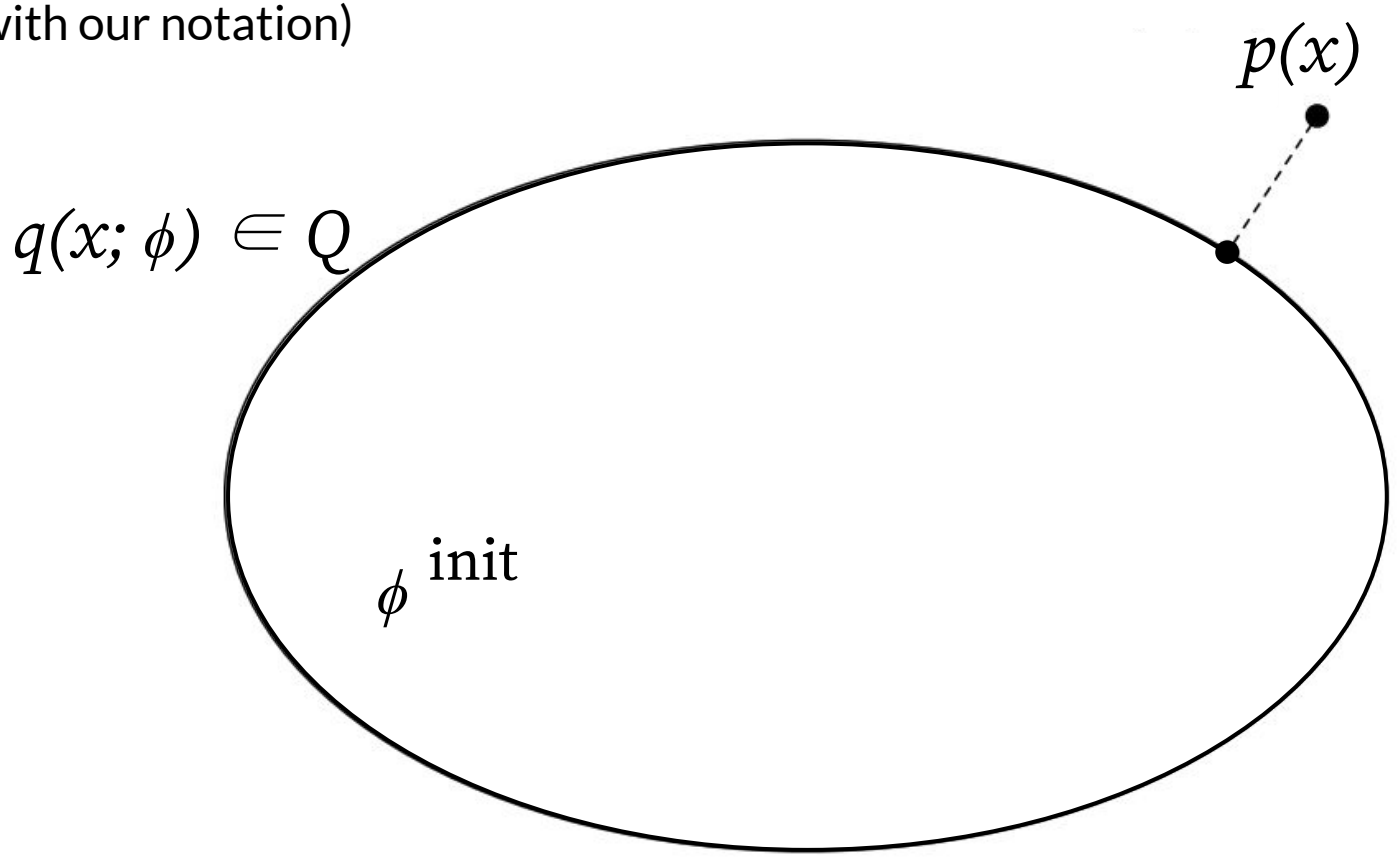


Source: Dave Blei, "Variational Inference"

Variational Inference — Overview

An illustration from Dave Blei's Variational Inference Tutorial:

(replaced with our notation)

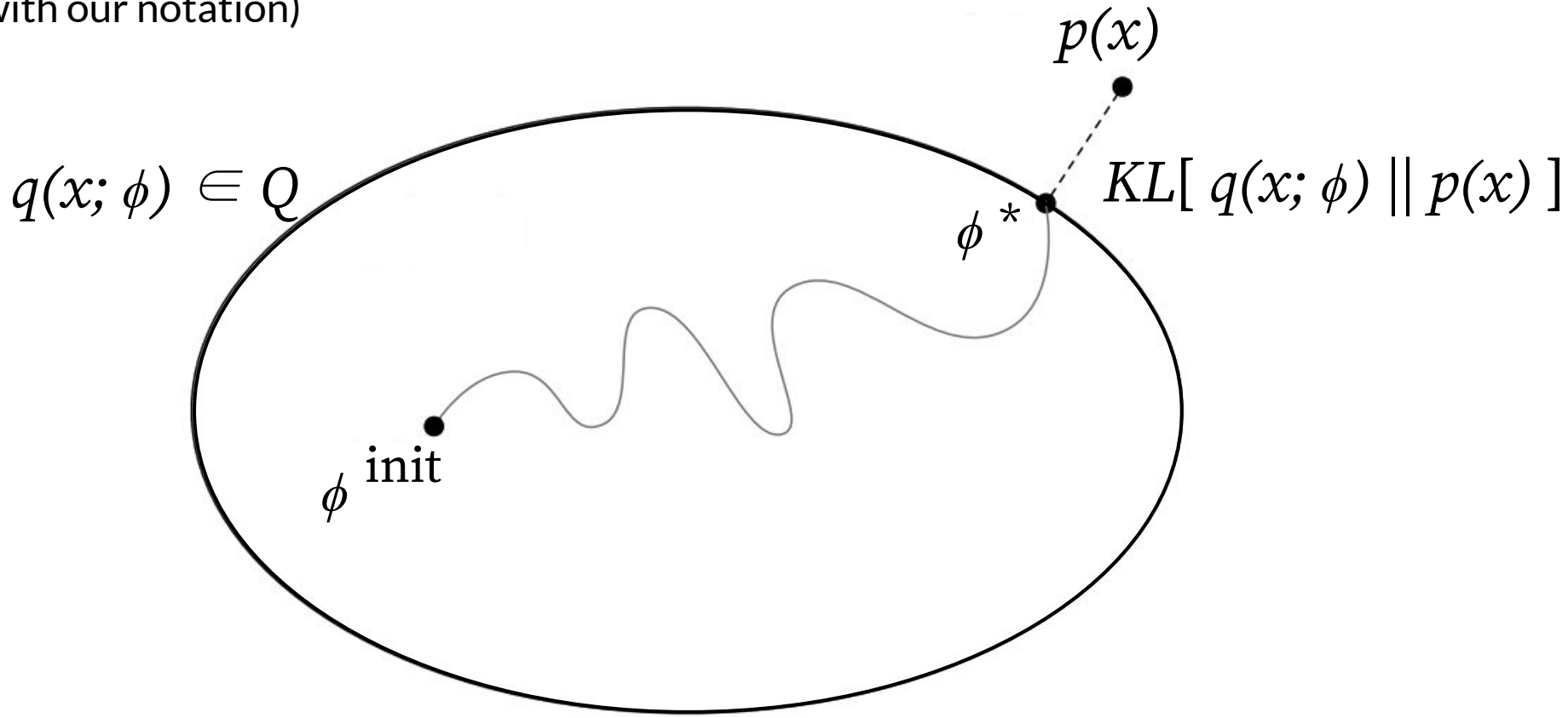


Source: Dave Blei, "Variational Inference"

Variational Inference — Overview

An illustration from Dave Blei's Variational Inference Tutorial:

(replaced with our notation)



Source: Dave Blei, "Variational Inference"

Kullback-Leibler Divergence

More formally:

Kullback-Leibler Divergence

More formally:

The KL divergence between distributions q and p with *discrete support* is:

$$\mathrm{KL}\left[q \parallel p\right] = \sum_{x} q(x) \log\left(\frac{q(x)}{p(x)}\right) = \mathbb{E}_{q(x)}\left[\log\left(\frac{q(x)}{p(x)}\right)\right]$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Kullback-Leibler Divergence

More formally:

The KL divergence between distributions q and p with *discrete support* is:

$$\mathrm{KL}\left[q \parallel p\right] = \sum_{x} q(x) \log\left(\frac{q(x)}{p(x)}\right) = \mathbb{E}_{q(x)}\left[\log\left(\frac{q(x)}{p(x)}\right)\right]$$

And the KL divergence between distributions q and p with *continuous support* is:

$$\mathrm{KL}\left[q \parallel p\right] = \int_{x \in \mathcal{X}} q(x) \log\left(\frac{q(x)}{p(x)}\right) dx = \mathbb{E}_{q(x)}\left[\log\left(\frac{q(x)}{p(x)}\right)\right]$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Kullback-Leibler Divergence

Intuitively, when minimizing KL divergence in variational inference:

(in the words of Dave Blei)

Source: Dave Blei, “Variational Inference”

Kullback-Leibler Divergence

$$\mathrm{KL}\left[q \parallel p\right] = \sum_{x} q(x) \log\left(\frac{q(x)}{p(x)}\right)$$

Intuitively, when minimizing KL divergence in variational inference:

(in the words of Dave Blei)

Source: Dave Blei, “Variational Inference”

Kullback-Leibler Divergence

$$\mathrm{KL}\left[q \parallel p\right] = \sum_{x} q(x) \log\left(\frac{q(x)}{p(x)}\right)$$

Intuitively, when minimizing KL divergence in variational inference:

(in the words of Dave Blei)

- If $q(x)$ is high and $p(x)$ is also high (or higher) then we are happy. 😊

Source: Dave Blei, "Variational Inference"

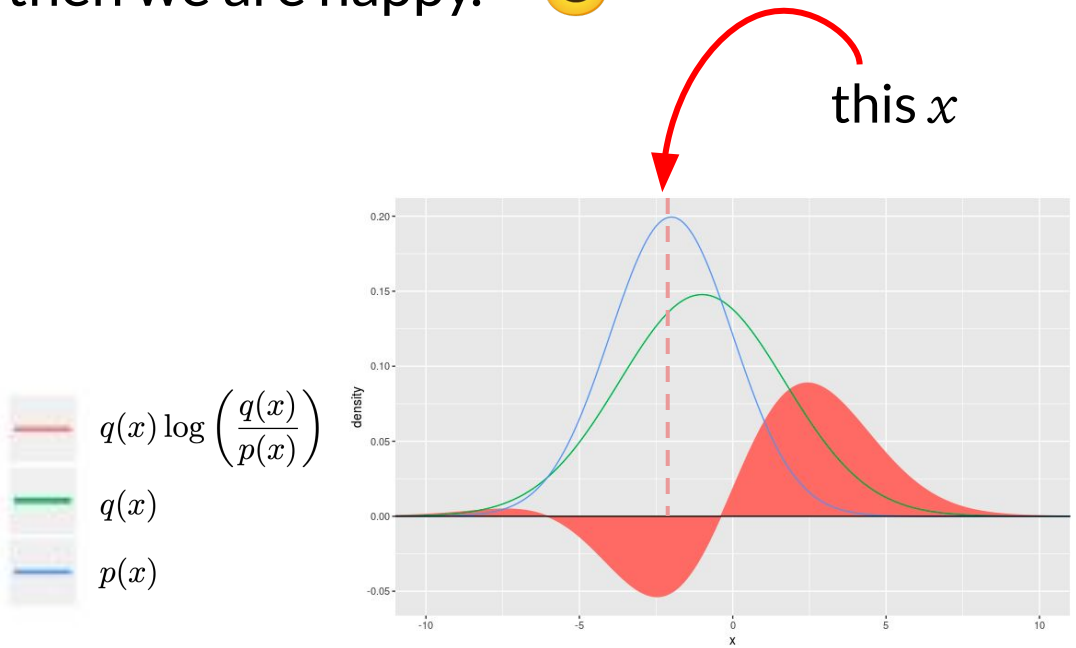
Kullback-Leibler Divergence

$$\mathrm{KL}\left[q \parallel p\right] = \sum_{x} q(x) \log\left(\frac{q(x)}{p(x)}\right)$$

Intuitively, when minimizing KL divergence in variational inference:

(in the words of Dave Blei)

- If $q(x)$ is high and $p(x)$ is also high (or higher) then we are happy. 😊



Source: Dave Blei, "Variational Inference"

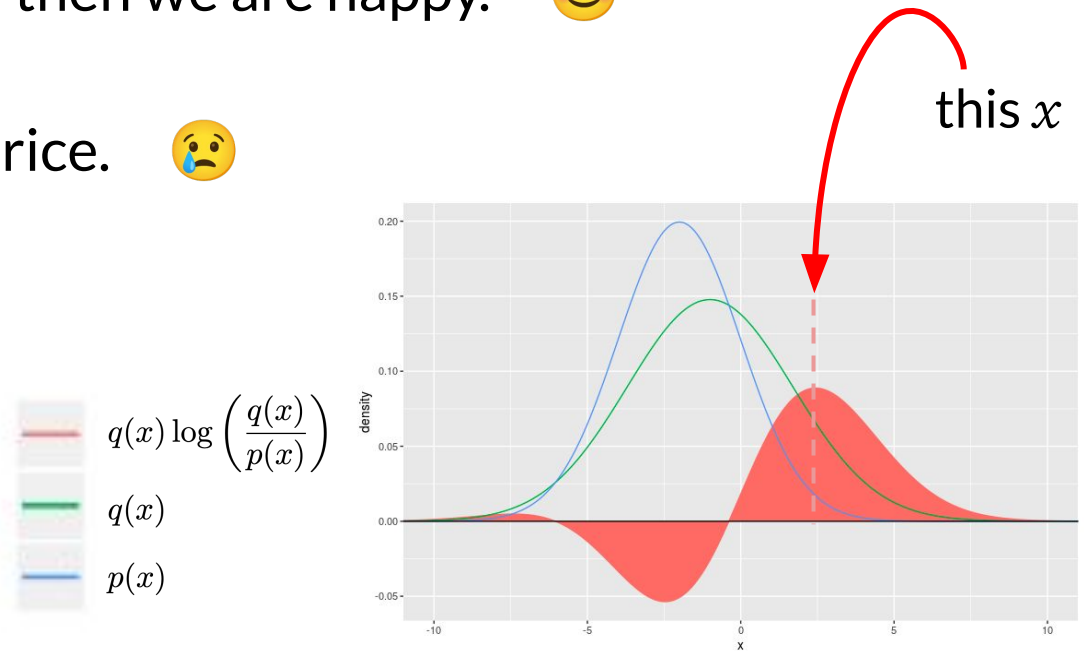
Kullback-Leibler Divergence

$$\mathrm{KL}\left[q \parallel p\right] = \sum_{x} q(x) \log\left(\frac{q(x)}{p(x)}\right)$$

Intuitively, when minimizing KL divergence in variational inference:

(in the words of Dave Blei)

- If $q(x)$ is high and $p(x)$ is also high (or higher) then we are happy. 😊
- If $q(x)$ is high and $p(x)$ is *low*, then we pay a price. 😢



Source: Dave Blei, "Variational Inference"

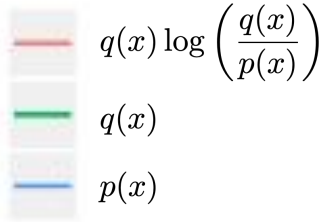
Kullback-Leibler Divergence

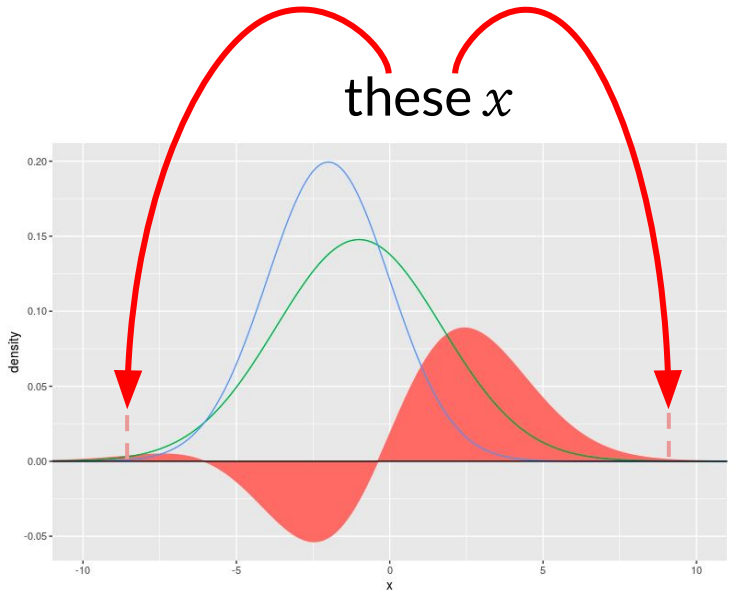
$$\mathrm{KL}\left[q \parallel p\right] = \sum_{x} q(x) \log\left(\frac{q(x)}{p(x)}\right)$$

Intuitively, when minimizing KL divergence in variational inference:

(in the words of Dave Blei)

- If $q(x)$ is high and $p(x)$ is also high (or higher) then we are happy. 😊
- If $q(x)$ is high and $p(x)$ is *low*, then we pay a price. 😢
- If $q(x)$ is low… then we don't care. 🤭





Source: Dave Blei, "Variational Inference"

The Evidence Lower Bound

The Evidence Lower Bound

Suppose we are given a probability model of the form: $p_\theta(x) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$

The Evidence Lower Bound

A general way of writing many PDFs of interest

Suppose we are given a probability model of the form: $p_\theta(x) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$

The Evidence Lower Bound

Suppose we are given a probability model of the form: $p_\theta(x) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$

And that we want to perform variational inference (*i.e.*, optimize $\mathrm{KL}\left[q \parallel p\right]$).

The Evidence Lower Bound

Suppose we are given a probability model of the form: $p_\theta(x) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$

And that we want to perform variational inference (*i.e.*, optimize $\mathrm{KL}\left[q \parallel p\right]$).

It is difficult to optimize $\mathrm{KL}\left[q \parallel p\right]$ directly! For a few reasons:

The Evidence Lower Bound

Suppose we are given a probability model of the form: $p_\theta(x) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$

And that we want to perform variational inference (*i.e.*, optimize $\mathrm{KL}\left[q \parallel p\right]$).

It is difficult to optimize $\mathrm{KL}\left[q \parallel p\right]$ directly! For a few reasons:

- (Often) intractable normalization constant.
- ⇒ difficult to take gradients due to this.
- In fact, it's difficult even to evaluate $\mathrm{KL}\left[q \parallel p\right]$ (or $p(x)$ for that matter).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Therefore, we will not work with the KL divergence directly.

The Evidence Lower Bound

Therefore, we will not work with the KL divergence directly.

Instead, we will use the following objective, which has a similar form as the KL divergence, but only involves the *unnormalized probability* $\tilde{p}(x) = \prod_k \phi_k(x_k; \theta)$.

The Evidence Lower Bound

Therefore, we will not work with the KL divergence directly.

Instead, we will use the following objective, which has a similar form as the KL divergence, but only involves the *unnormalized probability* $\tilde{p}(x) = \prod_k \phi_k(x_k; \theta)$.

This objective is:

The Evidence Lower Bound

Therefore, we will not work with the KL divergence directly.

Instead, we will use the following objective, which has a similar form as the KL divergence, but only involves the *unnormalized probability* $\tilde{p}(x) = \prod_k \phi_k(x_k; \theta)$.

This objective is:

$$J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)} = \mathbb{E}_{q(x)} \left[\log \frac{q(x)}{\tilde{p}(x)} \right]$$

For discrete distributions

The Evidence Lower Bound

Therefore, we will not work with the KL divergence directly.

Instead, we will use the following objective, which has a similar form as the KL divergence, but only involves the *unnormalized probability* $\tilde{p}(x) = \prod_k \phi_k(x_k; \theta)$.

This objective is:

$$J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)} = \mathbb{E}_{q(x)} \left[\log \frac{q(x)}{\tilde{p}(x)} \right]$$

For discrete distributions

$$J(q) = \int_{\mathcal{X}} q(x) \log \frac{q(x)}{\tilde{p}(x)} dx = \mathbb{E}_{q(x)} \left[\log \frac{q(x)}{\tilde{p}(x)} \right]$$

For continuous distributions

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

This function is *tractable* (we can evaluate it), and it has the following property:

The Evidence Lower Bound

This function is *tractable* (we can evaluate it), and it has the following property:

Assuming a discrete distribution WLOG

$$
\begin{aligned}
J(q) &= \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)} \\
&= \sum_x q(x) \log \frac{q(x)}{p(x)} - \log Z(\theta) \\
&= \mathrm{KL}\left[q(x) \parallel p(x)\right] - \log Z(\theta)
\end{aligned}
$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

This function is *tractable* (we can evaluate it), and it has the following property:

$$J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)}$$

Definition of $J(q)$

Assuming a discrete distribution WLOG

$$= \sum_x q(x) \log \frac{q(x)}{p(x)} - \log Z(\theta)$$

$$= \mathrm{KL}\left[q(x) \,\|\, p(x)\right] - \log Z(\theta)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

This function is *tractable* (we can evaluate it), and it has the following property:

$$J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)}$$

Definition of $J(q)$

Assuming a discrete distribution WLOG

$$= \sum_x q(x) \log \frac{q(x)}{p(x)} - \log Z(\theta)$$

Definition of $p(x)$

$$= \mathrm{KL}\left[q(x) \,\|\, p(x)\right] - \log Z(\theta)$$

The Evidence Lower Bound

This function is *tractable* (we can evaluate it), and it has the following property:

Assuming a discrete distribution WLOG

$$J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)}$$

Definition of $J(q)$

$$= \sum_x q(x) \log \frac{q(x)}{p(x)} - \log Z(\theta)$$

Definition of $p(x)$

$$= \mathrm{KL}\left[q(x) \parallel p(x)\right] - \log Z(\theta)$$

Definition of KL

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

This function is *tractable* (we can evaluate it), and it has the following property:

Assuming a discrete distribution WLOG

$$J(q) = \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)}$$

Definition of $J(q)$

$$= \sum_x q(x) \log \frac{q(x)}{p(x)} - \log Z(\theta)$$

Definition of $p(x)$

$$= \mathrm{KL}\left[q(x) \,\|\, p(x)\right] - \log Z(\theta)$$

Definition of KL

$\Rightarrow J(q)$ is equal to the KL divergence minus the *log partition function* (*aka log normalizer*).

(*aka "marginal log-likelihood"*).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Also note that KL divergence is non-negative: $\mathrm{KL}\left[q \parallel p\right] \geq 0$

Which implies the following relation:

The Evidence Lower Bound

Also note that KL divergence is non-negative: $\mathrm{KL}\left[q \parallel p\right] \geq 0$

Which implies the following relation:

$$\log Z(\theta) = \mathrm{KL}\left[q(x) \parallel p(x)\right] - J(q)$$

The Evidence Lower Bound

Also note that KL divergence is non-negative: $\mathrm{KL}\left[q \parallel p\right] \geq 0$

Which implies the following relation:

Since KL is 0, at lowest

$$\log Z(\theta) = \mathrm{KL}\left[q(x) \parallel p(x)\right] - J(q) \geq -J(q)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Also note that KL divergence is non-negative: $\mathrm{KL}\left[q \parallel p\right] \geq 0$

Which implies the following relation:

Since KL is 0, at lowest

$$\log Z(\theta) = \mathrm{KL}\left[q(x) \parallel p(x)\right] - J(q) \geq -J(q)$$

$$\Rightarrow \log Z(\theta) \geq -J(q)$$

The Evidence Lower Bound

Also note that KL divergence is non-negative: $\mathrm{KL}\left[q \parallel p\right] \geq 0$

Which implies the following relation:

Since KL is 0, at lowest

$$\log Z(\theta) = \mathrm{KL}\left[q(x) \parallel p(x)\right] - J(q) \geq -J(q)$$

$$\Rightarrow \log Z(\theta) \geq -J(q)$$

Therefore, $-J(q)$ is a *lower bound* on the log-partition function, $\log Z(\theta)$.

This is an important quantity! **We call $-J(q)$ the ELBO.**

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound — Why do we call it the ELBO?

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound — Why do we call it the ELBO?

Recall that we have been writing probabilistic models over $x = (x_1, \ldots, x_n)$ as a normalized product of factors:

$$p_\theta(x_1, \ldots, x_n) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x_1, \ldots, x_n) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound — Why do we call it the ELBO?

Recall that we have been writing probabilistic models over $x = (x_1, \ldots, x_n)$ as a normalized product of factors:

$$p_\theta(x_1, \ldots, x_n) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x_1, \ldots, x_n) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$$

A distribution (PDF) of interest

The Evidence Lower Bound — Why do we call it the ELBO?

Recall that we have been writing probabilistic models over $x = (x_1, \ldots, x_n)$ as a normalized product of factors:

$$p_\theta(x_1, \ldots, x_n) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x_1, \ldots, x_n) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$$

A distribution (PDF) of interest

Unnormalized PDF with normalization

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound — Why do we call it the ELBO?

Recall that we have been writing probabilistic models over $x = (x_1, \ldots, x_n)$ as a normalized product of factors:

$$p_\theta(x_1, \ldots, x_n) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x_1, \ldots, x_n) = \frac{1}{Z(\theta)} \prod_k \phi_k(x_k; \theta)$$

A distribution (PDF) of interest

Unnormalized PDF with normalization

Normalized product of factors

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound — Why do we call it the ELBO?

As an example

The Evidence Lower Bound — Why do we call it the ELBO?

As an example, consider **posterior distributions** (of latents given observed), *i.e.*, $p(y \mid x_1, \ldots, x_n)$. We can write

The Evidence Lower Bound — Why do we call it the ELBO?

As an example, consider **posterior distributions** (of latents given observed), *i.e.*, $p(y \mid x_1, \ldots, x_n)$. We can write

$$p_\theta(y \mid x_1, \ldots, x_n) = \frac{p_\theta(y, x_1, \ldots, x_n)}{p_\theta(x_1, \ldots, x_n)} = \frac{1}{Z(\theta)} p_\theta(y, x_1, \ldots, x_n)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound — Why do we call it the ELBO?

As an example, consider **posterior distributions** (of latents given observed), *i.e.*, $p(y \mid x_1, \ldots, x_n)$. We can write

$$p_\theta(y \mid x_1, \ldots, x_n) = \frac{p_\theta(y, x_1, \ldots, x_n)}{p_\theta(x_1, \ldots, x_n)} = \frac{1}{Z(\theta)} p_\theta(y, x_1, \ldots, x_n)$$

Definition of conditional PDF

The Evidence Lower Bound — Why do we call it the ELBO?

As an example, consider **posterior distributions** (of latents given observed), *i.e.*, $p(y \mid x_1, \ldots, x_n)$. We can write

$$p_\theta(y \mid x_1, \ldots, x_n) = \frac{p_\theta(y, x_1, \ldots, x_n)}{p_\theta(x_1, \ldots, x_n)} = \frac{1}{Z(\theta)} p_\theta(y, x_1, \ldots, x_n)$$

Definition of conditional PDF

Write as normalized joint PDF

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound — Why do we call it the ELBO?

As an example, consider **posterior distributions** (of latents given observed), *i.e.*, $p(y \mid x_1, \ldots, x_n)$. We can write

$$p_\theta(y \mid x_1, \ldots, x_n) = \frac{p_\theta(y, x_1, \ldots, x_n)}{p_\theta(x_1, \ldots, x_n)} = \frac{1}{Z(\theta)} p_\theta(y, x_1, \ldots, x_n)$$

Here, the partition function is $Z(\theta) = p_\theta(x_1, \ldots, x_n)$, often called the **evidence**.

(*or "marginal likelihood"*)

The Evidence Lower Bound — Why do we call it the ELBO?

As an example, consider **posterior distributions** (of latents given observed), *i.e.*, $p(y \mid x_1, \ldots, x_n)$. We can write

$$p_\theta(y \mid x_1, \ldots, x_n) = \frac{p_\theta(y, x_1, \ldots, x_n)}{p_\theta(x_1, \ldots, x_n)} = \frac{1}{Z(\theta)} p_\theta(y, x_1, \ldots, x_n)$$

Here, the partition function is $Z(\theta) = p_\theta(x_1, \ldots, x_n)$, often called the **evidence**.

(*or "marginal likelihood"*)

Thus, $-J(q) \leq \log Z(\theta)$ is often referred to as the ***evidence lower bound***, or **ELBO**.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

To summarize, the ELBO is:

The Evidence Lower Bound

To summarize, the ELBO is:

For discrete distributions

$$-J(q) = -\sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)} = \sum_x q(x) \log \frac{\tilde{p}(x)}{q(x)} = \mathbb{E}_{q(x)} \left[\log \frac{\tilde{p}(x)}{q(x)} \right]$$

The Evidence Lower Bound

To summarize, the ELBO is:

For discrete distributions

$$-J(q) = -\sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)} = \sum_x q(x) \log \frac{\tilde{p}(x)}{q(x)} = \mathbb{E}_{q(x)} \left[\log \frac{\tilde{p}(x)}{q(x)} \right]$$

For continuous distributions

$$-J(q) = -\int_{\mathcal{X}} q(x) \log \frac{q(x)}{\tilde{p}(x)} dx = \int_{\mathcal{X}} q(x) \log \frac{\tilde{p}(x)}{q(x)} dx = \mathbb{E}_{q(x)} \left[\log \frac{\tilde{p}(x)}{q(x)} \right]$$

The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$
\begin{aligned}
-J(q) = \mathbb{E}_{q(x)}\left[\log \tilde{p}(x) - \log q(x)\right] &= \log Z(\theta) - \mathrm{KL}\left[q(x) \parallel p(x)\right] \\
&\leq \log Z(\theta)
\end{aligned}
$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$-J(q) = \mathbb{E}_{q(x)}\left[\log \tilde{p}(x) - \log q(x)\right] = \log Z(\theta) - \mathrm{KL}\left[q(x) \parallel p(x)\right]$$
$$\leq \log Z(\theta)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$-J(q) = \mathbb{E}_{q(x)}\left[\log \tilde{p}(x) - \log q(x)\right] = \log Z(\theta) - \mathrm{KL}\left[q(x) \parallel p(x)\right]$$
$$\leq \log Z(\theta)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$
\begin{aligned}
-J(q) = \mathbb{E}_{q(x)}\left[\log \tilde{p}(x) - \log q(x)\right] &= \log Z(\theta) - \mathrm{KL}\left[q(x) \parallel p(x)\right] \\
&\leq \log Z(\theta)
\end{aligned}
$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$
\begin{aligned}
-J(q) = \mathbb{E}_{q(x)}\left[\log \tilde{p}(x) - \log q(x)\right] &= \log Z(\theta) - \mathrm{KL}\left[q(x) \parallel p(x)\right] \\
&\leq \log Z(\theta)
\end{aligned}
$$

Importantly, note that:

The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$
\begin{aligned}
-J(q) = \mathbb{E}_{q(x)}\left[\log \tilde{p}(x) - \log q(x)\right] &= \log Z(\theta) - \mathrm{KL}\left[q(x) \parallel p(x)\right] \\
&\leq \log Z(\theta)
\end{aligned}
$$

Importantly, note that:

- The difference between $\log Z(\theta)$ and $-J(q)$ is $\mathrm{KL}\left[q \parallel p\right]$.

The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$
\begin{aligned}
-J(q) = \mathbb{E}_{q(x)}\left[\log \tilde{p}(x) - \log q(x)\right] &= \log Z(\theta) - \mathrm{KL}\left[q(x) \parallel p(x)\right] \\
&\leq \log Z(\theta)
\end{aligned}
$$

Importantly, note that:

- The difference between $\log Z(\theta)$ and $-J(q)$ is $\mathrm{KL}\left[q \parallel p\right]$.
- Therefore, by maximizing the ELBO, we are *minimizing* $\mathrm{KL}\left[q \parallel p\right]$...

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Evidence Lower Bound

Also, the ELBO is often written in the form:

$$
\begin{aligned}
-J(q) = \mathbb{E}_{q(x)}\left[\log \tilde{p}(x) - \log q(x)\right] &= \log Z(\theta) - \mathrm{KL}\left[q(x) \parallel p(x)\right] \\
&\leq \log Z(\theta)
\end{aligned}
$$

Importantly, note that:

- The difference between $\log Z(\theta)$ and $-J(q)$ is $\mathrm{KL}\left[q \parallel p\right]$.
- Therefore, by maximizing the ELBO, we are *minimizing* $\mathrm{KL}\left[q \parallel p\right]$...
- ... by "*squeezing*" it between $-J(q)$ and $\log Z(\theta)$.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

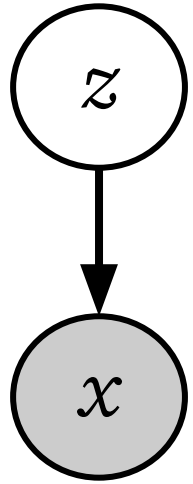
Setup

Setup

For simplicity, consider a latent variable model of the form:

$$p_\theta(x, z) = p_\theta(x \mid z) p_\theta(z)$$

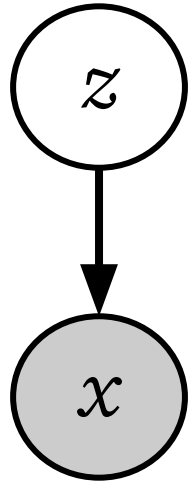
Example PGM



Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Setup

Example PGM



For simplicity, consider a latent variable model of the form:

$$p_\theta(x, z) = p_\theta(x \mid z) p_\theta(z)$$

where:

- $x \in \mathcal{X}$ are observed variables.
  - ($\mathcal{X}$ can be either discrete or continuous.)
- $z \in \mathbb{R}^k$ are latent variables.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

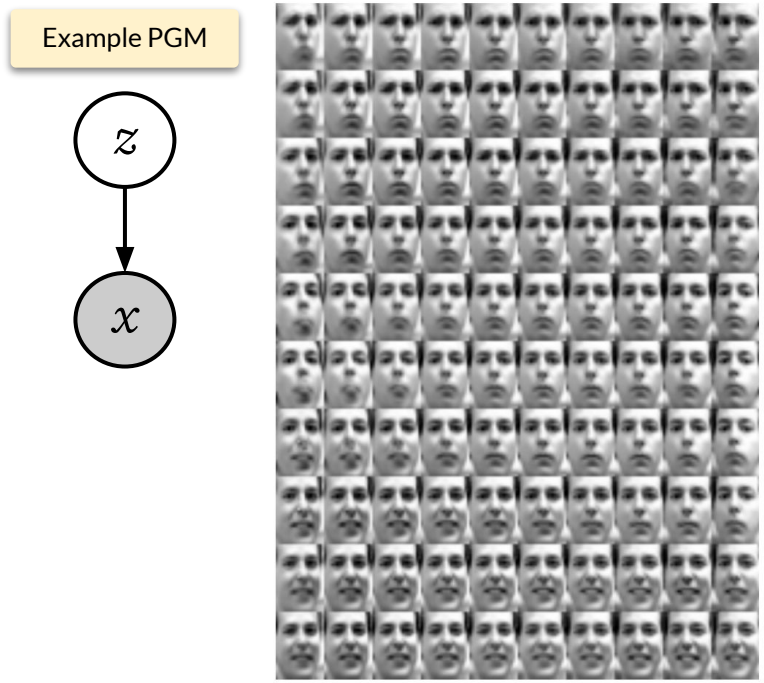
Setup

As an example, for intuition, you can imagine:

Setup

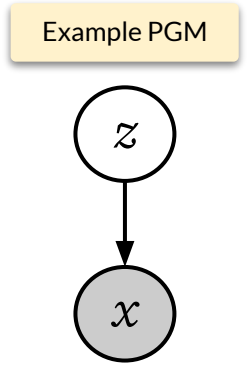
As an example, for intuition, you can imagine:

- x to be an image (*e.g.*, of a human face).



Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Setup



As an example, for intuition, you can imagine:

- x to be an image (*e.g.*, of a human face).
- z to be latent factors, which explain features of the face, *e.g.*:
  - One coordinate could encode whether the face is happy or sad.
  - One coordinate could encode whether the face has short vs long hair.
  - One coordinate could encode whether the face has a beard vs clean shaven.
  - etc.

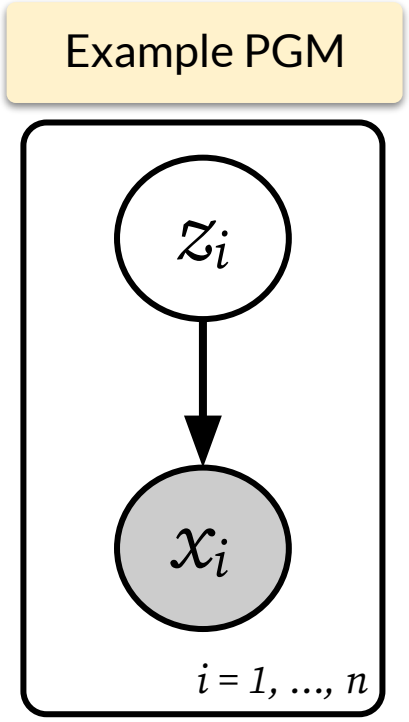
Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Setup

Suppose now that we have a dataset: $D = \{x_1, x_2, \ldots, x_n\}$

Setup

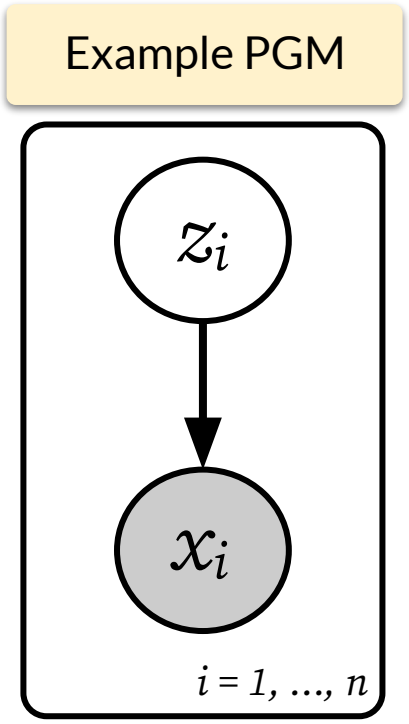
Suppose now that we have a dataset: $D = \{x_1, x_2, \ldots, x_n\}$



Setup

Suppose now that we have a dataset: $D = \{x_1, x_2, \ldots, x_n\}$

We are interested in the following inference and learning tasks:

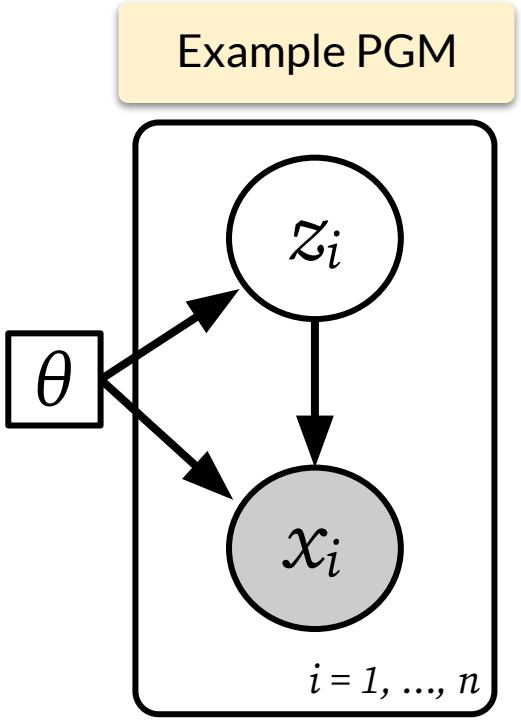


Setup

Suppose now that we have a dataset: $D = \{x_1, x_2, \ldots, x_n\}$

We are interested in the following inference and learning tasks:

(1) Learning the parameters θ of our model p_θ .



Setup

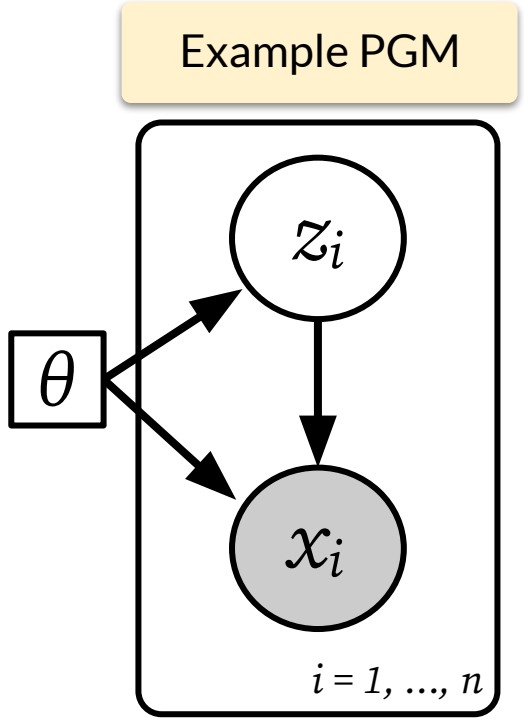
Suppose now that we have a dataset: $D = \{x_1, x_2, \ldots, x_n\}$

We are interested in the following inference and learning tasks:

(1) Learning the parameters θ of our model p_θ.

(2) Approximate posterior inference over z:

Given an image x, what are it's latent factors, *i.e.*, what is $p_\theta(z \mid x)$?



Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Background: VAE Paper

(From ICLR 2014)

Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions is two-fold. First, we show that a reparameterization of the variational lower bound with an independent noise variable yields a lower bound estimator that can be jointly optimized w.r.t. variational and generative parameters using standard gradient-based stochastic optimization methods. Second, we show that posterior inference can be made especially efficient by optimizing a probabilistic encoder (also called a recognition model) to approximate the intractable posterior, using the proposed estimator. Theoretical advantages are reflected in experimental results.

1 Introduction

How can we efficiently learn the parameters of directed probabilistic models whose continuous latent variables have intractable posterior distributions? The variational approach to Bayesian inference involves the introduction of an approximation to the intractable posterior, used to maximize the variational lower bound on the marginal likelihood. Unfortunately, the common mean-field approach requires analytical solutions of expectations w.r.t. the approximate posterior, which are also intractable in the general case. We show how a reparameterization of the variational lower bound yields a practical differentiable estimator of the lower bound. This SGVB (Stochastic Gradient Variational Bayes) estimator can be straightforwardly used as a stochastic objective function, and that can be jointly optimized w.r.t. both the variational and generative parameters, using standard stochastic gradient ascent techniques.

The SGVB algorithm can be applied to learning almost any generative model with continuous latent variables. When we use a neural network for the posterior approximation, we arrive at a *variational auto-encoder*. The SGVB objective for this case contains a regularization term dictated by the variational bound, and a stochastic data reconstruction term. From the learned generative model it is straightforward to generate samples simply by ancestral sampling. The probabilistic encoder can be used for fast approximate inference of latent variables, i.e. for recognition, representation or visualization purposes. Furthermore, the lower bound estimator can be used for inference tasks such as denoising and inpainting.

2 Method

The strategy in this section can be used to derive a lower bound estimator (a stochastic objective function) for a variety of directed graphical models with continuous latent variables. We will restrict ourselves here to the common case where we have an i.i.d. dataset with latent variables per datapoint, and where we like to perform maximum likelihood (ML) or maximum a posteriori (MAP) inference

Background: VAE Paper

(From ICLR 2014)

Introduces two distinct things:

(1) *Auto-encoding variational Bayes* (AEVB) algorithm.
(a stochastic-gradient VI procedure)

(2) A special case (involving neural networks): the *variational autoencoder* (VAE).

Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions is two-fold. First, we show that a reparameterization of the variational lower bound with an independent noise variable yields a lower bound estimator that can be jointly optimized w.r.t. variational and generative parameters using standard gradient-based stochastic optimization methods. Second, we show that posterior inference can be made especially efficient by optimizing a probabilistic encoder (also called a recognition model) to approximate the intractable posterior, using the proposed estimator. Theoretical advantages are reflected in experimental results.

1 Introduction

How can we efficiently learn the parameters of directed probabilistic models whose continuous latent variables have intractable posterior distributions? The variational approach to Bayesian inference involves the introduction of an approximation to the intractable posterior, used to maximize the variational lower bound on the marginal likelihood. Unfortunately, the common mean-field approach requires analytical solutions of expectations w.r.t. the approximate posterior, which are also intractable in the general case. We show how a reparameterization of the variational lower bound yields a practical differentiable estimator of the lower bound. This SGVB (Stochastic Gradient Variational Bayes) estimator can be straightforwardly used as a stochastic objective function, and that can be jointly optimized w.r.t. both the variational and generative parameters, using standard stochastic gradient ascent techniques.

The SGVB algorithm can be applied to learning almost any generative model with continuous latent variables. When we use a neural network for the posterior approximation, we arrive at a *variational auto-encoder*. The SGVB objective for this case contains a regularization term dictated by the variational bound, and a stochastic data reconstruction term. From the learned generative model it is straightforward to generate samples simply by ancestral sampling. The probabilistic encoder can be used for fast approximate inference of latent variables, i.e. for recognition, representation or visualization purposes. Furthermore, the lower bound estimator can be used for inference tasks such as denoising and inpainting.

2 Method

The strategy in this section can be used to derive a lower bound estimator (a stochastic objective function) for a variety of directed graphical models with continuous latent variables. We will restrict ourselves here to the common case where we have an i.i.d. dataset with latent variables per datapoint, and where we like to perform maximum likelihood (ML) or maximum a posteriori (MAP) inference

1

Auto-encoding Variational Bayes (AEVB)

Auto-encoding Variational Bayes (AEVB)

The Auto-encoding Variational Bayes (AEVB) algorithm aims to solve the tasks from before, in particular:

Auto-encoding Variational Bayes (AEVB)

The Auto-encoding Variational Bayes (AEVB) algorithm aims to solve the tasks from before, in particular:

(1) Learning the parameters θ of our model p_θ .

(2) Approximate posterior inference over z : $p_\theta(z \mid x)$

Auto-encoding Variational Bayes (AEVB)

The Auto-encoding Variational Bayes (AEVB) algorithm aims to solve the tasks from before, in particular:

(1) Learning the parameters θ of our model p_θ.

The task of generative modeling

(2) Approximate posterior inference over z: $p_\theta(z \mid x)$

Auto-encoding Variational Bayes (AEVB)

The Auto-encoding Variational Bayes (AEVB) algorithm aims to solve the tasks from before, in particular:

(1) Learning the parameters θ of our model p_θ.

The task of generative modeling

(2) Approximate posterior inference over z: $p_\theta(z \mid x)$

We'll start with methods for approximate posterior inference, and then show how this can lead to learning the parameters (and thus to generative modeling!)

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Auto-encoding Variational Bayes (AEVB) — Family of Approximations

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Auto-encoding Variational Bayes (AEVB) — Family of Approximations

Suppose we have a family of approximations for our model posterior $p_\theta(z \mid x)$, written $q_\phi(z \mid x)$, which is parameterized by some parameter ϕ.

Auto-encoding Variational Bayes (AEVB) — Family of Approximations

Suppose we have a family of approximations for our model posterior $p_\theta(z \mid x)$, written $q_\phi(z \mid x)$, which is parameterized by some parameter ϕ.

A note on the family of approximations $q_\phi(z \mid x)$:

- We are trying to approximate a posterior distribution for any given observation x.
- So, if we want, we can define a family of approximations as a function of x.
- This effectively yields a different variational approximation $q_\phi(z)$ for each x.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Auto-encoding Variational Bayes (AEVB) — The ELBO

Suppose we have a family of approximations for our model posterior $p_\theta(z \mid x)$, written $q_\phi(z \mid x)$, which is parameterized by some parameter ϕ.

Auto-encoding Variational Bayes (AEVB) — The ELBO

Suppose we have a family of approximations for our model posterior $p_\theta(z \mid x)$, written $q_\phi(z \mid x)$, which is parameterized by some parameter ϕ.

Consider an ELBO for this model, written as:

$$\mathcal{L}(p_\theta, q_\phi) = \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x, z) - \log q_\phi(z \mid x)\right]$$

Auto-encoding Variational Bayes (AEVB) — The ELBO

Suppose we have a family of approximations for our model posterior $p_\theta(z \mid x)$, written $q_\phi(z \mid x)$, which is parameterized by some parameter ϕ.

Consider an ELBO for this model, written as:

This is $-J(q)$ from before!

$$\mathcal{L}(p_\theta, q_\phi) = \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x, z) - \log q_\phi(z \mid x)\right]$$

Log of unnormalized PDF

Log of approximate PDF

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Auto-encoding Variational Bayes (AEVB) — The ELBO

Suppose we have a family of approximations for our model posterior $p_\theta(z \mid x)$, written $q_\phi(z \mid x)$, which is parameterized by some parameter ϕ.

Consider an ELBO for this model, written as:

This is $-J(q)$ from before!

$$\mathcal{L}(p_\theta, q_\phi) = \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x, z) - \log q_\phi(z \mid x)\right]$$

Note that the ELBO satisfies:

$$\log p_\theta(x) = \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z \mid x)\right] + \mathcal{L}(p_\theta, q_\phi).$$

Evidence equals KL divergence plus ELBO.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Auto-encoding Variational Bayes (AEVB) — BBVI

How can we optimize the ELBO to find the optimal approximation $q_\phi(z \mid x)$?

Auto-encoding Variational Bayes (AEVB) — BBVI

How can we optimize the ELBO to find the optimal approximation $q_\phi(z \mid x)$?

⇒ We'll use a *black-box variational inference* approach.

This means a VI method that aims to be automatic / applied in a general purpose way.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Auto-encoding Variational Bayes (AEVB) — BBVI

How can we optimize the ELBO to find the optimal approximation $q_\phi(z \mid x)$?

⇒ We'll use a *black-box variational inference* approach.

This means a VI method that aims to be automatic / applied in a general purpose way.

In particular:

- Maximize the ELBO via gradient descent over ϕ.
- Simply requires q_ϕ to be differentiable with respect to ϕ.
- (Along with a couple of other things…)

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Auto-encoding Variational Bayes (AEVB) — Learning and Inference

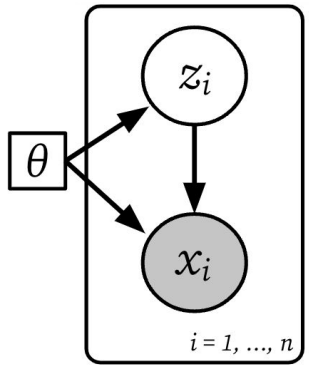
Furthermore

- Instead of just inference, we'll simultaneously do learning of parameters θ.

Auto-encoding Variational Bayes (AEVB) — Learning and Inference

Furthermore

- Instead of just inference, we'll simultaneously do learning of parameters θ.



Intuitively:

Simultaneously update our probabilistic model, *while* we are performing approximate posterior inference in the model.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Auto-encoding Variational Bayes (AEVB) — Learning and Inference

Furthermore

- Instead of just inference, we'll simultaneously do learning of parameters θ.
- ⇒ Via gradient descent over both ϕ and θ.

Auto-encoding Variational Bayes (AEVB) — Learning and Inference

Furthermore

- Instead of just inference, we'll simultaneously do learning of parameters θ.
- ⇒ Via gradient descent over both ϕ and θ.
- Optimization of ϕ moves the ELBO closer to $\log p(x)$ (*i.e.*, minimizes KL divergence).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Auto-encoding Variational Bayes (AEVB) — Learning and Inference

Furthermore

- Instead of just inference, we'll simultaneously do learning of parameters θ.
- ⇒ Via gradient descent over both ϕ and θ.
- Optimization of ϕ moves the ELBO closer to $\log p(x)$ (*i.e.*, minimizes KL divergence).
- Optimization of θ keeps pushing the upper bound (*i.e.*, the evidence $\log p(x)$) up.
  - ⇒ Higher likelihood, better learning, and and improved generative model.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Auto-encoding Variational Bayes (AEVB) — Learning and Inference

Furthermore

- Instead of just inference, we'll simultaneously do learning of parameters θ.
- ⇒ Via gradient descent over both ϕ and θ.
- Optimization of ϕ moves the ELBO closer to $\log p(x)$ (*i.e.*, minimizes KL divergence).
- Optimization of θ keeps pushing the upper bound (*i.e.*, the evidence $\log p(x)$) up.
  - ⇒ Higher likelihood, better learning, and and improved generative model.

This is somewhat similar to the expectation-maximization (EM) algorithm…

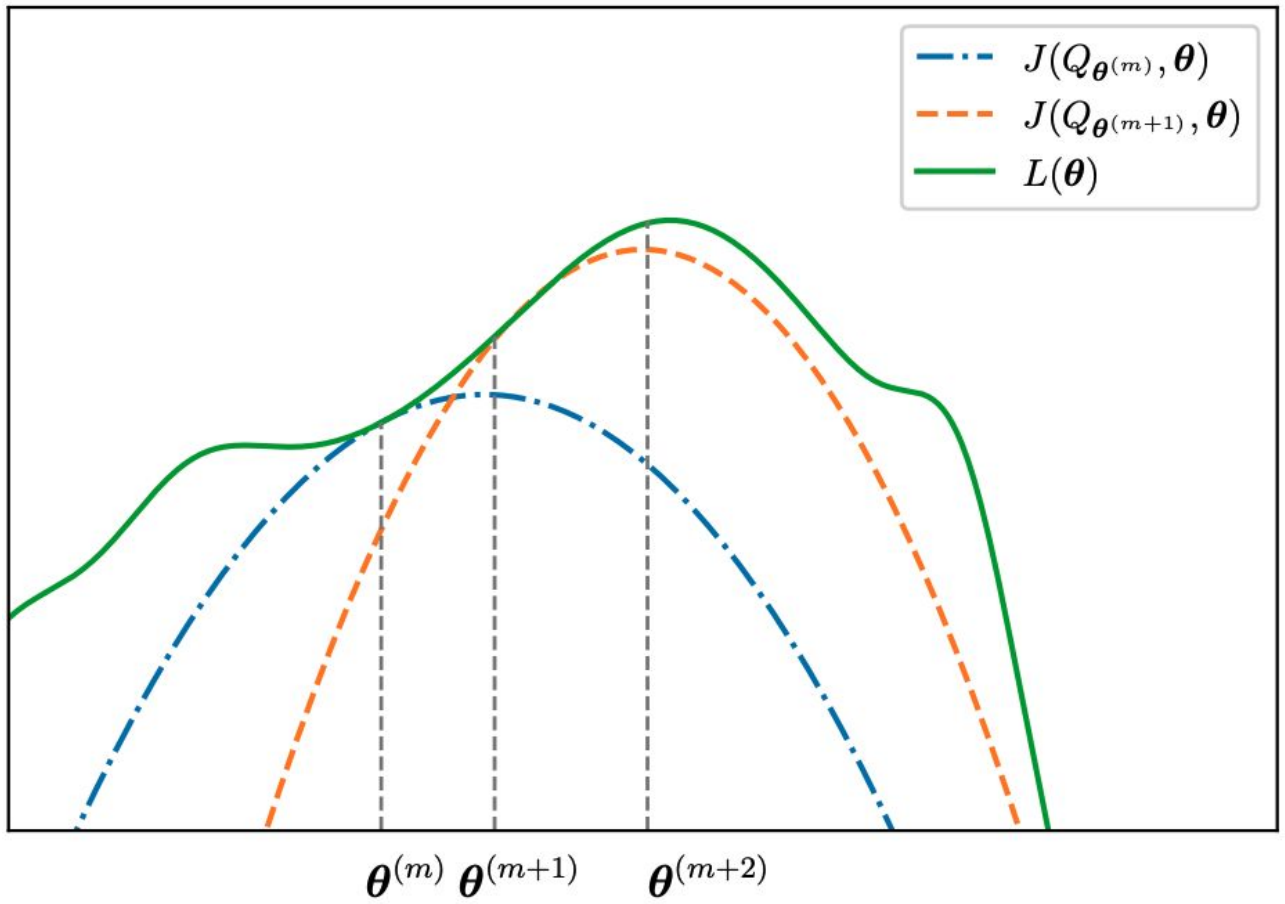
Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Auto-encoding Variational Bayes (AEVB) — Learning and Inference

Recall the EM algorithm:

Auto-encoding Variational Bayes (AEVB) — Learning and Inference

Recall the EM algorithm:

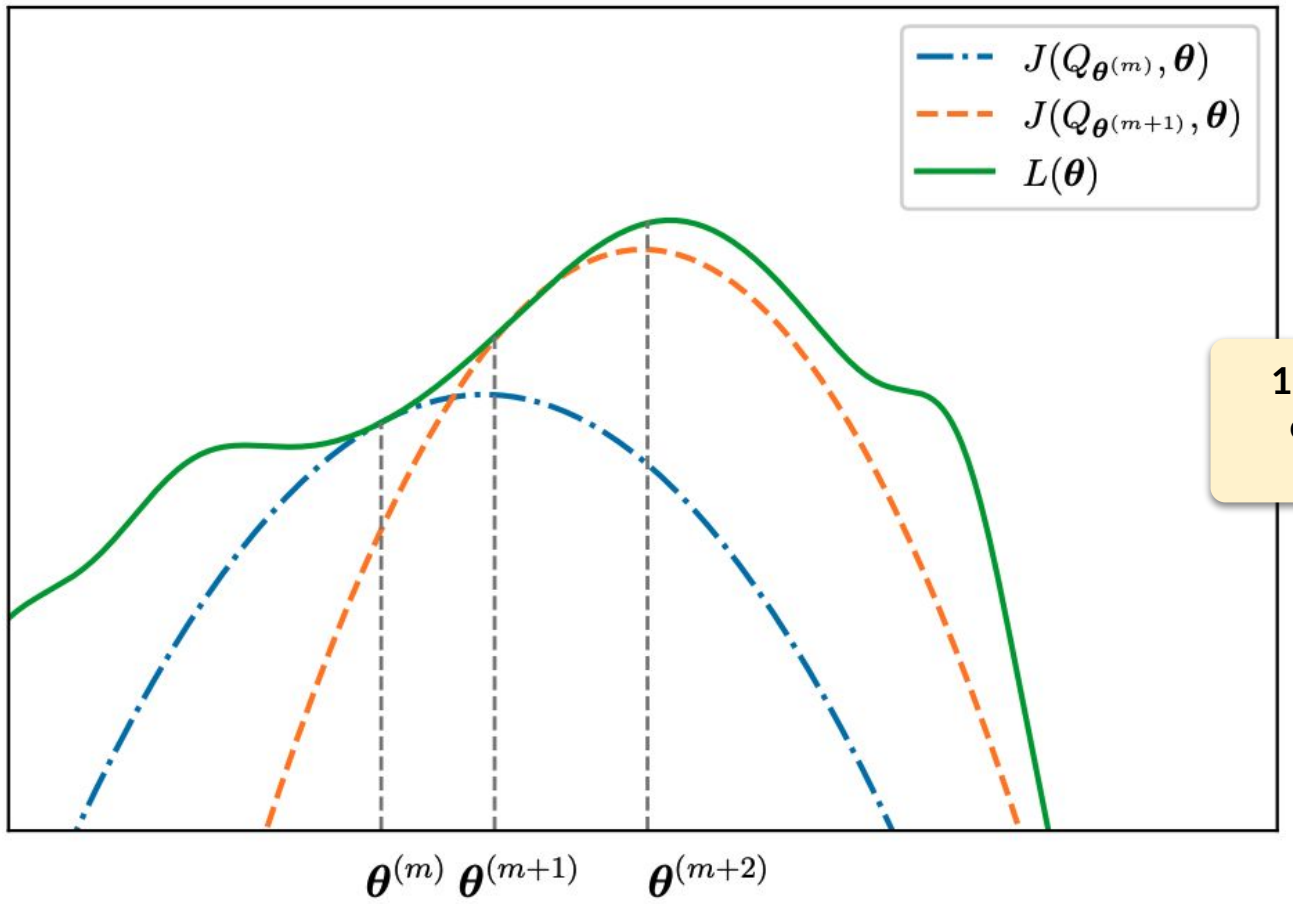


Iteratively forming then optimizing lower bounds to the (*marginal) likelihood* in latent-variable models.

Source: Inspired by Herman Kamper

Auto-encoding Variational Bayes (AEVB) — Learning and Inference

Recall the EM algorithm:



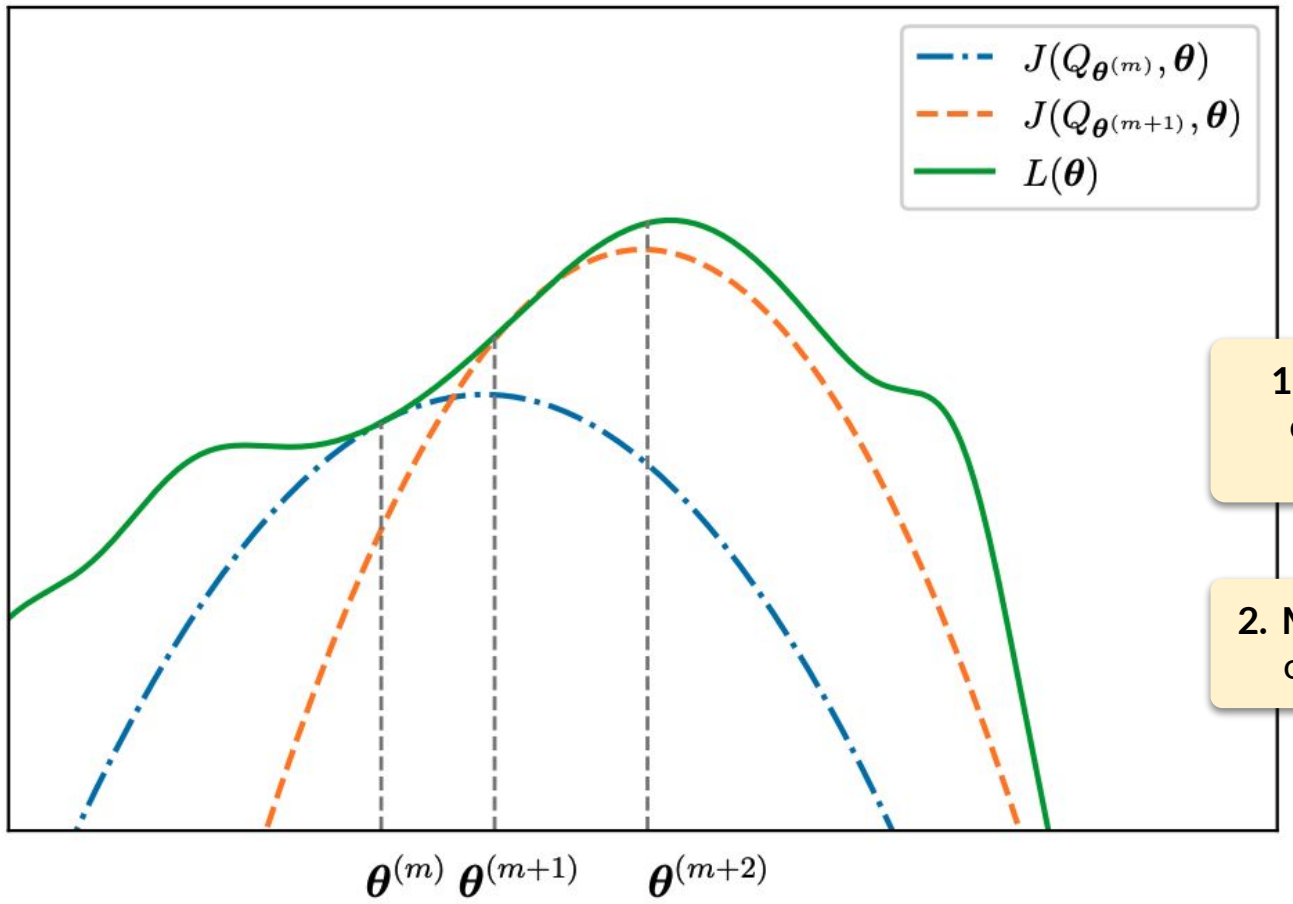
Iteratively forming then optimizing lower bounds to the (*marginal) likelihood* in latent-variable models.

1. **E-Step:** do inference to compute next objective (next lower bound)

Source: Inspired by Herman Kamper

Auto-encoding Variational Bayes (AEVB) — Learning and Inference

Recall the EM algorithm:



Iteratively forming then optimizing lower bounds to the (*marginal) likelihood* in latent-variable models.

1. **E-Step:** do inference to compute next objective (next lower bound)

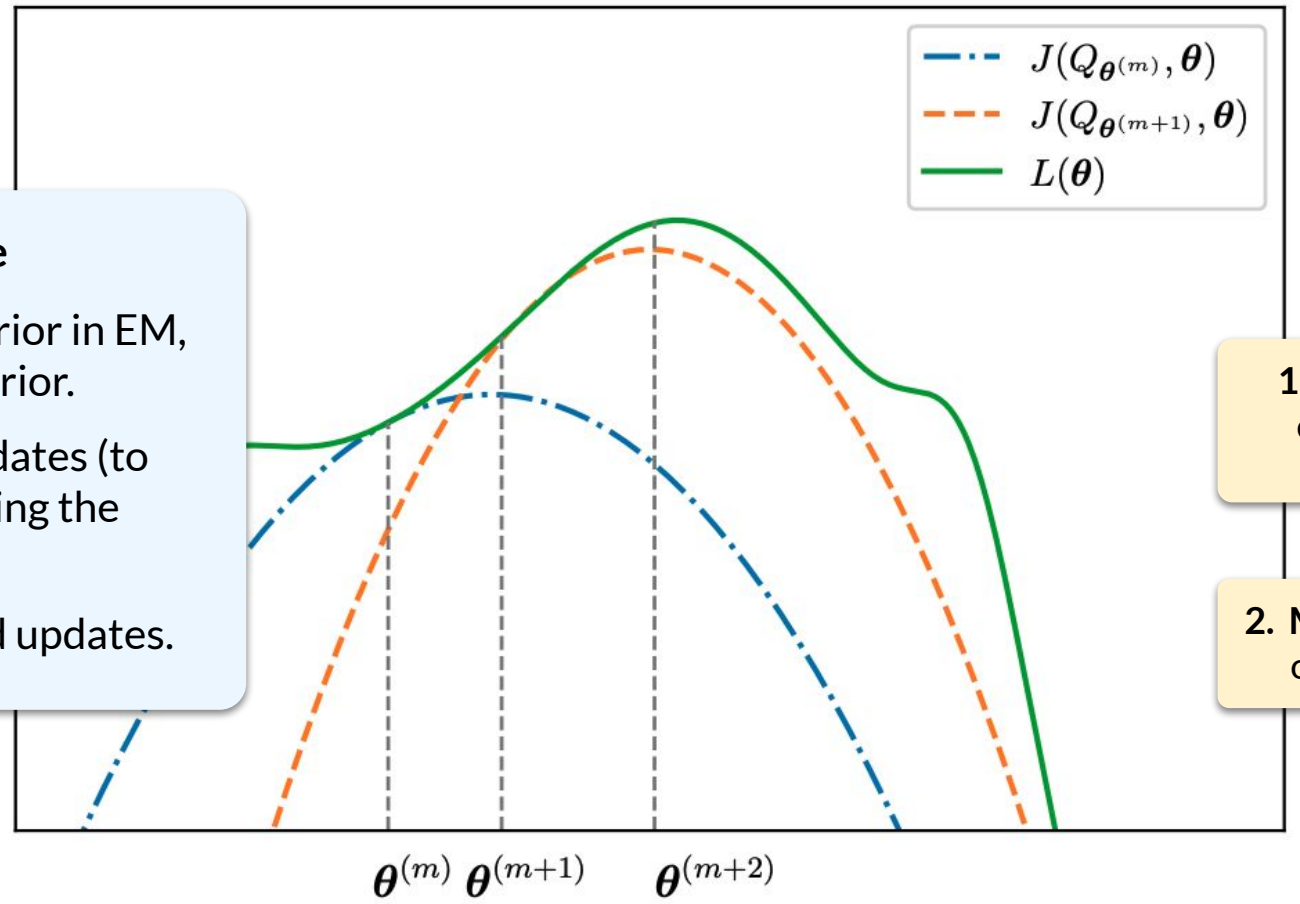
2. **M-Step:** maximize this new objective (lower bound).

Source: Inspired by Herman Kamper

Auto-encoding Variational Bayes (AEVB) — Learning and Inference

Recall the EM algorithm:

Iteratively forming then optimizing lower bounds to the (*marginal) likelihood* in latent-variable models.



Differences Here

1. Instead of an exact posterior in EM, using an approximate posterior.

2. Instead of coordinate updates (to convergence), jointly updating the objective and parameters.

3. And using gradient-based updates.

1. **E-Step:** do inference to compute next objective (next lower bound)

2. **M-Step:** maximize this new objective (lower bound).

Source: Inspired by Herman Kamper

Computing the Gradient

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

Dropping the conditioning of q on x for now to keep it general.

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

Note that an expression for this expectation in closed form is often *not possible*.

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

Note that an expression for this expectation in closed form is often *not possible*.

Instead, we could take gradient of a Monte Carlo estimate by sampling z from q_ϕ .

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

If we do this (gradient of MC estimate), note it is easy to write out the **gradient for p_θ:**

$$\nabla_\theta \mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right] = \mathbb{E}_{q_\phi(z)}\left[\nabla_\theta \log p_\theta(x,z)\right]$$

$$\approx \frac{1}{M}\sum_{m=1}^{M} \nabla_\theta \log p_\theta(x, z^{(m)})$$

where $z^{(m)} \sim q_\phi(z)$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

If we do this (gradient of MC estimate), note it is easy to write out the **gradient for p_θ:**

$$\nabla_\theta\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right] = \mathbb{E}_{q_\phi(z)}\left[\nabla_\theta \log p_\theta(x,z)\right]$$

$$\approx \frac{1}{M}\sum_{m=1}^{M} \nabla_\theta \log p_\theta(x, z^{(m)})$$

where $z^{(m)} \sim q_\phi(z)$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

If we do this (gradient of MC estimate), note it is easy to write out the **gradient for p_θ:**

$$\nabla_\theta\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right] = \mathbb{E}_{q_\phi(z)}\left[\nabla_\theta \log p_\theta(x,z)\right]$$

Just swap the gradient and expectation here.

$$\approx \frac{1}{M}\sum_{m=1}^{M} \nabla_\theta \log p_\theta(x, z^{(m)})$$

where $z^{(m)} \sim q_\phi(z)$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

If we do this (gradient of MC estimate), note it is easy to write out the **gradient for p_θ:**

$$\nabla_\theta\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right] = \mathbb{E}_{q_\phi(z)}\left[\nabla_\theta \log p_\theta(x,z)\right]$$

$$\approx \frac{1}{M}\sum_{m=1}^{M} \nabla_\theta \log p_\theta(x, z^{(m)})$$

where $z^{(m)} \sim q_\phi(z)$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

But it's harder to write out the **gradient for** q_ϕ **:**

$$\nabla_{\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right] = \ ??$$

Computing the Gradient

So to optimize the ELBO, we need to compute the gradient:

Dropping the conditioning of q on x for now to keep it general.

$$\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right]$$

But it's harder to write out the **gradient for** q_ϕ **:**

$$\nabla_{\phi}\mathbb{E}_{q_\phi(z)}\left[\log p_\theta(x,z) - \log q_\phi(z)\right] = \ ??$$

Note that we cannot just swap the gradient and expectation, because the expectation is being taken with respect to $q_\phi(z)$.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Score Function Gradient Estimator

One way to estimate this gradient is via the **score function estimator**:

The Score Function Gradient Estimator

One way to estimate this gradient is via the **score function estimator**:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)} \left[\log p_\theta(x, z) - \log q_\phi(z) \right] =$$

$$\mathbb{E}_{q_\phi(z)} \left[\left(\log p_\theta(x, z) - \log q_\phi(z) \right) \nabla_\phi \log q_\phi(z) \right]$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Score Function Gradient Estimator

One way to estimate this gradient is via the **score function estimator**:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)} \left[\log p_\theta(x, z) - \log q_\phi(z) \right] =$$

$$\mathbb{E}_{q_\phi(z)} \left[\left(\log p_\theta(x, z) - \log q_\phi(z) \right) \nabla_\phi \log q_\phi(z) \right]$$

Expectation with respect to q of…

The Score Function Gradient Estimator

One way to estimate this gradient is via the **score function estimator**:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)} \left[\log p_\theta(x, z) - \log q_\phi(z) \right] =$$

$$\mathbb{E}_{q_\phi(z)} \left[\left(\log p_\theta(x, z) - \log q_\phi(z) \right) \nabla_\phi \log q_\phi(z) \right]$$

Expectation with respect to q of…

Log joint minus log q (same as in the ELBO)

The Score Function Gradient Estimator

One way to estimate this gradient is via the **score function estimator**:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)} \left[\log p_\theta(x, z) - \log q_\phi(z) \right] =$$

$$\mathbb{E}_{q_\phi(z)} \left[\left(\log p_\theta(x, z) - \log q_\phi(z) \right) \nabla_\phi \log q_\phi(z) \right]$$

Expectation with respect to q of…

Log joint minus log q (same as in the ELBO)

Multiplied by the gradient of the log of q.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Score Function Gradient Estimator

One way to estimate this gradient is via the **score function estimator**:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)} \left[\log p_\theta(x, z) - \log q_\phi(z) \right] =$$

$$\mathbb{E}_{q_\phi(z)} \left[\left(\log p_\theta(x, z) - \log q_\phi(z) \right) \nabla_\phi \log q_\phi(z) \right]$$

Expectation with respect to q of…

Log joint minus log q (same as in the ELBO)

Multiplied by the gradient of the log of q.

Follows from some basic algebra/calculus, and takes about half a page to derive :D.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Score Function Gradient Estimator

One way to estimate this gradient is via the **score function estimator**:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)} \left[\log p_\theta(x, z) - \log q_\phi(z) \right] =$$

$$\mathbb{E}_{q_\phi(z)} \left[\left(\log p_\theta(x, z) - \log q_\phi(z) \right) \nabla_\phi \log q_\phi(z) \right]$$

Notably: since the gradient is now inside the expectation, we can evaluate this expectation using Monte Carlo methods!

Expectation with respect to q of…

Log joint minus log q (same as in the ELBO)

Multiplied by the gradient of the log of q.

Follows from some basic algebra/calculus, and takes about half a page to derive :D.

⇒ Now can estimate via Monte Carlo methods.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Score Function Gradient Estimator

One way to estimate this gradient is via the **score function estimator**:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)} \left[\log p_\theta(x, z) - \log q_\phi(z) \right]$$

$$= \mathbb{E}_{q_\phi(z)} \left[\left(\log p_\theta(x, z) - \log q_\phi(z) \right) \nabla_\phi \log q_\phi(z) \right]$$

$$= \frac{1}{M} \sum_{m=1}^{M} \left(\log p_\theta(x, z^{(m)}) - \log q_\phi(z^{(m)}) \right) \nabla_\phi \log q_\phi(z^{(m)}),$$

where $z^{(m)} \sim q_\phi(z)$

Notably: since the gradient is now inside the expectation, we can evaluate this expectation using Monte Carlo methods!

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Score Function Gradient Estimator

One way to estimate this gradient is via the **score function estimator**:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)} \left[\log p_\theta(x, z) - \log q_\phi(z) \right]$$

$$= \mathbb{E}_{q_\phi(z)} \left[\left(\log p_\theta(x, z) - \log q_\phi(z) \right) \nabla_\phi \log q_\phi(z) \right]$$

$$= \frac{1}{M} \sum_{m=1}^{M} \left(\log p_\theta(x, z^{(m)}) - \log q_\phi(z^{(m)}) \right) \nabla_\phi \log q_\phi(z^{(m)}),$$

where $z^{(m)} \sim q_\phi(z)$

Notably: since the gradient is now inside the expectation, we can evaluate this expectation using Monte Carlo methods!

Monte Carlo estimate of the gradient

The Score Function Gradient Estimator

However, there is an issue:

The Score Function Gradient Estimator

However, there is an issue:

- The score-function estimator has an important shortcoming: it has **high variance**.

The Score Function Gradient Estimator

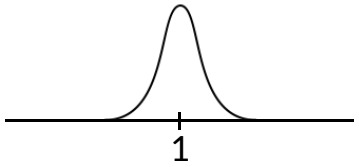
However, there is an issue:

- The score-function estimator has an important shortcoming: it has **high variance**.
- What does this mean?
    - Suppose we are using Monte Carlo to estimate a quantity with expected value = 1.

The Score Function Gradient Estimator

However, there is an issue:

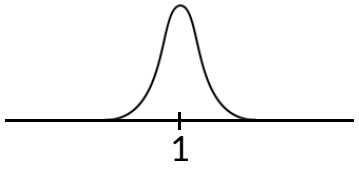
- The score-function estimator has an important shortcoming: it has **high variance**.
- What does this mean?
  - Suppose we are using Monte Carlo to estimate a quantity with expected value = 1.
  - If our samples are: *0.9, 1.1, 0.96, 1.05* (*i.e.*, close to 1), then after a few samples we'll get a good estimate of the true expected value.



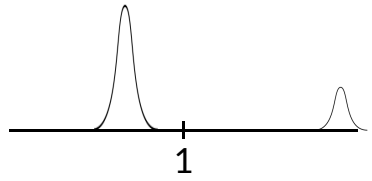
The Score Function Gradient Estimator

However, there is an issue:

- The score-function estimator has an important shortcoming: it has **high variance**.
- What does this mean?
  - Suppose we are using Monte Carlo to estimate a quantity with expected value = 1.
  - If our samples are: *0.9, 1.1, 0.96, 1.05* (*i.e.*, close to 1), then after a few samples we'll get a good estimate of the true expected value.



  - If our samples are: zero *99%* of the time, and one-hundred *1%* of the time ⇒ expected value is correct, but estimate is worse! (You need many more samples).

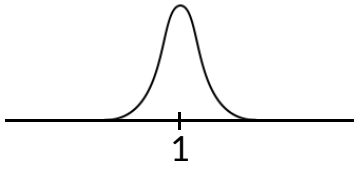


Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

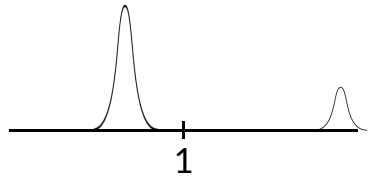
The Score Function Gradient Estimator

However, there is an issue:

- The score-function estimator has an important shortcoming: it has **high variance**.
- What does this mean?
  - Suppose we are using Monte Carlo to estimate a quantity with expected value = 1.
  - If our samples are: *0.9, 1.1, 0.96, 1.05* (*i.e.*, close to 1), then after a few samples we'll get a good estimate of the true expected value.



  - If our samples are: zero *99%* of the time, and one-hundred *1%* of the time ⇒ expected value is correct, but estimate is worse! (You need many more samples).



- This latter issue is an example of a high variance MC estimator.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

The Score Function Gradient Estimator → SGVB Estimator

One key contribution of the VAE paper is to propose an alternative estimator, which is better behaved (lower variance).

The Score Function Gradient Estimator → SGVB Estimator

One key contribution of the VAE paper is to propose an alternative estimator, which is better behaved (lower variance).

Sometimes referred to as the *stochastic gradient variational Bayes* (SGVB) estimator.

The Score Function Gradient Estimator → SGVB Estimator

One key contribution of the VAE paper is to propose an alternative estimator, which is better behaved (lower variance).

Sometimes referred to as the *stochastic gradient variational Bayes* (SGVB) estimator.

This is done in two steps:

(1) We reformulate the ELBO so that parts of it can be computed in closed form (*i.e.*, without Monte Carlo).

(2) Then we use an alternative gradient estimator (SGVB) based on the so-called **reparameterization trick**.

Reformulating the ELBO

The reformulation of the ELBO is as follows:

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z) \right] - \mathrm{KL} \left[q_\phi(z \mid x) || p_\theta(z) \right]$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

ELBO (lower bound on the evidence), consisting of two terms:

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

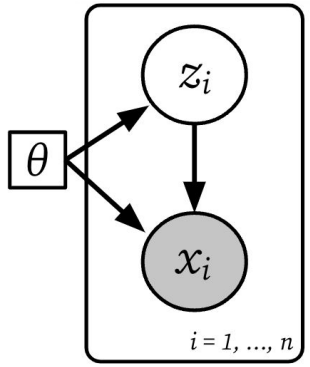
Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*



We know the log-likelihood (typically chosen when we define our probabilistic model).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

Reformulating the ELBO

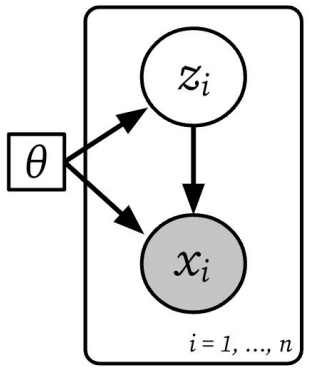
The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*



The prior is also known/chosen when we first define our model.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

(Can verify this using some algebra on the previous ELBO formulation).

Reformulating the ELBO

The reformulation of the ELBO is as follows:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

This has an interesting interpretation!

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

First:

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

First:

- Think of x as an observed data point.

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence | Expectation of the *log-likelihood* | KL between q *posterior* and p *prior*

First:

- Think of x as an observed data point.
- Think of $z \sim q_\phi(z \mid x)$ as a *"code"* that describes *x*.
  - (Note that both terms in the right hand side involve an expectation with respect to q.)

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence | Expectation of the *log-likelihood* | KL between q *posterior* and p *prior*

First:

- Think of x as an observed data point.
- Think of $z \sim q_\phi(z \mid x)$ as a *"code"* that describes *x*.
  - (Note that both terms in the right hand side involve an expectation with respect to q.)
- ⇒ We will therefore call q_ϕ the **encoder**.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

Next:

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence | Expectation of the *log-likelihood* | KL between q *posterior* and p *prior*

Next:

encoder

- The first term is the log-likelihood of an x, given the code $z \sim q_\phi(z \mid x)$ conditioned on x.

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x)||p_\theta(z)\right]$$

Evidence | Expectation of the *log-likelihood* | KL between q *posterior* and p *prior*

Next:

encoder

- The first term is the log-likelihood of an x, given the code $z \sim q_\phi(z \mid x)$ conditioned on x.
- ⇒ this first term is maximized when $p_\theta(x \mid z)$ assigns high probability to this x.

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence | Expectation of the *log-likelihood* | KL between q *posterior* and p *prior*

Next:

encoder

- The first term is the log-likelihood of an x, given the code $z \sim q_\phi(z \mid x)$ conditioned on x.
- ⇒ this first term is maximized when $p_\theta(x \mid z)$ assigns high probability to this x.
- ⇒ *i.e.*, It is trying to *reconstruct x given the code z (drawn from $q_\phi(z \mid x)$).*

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence | Expectation of the *log-likelihood* | KL between q *posterior* and p *prior*

Next:

encoder

- The first term is the log-likelihood of an x, given the code $z \sim q_\phi(z \mid x)$ conditioned on x.
- ⇒ this first term is maximized when $p_\theta(x \mid z)$ assigns high probability to this x.
- ⇒ *i.e.*, It is trying to *reconstruct x given the code z (drawn from $q_\phi(z \mid x)$).*
- ⇒ We will therefore call $p_\theta(x \mid z)$ the **decoder**.
  - And the first term we call the *reconstruction error.*

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

Also:

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

Also:

- The second term is the KL divergence between $q_\phi(z \mid x)$ and the model's prior $p_\theta(z)$.

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

Also:

- The second term is the KL divergence between $q_\phi(z \mid x)$ and the model's prior $p_\theta(z)$.
- (In this model, we will assume the prior on z is a unit Normal distribution). $\mathcal{N}(0, I)$
  - ⇒ this encourages the codes z to look Gaussian distributed.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

Also:

- The second term is the KL divergence between $q_\phi(z \mid x)$ and the model's prior $p_\theta(z)$.
- (In this model, we will assume the prior on z is a unit Normal distribution). $\mathcal{N}(0, I)$
  - ⇒ this encourages the codes z to look Gaussian distributed.
- We call this second term the *regularization term*.
  - It prevents $q_\phi(z \mid x)$ from, *e.g.*, simply encoding an identity mapping.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

In summary:

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

In summary:

When we optimize this ELBO, we are trying to fit a $q_\phi(z \mid x)$ that will map an observation x onto a useful latent variable z …

… and also to fit a $p_\theta(x \mid z)$ which reconstructs this x, given the latent z.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reformulating the ELBO

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Evidence | Expectation of the *log-likelihood* | KL between q *posterior* and p *prior*

In summary:

When we optimize this ELBO, we are trying to fit a $q_\phi(z \mid x)$ that will map an observation x onto a useful latent variable z …

… and also to fit a $p_\theta(x \mid z)$ which reconstructs this x, given the latent z.

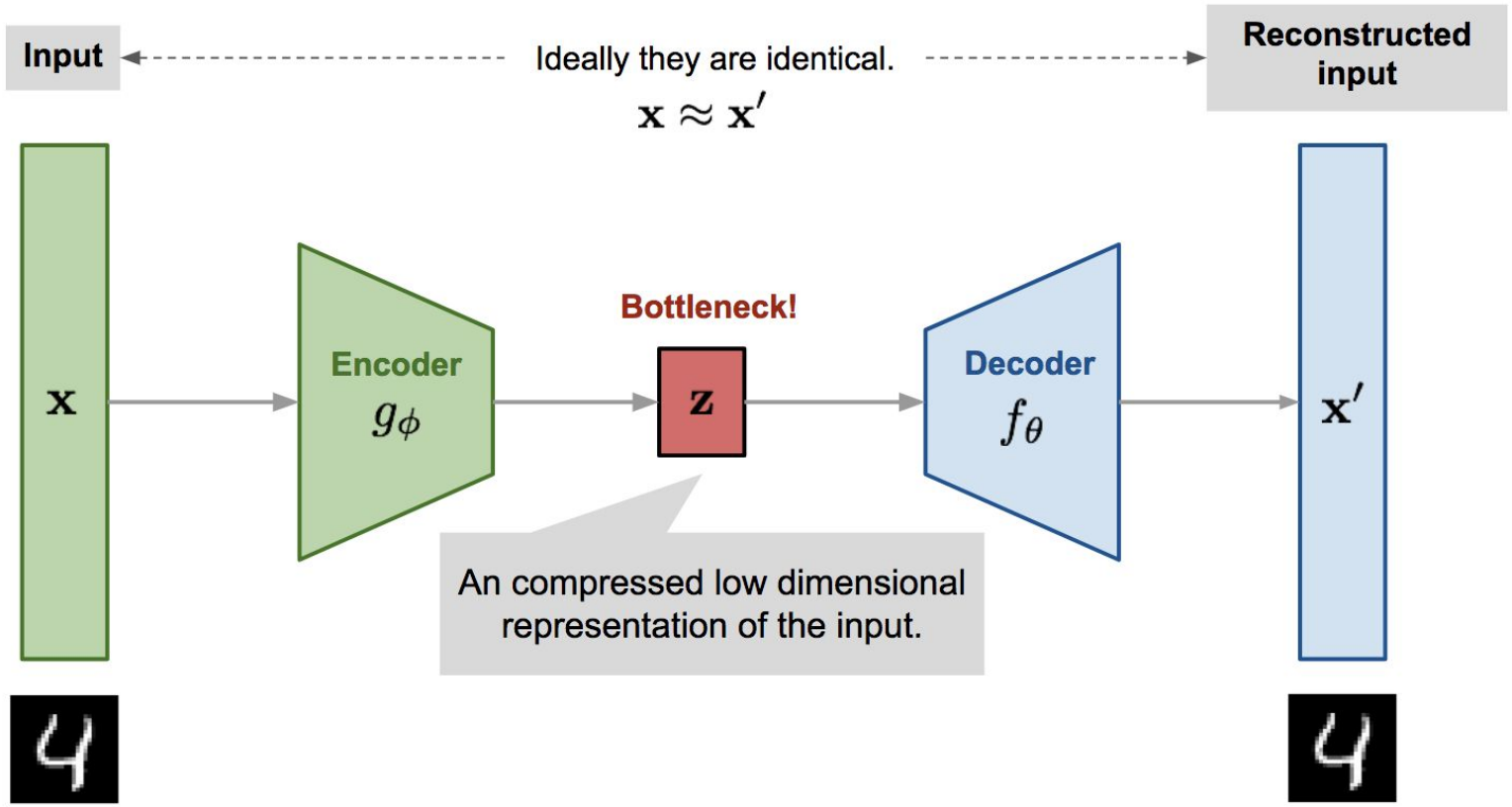
⇒ This is reminiscent of an *autoencoder neural network*.

(*i.e.*, a pair of neural networks, which encode/decode data, aiming to minimize the reconstruction error).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reconstruction Term ⇔ Autoencoder

Aside — illustration of an **autoencoder** (first developed in 1990s, or even late 1980s).



Source: Lil'Log, "From Autoencoder to Beta-VAE"

Reparameterization Trick

So how can we use this reformulation to form a *lower-variance gradient estimator*, and then optimize the ELBO?

Reparameterization Trick

So how can we use this reformulation to form a *lower-variance gradient estimator*, and then optimize the ELBO?

We will use something call the **reparameterization trick**!

Reparameterization Trick

So how can we use this reformulation to form a *lower-variance gradient estimator*, and then optimize the ELBO?

We will use something call the **reparameterization trick**!

Recall:

SGVB Estimator

This is done in two steps:

(1) We reformulate the ELBO ✅ .

(2) Then we use an alternative gradient estimator (SGVB) based on the so-called **reparameterization trick**.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Under certain mild conditions, we can express $q_\phi(z \mid x)$ via the following two-step generative process:

Reparameterization Trick

Under certain mild conditions, we can express $q_\phi(z \mid x)$ via the following two-step generative process:

- First sample a noise variable ϵ from a simple distribution $p(\epsilon)$ (such as a standard Normal $\mathcal{N}(0, I)$):

$$\epsilon \sim p(\epsilon)$$

Note: this does not depend on ϕ.

Reparameterization Trick

Under certain mild conditions, we can express $q_\phi(z \mid x)$ via the following two-step generative process:

- First sample a noise variable ϵ from a simple distribution $p(\epsilon)$ (such as a standard Normal $\mathcal{N}(0, I)$):

$$\epsilon \sim p(\epsilon)$$

Note: this does not depend on ϕ.

- Then we apply a deterministic transformation $g_\phi(\epsilon, x)$ that maps the random noise into a more-complex distribution:

$$z = g_\phi(\epsilon, x)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Note: for many interesting classes of q_ϕ, it will be possible to define a g_ϕ such that $z = g_\phi(\epsilon, x)$ is distributed according to $q_\phi(z \mid x)$.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Note: for many interesting classes of q_ϕ, it will be possible to define a g_ϕ such that $z = g_\phi(\epsilon, x)$ is distributed according to $q_\phi(z \mid x)$.

Graphical model of $q_\phi(z \mid x)$ can be viewed as similar to…

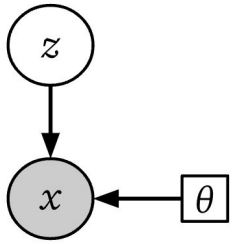
(recall from first PGM lecture)

Famous Bayesian Networks — Neural Transformations in PGMs

Can incorporate more-complex function transformations (or neural networks) into Bayesian networks!

But instead we could do:

$$z \sim p(z) = \mathcal{N}(0, 1)$$
$$x \sim p_\theta(x \mid z) = \mathcal{N}(g_\theta(z), 1)$$



$g_\theta(z)$ might be a complex fixed/known, or have parameters that we learn (*e.g.*, if we use a neural network!).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

We refer to this as a ***reparameterization trick*** (we are reparameterizing, *i.e.*, writing out $q_\phi(z \mid x)$ in a different way).

Reparameterization Trick

We refer to this as a ***reparameterization trick*** (we are reparameterizing, *i.e.*, writing out $q_\phi(z \mid x)$ in a different way).

Simplest example of the reparameterization trick:

Reparameterization Trick

We refer to this as a ***reparameterization trick*** (we are reparameterizing, *i.e.*, writing out $q_\phi(z \mid x)$ in a different way).

Simplest example of the reparameterization trick:

Parameter ϕ here is: (μ, σ).

- Instead of writing: $z \sim q_{\mu,\sigma}(z) = \mathcal{N}(z; \mu, \sigma^2)$

Reparameterization Trick

We refer to this as a ***reparameterization trick*** (we are reparameterizing, *i.e.*, writing out $q_\phi(z \mid x)$ in a different way).

Simplest example of the reparameterization trick:

Parameter ϕ here is: (μ, σ).

- Instead of writing: $z \sim q_{\mu,\sigma}(z) = \mathcal{N}(z; \mu, \sigma^2)$
- We can write: $z = g_{\mu,\sigma}(\epsilon) = \mu + \epsilon \cdot \sigma, \quad \text{where} \;\; \epsilon \sim \mathcal{N}(\epsilon\,;\, 0, 1)$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Main advantages of the reparameterization trick:

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Main advantages of the reparameterization trick:

We can now write the gradient of an expectation (gradient wrt ϕ, expectation of q), for a given test function f, as:

Reparameterization Trick

Main advantages of the reparameterization trick:

We can now write the gradient of an expectation (gradient wrt ϕ, expectation of q), for a given test function f, as:

$$\nabla_\phi \mathbb{E}_{z\sim q_\phi(z|x)}\left[f(x,z)\right] = \nabla_\phi \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[f(x, g_\phi(\epsilon,x))\right]$$

$$= \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[\nabla_\phi f(x, g_\phi(\epsilon,x))\right]$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Main advantages of the reparameterization trick:

We can now write the gradient of an expectation (gradient wrt ϕ, expectation of q), for a given test function f, as:

Replace q with $p(\epsilon)$, and z with $g(\epsilon, x)$.

$$\nabla_\phi \mathbb{E}_{z \sim q_\phi(z|x)} [f(x, z)] = \nabla_\phi \mathbb{E}_{\epsilon \sim p(\epsilon)} [f(x, g_\phi(\epsilon, x))]$$

$$= \mathbb{E}_{\epsilon \sim p(\epsilon)} [\nabla_\phi f(x, g_\phi(\epsilon, x))]$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Main advantages of the reparameterization trick:

We can now write the gradient of an expectation (gradient wrt ϕ, expectation of q), for a given test function f, as:

$$\nabla_\phi \mathbb{E}_{z\sim q_\phi(z|x)}\left[f(x,z)\right] = \nabla_\phi \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[f(x, g_\phi(\epsilon,x))\right]$$

$$= \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[\nabla_\phi f(x, g_\phi(\epsilon,x))\right]$$

Replace q with $p(\epsilon)$, and z with $g(\epsilon, x)$.

Can now swap gradient and expectation!

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Reparameterization Trick

Main advantages of the reparameterization trick:

We can now write the gradient of an expectation (gradient wrt ϕ, expectation of q), for a given test function f, as:

$$\nabla_\phi \mathbb{E}_{z\sim q_\phi(z|x)}\left[f(x,z)\right] = \nabla_\phi \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[f(x, g_\phi(\epsilon,x))\right]$$

$$= \mathbb{E}_{\epsilon\sim p(\epsilon)}\left[\nabla_\phi f(x, g_\phi(\epsilon,x))\right]$$

Replace q with $p(\epsilon)$, and z with $g(\epsilon, x)$.

Can now swap gradient and expectation!

Can now take a Monte Carlo estimate of this expectation.

⇒ yields an estimate of the gradient for ϕ, with much lower variance than the score-function estimator.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

What to choose for p and q?

What to choose for p and q?

Suppose, after learning θ, we hope to use $p_\theta(x \mid z)$ as a generative model.

(And we hope to use $q_\phi(z \mid x)$ to perform inference in order to learn θ).

What to choose for p and q?

Suppose, after learning θ, we hope to use $p_\theta(x \mid z)$ as a generative model.

(And we hope to use $q_\phi(z \mid x)$ to perform inference in order to learn θ).

Question: how should we specify the form of p_θ and q_ϕ ?

What to choose for p and q?

Suppose, after learning θ, we hope to use $p_\theta(x \mid z)$ as a generative model.

(And we hope to use $q_\phi(z \mid x)$ to perform inference in order to learn θ).

Question: how should we specify the form of p_θ and q_ϕ ?

Note:

What to choose for p and q?

Suppose, after learning θ, we hope to use $p_\theta(x \mid z)$ as a generative model.

(And we hope to use $q_\phi(z \mid x)$ to perform inference in order to learn θ).

Question: how should we specify the form of p_θ and q_ϕ ?

Note:

- We want $p_\theta(x \mid z)$ to be flexible enough to represent the richness of complex/high-dimensional data.

What to choose for p and q?

Suppose, after learning θ, we hope to use $p_\theta(x \mid z)$ as a generative model.

(And we hope to use $q_\phi(z \mid x)$ to perform inference in order to learn θ).

Question: how should we specify the form of p_θ and q_ϕ ?

Note:

- We want $p_\theta(x \mid z)$ to be flexible enough to represent the richness of complex/high-dimensional data.
- We want $q_\phi(z \mid x)$ to be flexible enough to approximate $p_\theta(z \mid x)$ (*i.e.*, the true posterior) well.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

What to choose for p and q?

Answer: in VAEs, we will parameterize p_θ and q_ϕ using neural networks. 

What to choose for p and q?

Answer: in VAEs, we will parameterize p_θ and q_ϕ using neural networks. 

As an example, we can define $q_\phi(z \mid x)$ to be:

$$q_\phi(z \mid x) = \mathcal{N}\left(z\ ;\ \mu_\phi(x), \mathrm{diag}(\sigma_\phi(x))^2\right)$$

Where $\mu_\phi(x)$ and $\sigma_\phi(x)$ are vector-valued functions of x parametrized by an arbitrary neural network.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

What to choose for p and q?

Answer: in VAEs, we will parameterize p_θ and q_ϕ using neural networks. 

As an example, we can define $q_\phi(z \mid x)$ to be:

$$q_\phi(z \mid x) = \mathcal{N}\left(z\ ;\ \mu_\phi(x), \mathrm{diag}(\sigma_\phi(x))^2\right)$$

Where $\mu_\phi(x)$ and $\sigma_\phi(x)$ are vector-valued functions of x parametrized by an arbitrary neural network.

⇒ could call them: "*mean network*" and "*standard deviation network*".

What to choose for p and q?

Answer: in VAEs, we will parameterize p_θ and q_ϕ using neural networks.

As an example, we can define $q_\phi(z \mid x)$ to be:

$$q_\phi(z \mid x) = \mathcal{N}\left(z\,;\ \mu_\phi(x), \mathrm{diag}(\sigma_\phi(x))^2\right)$$

More generally, can be applied to any exponential family distribution by parameterizing the sufficient statistics by a function of x.

Where $\mu_\phi(x)$ and $\sigma_\phi(x)$ are vector-valued functions of x parametrized by an arbitrary neural network.

⇒ could call them: "*mean network*" and "*standard deviation network*".

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

So what is the variational autoencoder (VAE), and how does it relate to the AEVB algorithm?

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

First, to summarize — the AEVB algorithm is simply the combination of:

Variational Autoencoder (VAE)

First, to summarize — the AEVB algorithm is simply the combination of:

(1) The alternative ELBO reformulation.

$$\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x)||p_\theta(z)\right]$$

Expectation of the *log-likelihood*

KL between q *posterior* and p *prior*

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

First, to summarize — the AEVB algorithm is simply the combination of:

(1) The alternative ELBO reformulation.

$$\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x)||p_\theta(z)\right]$$

Expectation of the *log-likelihood* — KL between q *posterior* and p *prior*

(2) The black-box (gradient-based) variational inference approach.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

First, to summarize — the AEVB algorithm is simply the combination of:

(1) The alternative ELBO reformulation.

$$\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Expectation of the *log-likelihood* — KL between q *posterior* and p *prior*

(2) The black-box (gradient-based) variational inference approach.

(3) The reparameterization-based SGVB (low-variance) estimator.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

First, to summarize — the AEVB algorithm is simply the combination of:

(1) The alternative ELBO reformulation.

$$\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Expectation of the *log-likelihood* — KL between *q posterior* and *p prior*

(2) The black-box (gradient-based) variational inference approach.

(3) The reparameterization-based SGVB (low-variance) estimator.

⇒ takes gradient steps on θ and ϕ to optimize the alternative ELBO.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

The model p_θ is parameterized as:

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

The model p_θ is parameterized as:

$$p_\theta(z) = \mathcal{N}(z;\ 0, I)$$
$$p_\theta(x \mid z) = \mathcal{N}\left(x;\ \mu_\theta(z), \mathrm{diag}(\sigma_\theta(z))^2\right)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

The model p_θ is parameterized as:

Prior on latent variable z is a Gaussian.

$$p_\theta(z) = \mathcal{N}(z;\ 0, I)$$

$$p_\theta(x \mid z) = \mathcal{N}\left(x;\ \mu_\theta(z), \mathrm{diag}(\sigma_\theta(z))^2\right)$$

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

The model p_θ is parameterized as:

Prior on latent variable z is a Gaussian.

$$p_\theta(z) = \mathcal{N}(z;\ 0, I)$$

$$p_\theta(x \mid z) = \mathcal{N}\left(x;\ \mu_\theta(z), \mathrm{diag}(\sigma_\theta(z))^2\right)$$

Decode latent variable z into data point x.
(Also Gaussian).

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

The posterior approximation q_ϕ is parameterized as:

Variational Autoencoder (VAE)

The **VAE** is a special case — using the AEVB algorithm for a specific choice of p and q.

The posterior approximation q_ϕ is parameterized as:

$$q_\phi(z \mid x) = \mathcal{N}\left(z;\ \mu_\phi(x), \mathrm{diag}(\sigma_\phi(x))^2\right)$$

Encode data point x as latent variable z.
(Also Gaussian).

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

Useful properties of this choice of p_θ and q_ϕ :

Variational Autoencoder (VAE)

Useful properties of this choice of p_θ and q_ϕ :

- Closed-form expression to compute the regularization term (2nd term)

$$\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Expectation of the *log-likelihood* | KL between q *posterior* and p *prior*

  - KL divergence of two Gaussians ⇒ closed form expression and gradient!

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

Useful properties of this choice of p_θ and q_ϕ :

- Closed-form expression to compute the regularization term (2nd term)

$$\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}\left[q_\phi(z \mid x) || p_\theta(z)\right]$$

Expectation of the *log-likelihood* | KL between q *posterior* and p *prior*

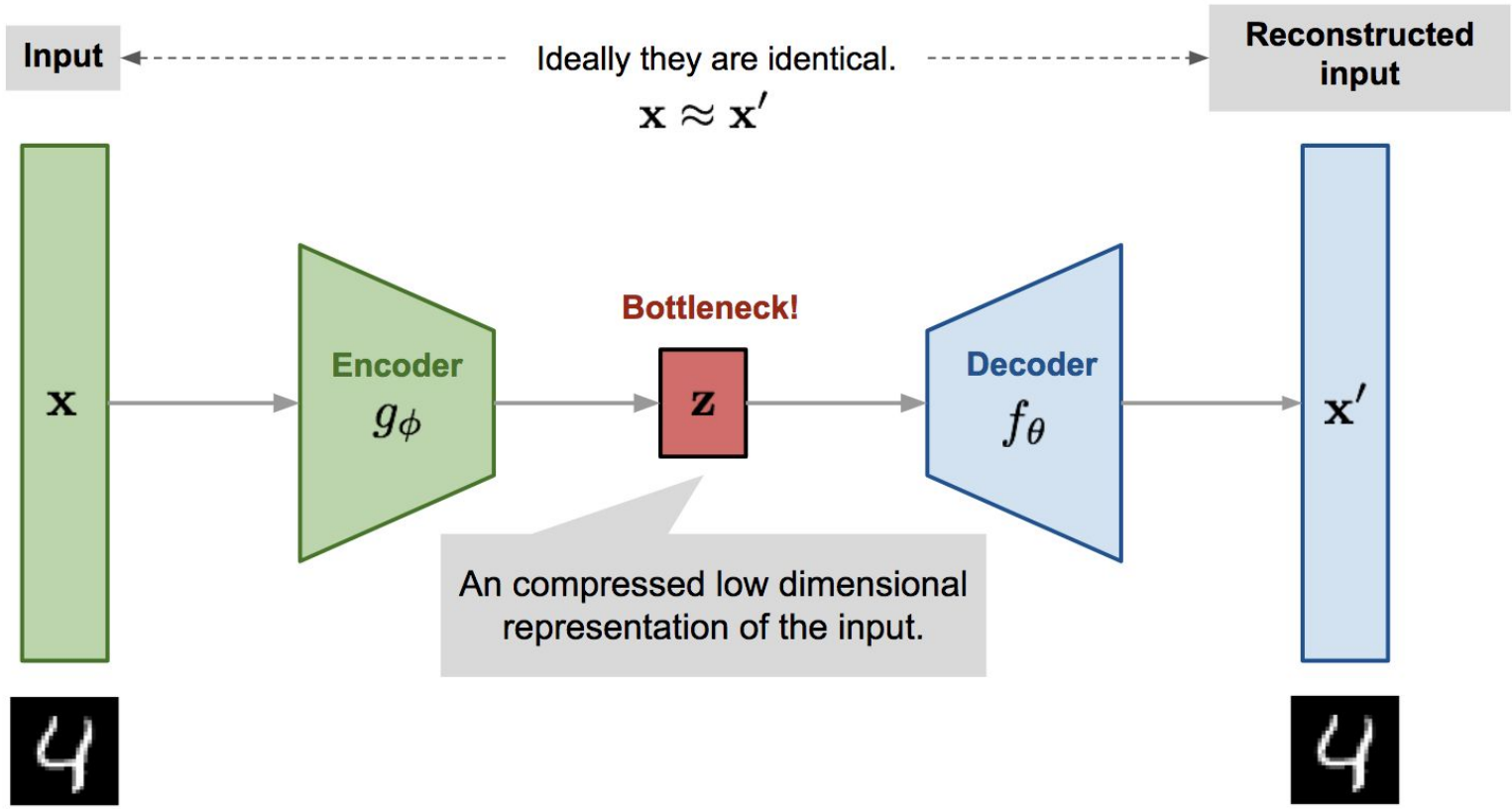
  - KL divergence of two Gaussians ⇒ closed form expression and gradient!

- So we only need to perform Monte Carlo gradient step on the 1st term.
  - And can carry out SGVB estimator using neural network reparameterization trick as described above.

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

Visual summary — back to the *autoencoder*:

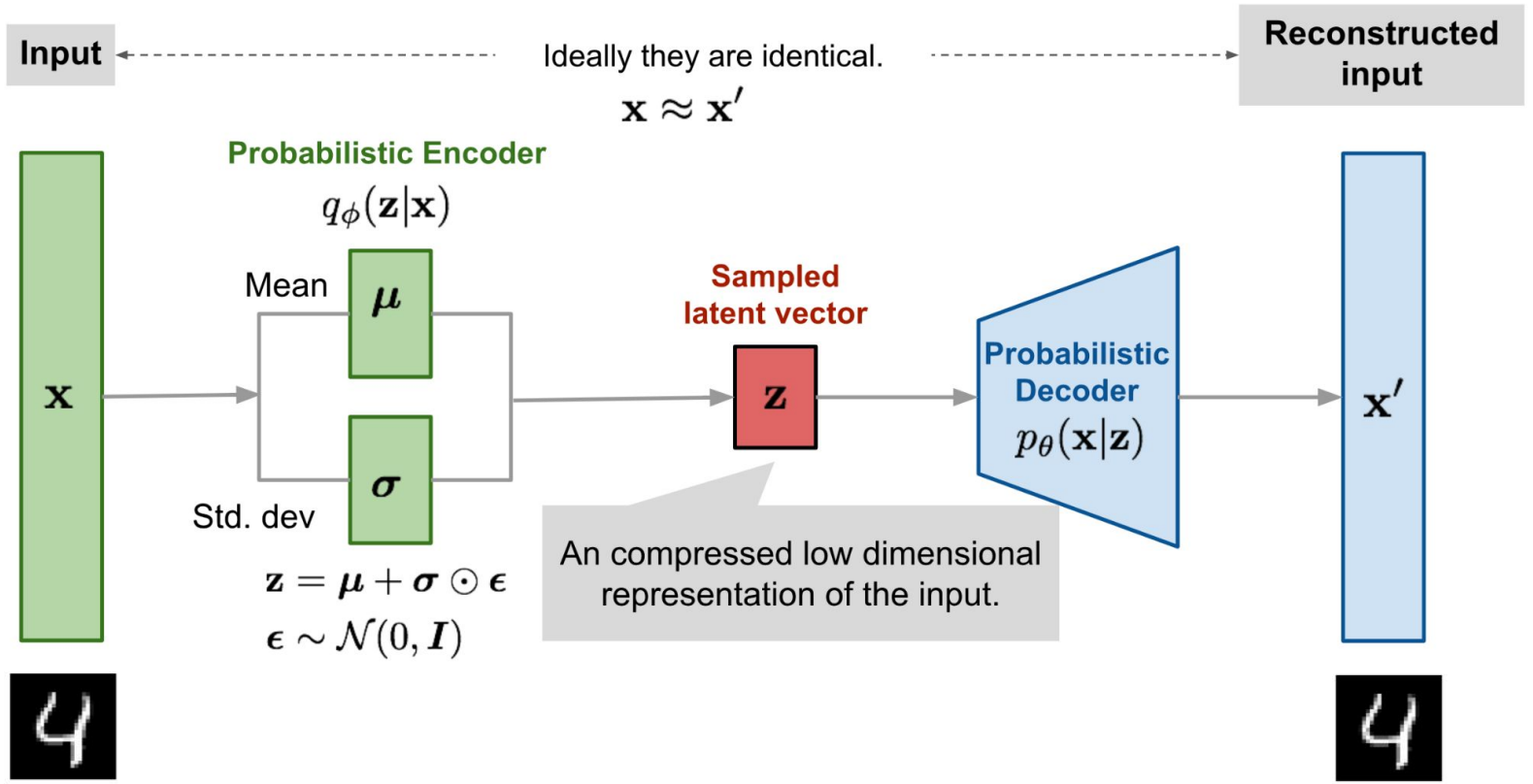


Source: Lil'Log, "From Autoencoder to Beta-VAE"

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE)

Visual summary — instead, a ***variational autoencoder***:



Source: Lil'Log, "From Autoencoder to Beta-VAE"

Source: Stefano Ermon, Probabilistic Graphical Models (CS228) Class

Variational Autoencoder (VAE) — Results

So does it work?

Variational Autoencoder (VAE) — Results

So does it work? → Results from the original paper (*ICLR 2014*)

Variational Autoencoder (VAE) — Results

So does it work? → Results from the original paper (*ICLR 2014*)

Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions are two-fold. First, we show that a reparameterization of the variational lower bound yields a lower bound estimator that can be straightforwardly optimized using standard stochastic gradient methods. Second, we show that for i.i.d. datasets with continuous latent variables per datapoint, posterior inference can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator. Theoretical advantages are reflected in experimental results.

1 Introduction

How can we perform efficient approximate inference and learning with directed probabilistic models whose continuous latent variables and/or parameters have intractable posterior distributions? The variational Bayesian (VB) approach involves the optimization of an approximation to the intractable posterior. Unfortunately, the common mean-field approach requires analytical solutions of expectations w.r.t. the approximate posterior, which are also intractable in the general case. We show how a reparameterization of the variational lower bound yields a simple differentiable unbiased estimator of the lower bound; this SGVB (Stochastic Gradient Variational Bayes) estimator can be used for efficient approximate posterior inference in almost any model with continuous latent variables and/or parameters, and is straightforward to optimize using standard stochastic gradient ascent techniques.

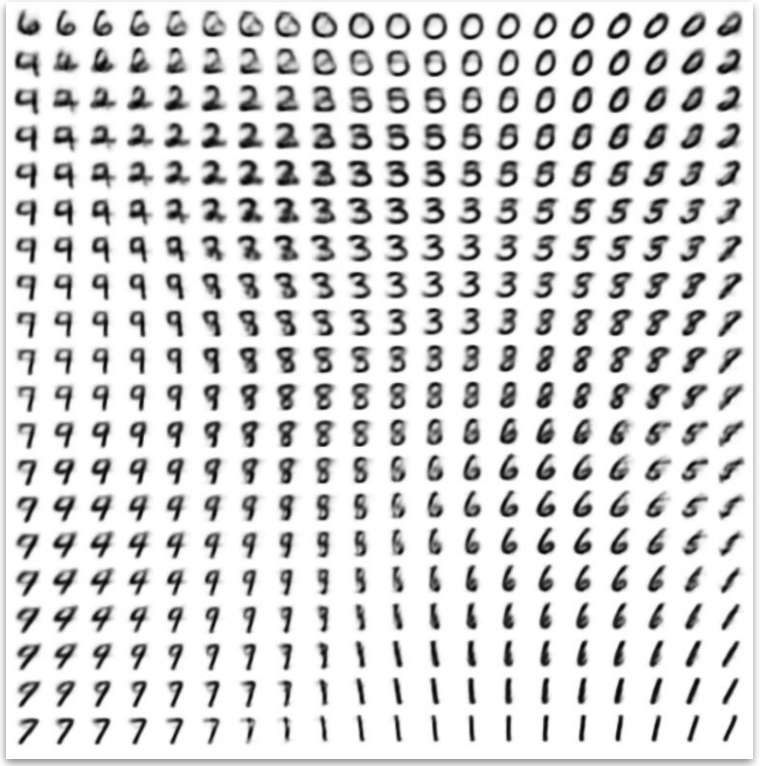
For the case of an i.i.d. dataset and continuous latent variables per datapoint, we propose the Auto-Encoding VB (AEVB) algorithm. In the AEVB algorithm we make inference and learning especially efficient by using the SGVB estimator to optimize a recognition model that allows us to perform very efficient approximate posterior inference using simple ancestral sampling, which in turn allows us to efficiently learn the model parameters, without the need of expensive iterative inference schemes (such as MCMC) per datapoint. The learned approximate posterior inference model can also be used for a host of tasks such as recognition, denoising, representation and visualization purposes. When a neural network is used for the recognition model, we arrive at the *variational auto-encoder*.

2 Method

The strategy in this section can be used to derive a lower bound estimator (a stochastic objective function) for a variety of directed graphical models with continuous latent variables. We will restrict ourselves here to the common case where we have an i.i.d. dataset with latent variables per datapoint, and where we like to perform maximum likelihood (ML) or maximum a posteriori (MAP) inference on the (global) parameters, and variational inference on the latent variables. It is, for example,

1

Frey Faces Dataset



MNIST Dataset

Source: Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes", 2014

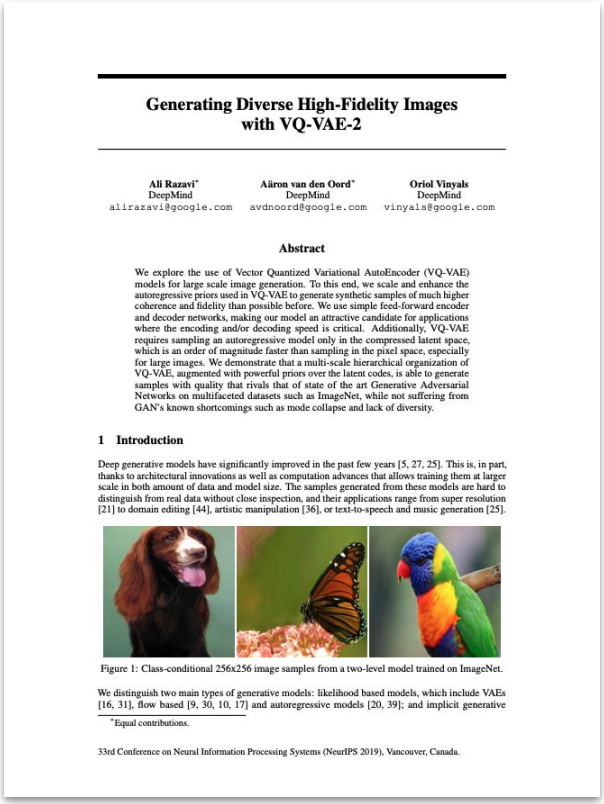
Variational Autoencoder (VAE) — Results

So does it work? → A few years later … (*NeurIPS 2019*)

Source: Razavi, Ali, Aaron Van den Oord, and Oriol Vinyals. "Generating diverse high-fidelity images with vq-vae-2", 2019.

Variational Autoencoder (VAE) — Results

So does it work? → A few years later … (*NeurIPS 2019*)



Generating Diverse High-Fidelity Images with VQ-VAE-2

Ali Razavi* DeepMind alirazavi@google.com
Aäron van den Oord* DeepMind avdnoord@google.com
Oriol Vinyals DeepMind vinyals@google.com

Abstract

We explore the use of Vector Quantized Variational AutoEncoder (VQ-VAE) models for large scale image generation. To this end, we scale and enhance the autoregressive priors used in VQ-VAE to generate synthetic samples of much higher coherence and fidelity than possible before. We use simple feed-forward encoder and decoder networks, making our model an attractive candidate for applications where the encoding and/or decoding speed is critical. Additionally, VQ-VAE requires sampling an autoregressive model only in the compressed latent space, which is an order of magnitude faster than sampling in the pixel space, especially for large images. We demonstrate that a multi-scale hierarchical organization of VQ-VAE, augmented with powerful priors over the latent codes, is able to generate samples with quality that rivals that of state of the art Generative Adversarial Networks on multifaceted datasets such as ImageNet, while not suffering from GAN's known shortcomings such as mode collapse and lack of diversity.

1 Introduction

Deep generative models have significantly improved in the past few years [5, 27, 25]. This is, in part, thanks to architectural innovations as well as computation advances that allows training them at larger scale in both amount of data and model size. The samples generated from these models are hard to distinguish from real data without close inspection, and their applications range from super resolution [21] to domain editing [44], artistic manipulation [36], or text-to-speech and music generation [25].

Figure 1: Class-conditional 256x256 image samples from a two-level model trained on ImageNet.

We distinguish two main types of generative models: likelihood based models, which include VAEs [16, 31], flow based [9, 30, 10, 17] and autoregressive models [20, 39]; and implicit generative

*Equal contributions.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

FFQH Dataset

Source: Razavi, Ali, Aaron Van den Oord, and Oriol Vinyals. "Generating diverse high-fidelity images with vq-vae-2", 2019.

Intro to Score-based Generative Models

VAEs $\rightarrow$ Score-based/Diffusion Generative Models

VAEs → Score-based/Diffusion Generative Models

VAEs and GANs were the two most-popular generative models for a few years, until around 2019*.

When a new paradigm of generative modeling came around…

**Actually, the first diffusion paper was in 2015 (or before), but it didn't really gain much attention until around 2019.*

Score-based/Diffusion Generative Models

Score-based generative models, and diffusion generative models:

Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

Eric A. Weiss EAWEISS@BERKELEY.EDU
University of California, Berkeley

Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

Abstract

A central problem in machine learning involves modeling complex data-sets using highly flexible families of probability distributions in which learning, sampling, inference, and evaluation are still analytically or computationally tractable. Here, we develop an approach that simultaneously achieves both flexibility and tractability. The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. This approach allows us to rapidly learn, sample from, and evaluate probabilities in deep generative models with thousands of layers or time steps, as well as to compute conditional and posterior probabilities under the learned model. We additionally release an open source reference implementation of the algorithm.

1. Introduction

Historically, probabilistic models suffer from a tradeoff between two conflicting objectives: *tractability* and *flexibility*. Models that are *tractable* can be analytically evaluated and easily fit to data (e.g. a Gaussian or Laplace). However, these models are unable to aptly describe structure in rich datasets. On the other hand, models that are *flexible* can be molded to fit structure in arbitrary data. For example, we can define models in terms of any (non-negative) function $\phi(\mathbf{x})$ yielding the flexible distribution $p(\mathbf{x}) = \frac{\phi(\mathbf{x})}{Z}$, where Z is a normalization constant. However, computing this normalization constant is generally intractable. Evaluating, training, or drawing samples from such flexible models typically requires a very expensive Monte Carlo process.

A variety of analytic approximations exist which ameliorate, but do not remove, this tradeoff–for instance mean field theory and its expansions (T, 1982; Tanaka, 1998), variational Bayes (Jordan et al., 1999), contrastive divergence (Welling & Hinton, 2002; Hinton, 2002), minimum probability flow (Sohl-Dickstein et al., 2011b;a), minimum KL contraction (Lyu, 2011), proper scoring rules (Gneiting & Raftery, 2007; Parry et al., 2012), score matching (Hyvärinen, 2005), pseudolikelihood (Besag, 1975), loopy belief propagation (Murphy et al., 1999), and many, many more. Non-parametric methods (Gershman & Blei, 2012) can also be very effective[1].

1.1. Diffusion probabilistic models

We present a novel way to define probabilistic models that allows:

1. extreme flexibility in model structure,
2. exact sampling,

[1]Non-parametric methods can be seen as transitioning smoothly between tractable and flexible models. For instance, a non-parametric Gaussian mixture model will represent a small amount of data using a single Gaussian, but may represent infinite data as a mixture of an infinite number of Gaussians.

Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

Generative Modeling by Estimating Gradients of the Data Distribution

Yang Song
Stanford University
yangsong@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Abstract

We introduce a new generative model where samples are produced via Langevin dynamics using gradients of the data distribution estimated with score matching. Because gradients can be ill-defined and hard to estimate when the data resides on low-dimensional manifolds, we perturb the data with different levels of Gaussian noise, and jointly estimate the corresponding scores, *i.e.*, the vector fields of gradients of the perturbed data distribution for all noise levels. For sampling, we propose an annealed Langevin dynamics where we use gradients corresponding to gradually decreasing noise levels as the sampling process gets closer to the data manifold. Our framework allows flexible model architectures, requires no sampling during training or the use of adversarial methods, and provides a learning objective that can be used for principled model comparisons. Our models produce samples comparable to GANs on MNIST, CelebA and CIFAR-10 datasets, achieving a new state-of-the-art inception score of 8.87 on CIFAR-10. Additionally, we demonstrate that our models learn effective representations via image inpainting experiments.

1 Introduction

Generative models have many applications in machine learning. To list a few, they have been used to generate high-fidelity images [26, 6], synthesize realistic speech and music fragments [58], improve the performance of semi-supervised learning [28, 10], detect adversarial examples and other anomalous data [54], imitation learning [22], and explore promising states in reinforcement learning [41]. Recent progress is mainly driven by two approaches: likelihood-based methods [17, 29, 11, 60] and generative adversarial networks (GAN [15]). The former uses log-likelihood (or a suitable surrogate) as the training objective, while the latter uses adversarial training to minimize f-divergences [40] or integral probability metrics [2, 55] between model and data distributions.

Although likelihood-based models and GANs have achieved great success, they have some intrinsic limitations. For example, likelihood-based models either have to use specialized architectures to build a normalized probability model (*e.g.*, autoregressive models, flow models), or use surrogate losses (*e.g.*, the evidence lower bound used in variational auto-encoders [29], contrastive divergence in energy-based models [21]) for training. GANs avoid some of the limitations of likelihood-based models, but their training can be unstable due to the adversarial training procedure. In addition, the GAN objective is not suitable for evaluating and comparing different GAN models. While other objectives exist for generative modeling, such as noise contrastive estimation [19] and minimum probability flow [50], these methods typically only work well for low-dimensional data.

In this paper, we explore a new principle for generative modeling based on estimating and sampling from the *(Stein) score* [33] of the logarithmic data density, which is the gradient of the log-density function at the input data point. This is a vector field pointing in the direction where the log data density grows the most. We use a neural network trained with score matching [24] to learn this vector field from data. We then produce samples using Langevin dynamics, which approximately

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Published as a conference paper at ICLR 2021

SCORE-BASED GENERATIVE MODELING THROUGH STOCHASTIC DIFFERENTIAL EQUATIONS

Yang Song*
Stanford University
yangsong@cs.stanford.edu

Jascha Sohl-Dickstein
Google Brain
jaschasd@google.com

Diederik P. Kingma
Google Brain
durk@google.com

Abhishek Kumar
Google Brain
abhishk@google.com

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Ben Poole
Google Brain
pooleb@google.com

ABSTRACT

Creating noise from data is easy; creating data from noise is generative modeling. We present a stochastic differential equation (SDE) that smoothly transforms a complex data distribution to a known prior distribution by slowly injecting noise, and a corresponding reverse-time SDE that transforms the prior distribution back into the data distribution by slowly removing the noise. Crucially, the reverse-time SDE depends only on the time-dependent gradient field (a.k.a., score) of the perturbed data distribution. By leveraging advances in score-based generative modeling, we can accurately estimate these scores with neural networks, and use numerical SDE solvers to generate samples. We show that this framework encapsulates previous approaches in score-based generative modeling and diffusion probabilistic modeling, allowing for new sampling procedures and new modeling capabilities. In particular, we introduce a predictor-corrector framework to correct errors in the evolution of the discretized reverse-time SDE. We also derive an equivalent neural ODE that samples from the same distribution as the SDE, but additionally enables exact likelihood computation, and improved sampling efficiency. In addition, we provide a new way to solve inverse problems with score-based models, as demonstrated with experiments on class-conditional generation, image inpainting, and colorization. Combined with multiple architectural improvements, we achieve record-breaking performance for unconditional image generation on CIFAR-10 with an Inception score of 9.89 and FID of 2.20, a competitive likelihood of 2.99 bits/dim, and demonstrate high fidelity generation of 1024 × 1024 images for the first time from a score-based generative model.

1 INTRODUCTION

Two successful classes of probabilistic generative models involve sequentially corrupting training data with slowly increasing noise, and then learning to reverse this corruption in order to form a generative model of the data. *Score matching with Langevin dynamics* (SMLD) (Song & Ermon, 2019) estimates the *score* (*i.e.*, the gradient of the log probability density with respect to data) at each noise scale, and then uses Langevin dynamics to sample from a sequence of decreasing noise scales during generation. *Denoising diffusion probabilistic modeling* (DDPM) (Sohl-Dickstein et al., 2015; Ho et al., 2020) trains a sequence of probabilistic models to reverse each step of the noise corruption, using knowledge of the functional form of the reverse distributions to make training tractable. For continuous state spaces, the DDPM training objective implicitly computes scores at each noise scale. We therefore refer to these two model classes together as *score-based generative models*.

Score-based generative models, and related techniques (Bordes et al., 2017; Goyal et al., 2017; Du & Mordatch, 2019), have proven effective at generation of images (Song & Ermon, 2019; 2020; Ho et al., 2020), audio (Chen et al., 2020; Kong et al., 2020), graphs (Niu et al., 2020), and shapes (Cai

*Work partially done during an internship at Google Brain.

Score-based/Diffusion Generative Models

Score-based generative models, and diffusion generative models:

Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

Eric A. Weiss EAWEISS@BERKELEY.EDU
University of California, Berkeley

Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

Abstract

A central problem in machine learning involves modeling complex data-sets using highly flexible families of probability distributions in which learning, sampling, inference, and evaluation are still analytically or computationally tractable. Here, we develop an approach that simultaneously achieves both flexibility and tractability. The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. This approach allows us to rapidly learn, sample from, and evaluate probabilities in deep generative models with thousands of layers or time steps, as well as to compute conditional and posterior probabilities under the learned model. We additionally release an open source reference implementation of the algorithm.

1. Introduction

Historically, probabilistic models suffer from a tradeoff between two conflicting objectives: *tractability* and *flexibility*. Models that are *tractable* can be analytically evaluated and easily fit to data (e.g. a Gaussian or Laplace). However, these models are unable to aptly describe structure in rich datasets. On the other hand, models that are *flexible* can be molded to fit structure in arbitrary data. For example, we can define models in terms of any (non-negative) function $\phi(\mathbf{x})$ yielding the flexible distribution $p(\mathbf{x}) = \frac{\phi(\mathbf{x})}{Z}$, where Z is a normalization constant. However, computing this normalization constant is generally intractable. Evaluating, training, or drawing samples from such flexible models typically requires a very expensive Monte Carlo process.

A variety of analytic approximations exist which ameliorate, but do not remove, this tradeoff–for instance mean field theory and its expansions (T, 1982; Tanaka, 1998), variational Bayes (Jordan et al., 1999), contrastive divergence (Welling & Hinton, 2002; Hinton, 2002), minimum probability flow (Sohl-Dickstein et al., 2011b;a), minimum KL contraction (Lyu, 2011), proper scoring rules (Gneiting & Raftery, 2007; Parry et al., 2012), score matching (Hyvärinen, 2005), pseudolikelihood (Besag, 1975), loopy belief propagation (Murphy et al., 1999), and many, many more. Non-parametric methods (Gershman & Blei, 2012) can also be very effective[1].

1.1. Diffusion probabilistic models

We present a novel way to define probabilistic models that allows:

1. extreme flexibility in model structure,
2. exact sampling,

[1]Non-parametric methods can be seen as transitioning smoothly between tractable and flexible models. For instance, a non-parametric Gaussian mixture model will represent a small amount of data using a single Gaussian, but may represent infinite data as a mixture of an infinite number of Gaussians.

Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

Generative Modeling by Estimating Gradients of the Data Distribution

Yang Song
Stanford University
yangsong@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Abstract

We introduce a new generative model where samples are produced via Langevin dynamics using gradients of the data distribution estimated with score matching. Because gradients can be ill-defined and hard to estimate when the data resides on low-dimensional manifolds, we perturb the data with different levels of Gaussian noise, and jointly estimate the corresponding scores, *i.e.*, the vector fields of gradients of the perturbed data distribution for all noise levels. For sampling, we propose an annealed Langevin dynamics where we use gradients corresponding to gradually decreasing noise levels as the sampling process gets closer to the data manifold. Our framework allows flexible model architectures, requires no sampling during training or the use of adversarial methods, and provides a learning objective that can be used for principled model comparisons. Our models produce samples comparable to GANs on MNIST, CelebA and CIFAR-10 datasets, achieving a new state-of-the-art inception score of 8.87 on CIFAR-10. Additionally, we demonstrate that our models learn effective representations via image inpainting experiments.

1 Introduction

Generative models have many applications in machine learning. To list a few, they have been used to generate high-fidelity images [26, 6], synthesize realistic speech and music fragments [58], improve the performance of semi-supervised learning [28, 10], detect adversarial examples and other anomalous data [54], imitation learning [22], and explore promising states in reinforcement learning [41]. Recent progress is mainly driven by two approaches: likelihood-based methods [17, 29, 11, 60] and generative adversarial networks (GAN [15]). The former uses log-likelihood (or a suitable surrogate) as the training objective, while the latter uses adversarial training to minimize f-divergences [40] or integral probability metrics [2, 55] between model and data distributions.

Although likelihood-based models and GANs have achieved great success, they have some intrinsic limitations. For example, likelihood-based models either have to use specialized architectures to build a normalized probability model (*e.g.*, autoregressive models, flow models), or use surrogate losses (*e.g.*, the evidence lower bound used in variational auto-encoders [29], contrastive divergence in energy-based models [21]) for training. GANs avoid some of the limitations of likelihood-based models, but their training can be unstable due to the adversarial training procedure. In addition, the GAN objective is not suitable for evaluating and comparing different GAN models. While other objectives exist for generative modeling, such as noise contrastive estimation [19] and minimum probability flow [50], these methods typically only work well for low-dimensional data.

In this paper, we explore a new principle for generative modeling based on estimating and sampling from the *(Stein) score* [33] of the logarithmic data density, which is the gradient of the log-density function at the input data point. This is a vector field pointing in the direction where the log data density grows the most. We use a neural network trained with score matching [24] to learn this vector field from data. We then produce samples using Langevin dynamics, which approximately

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

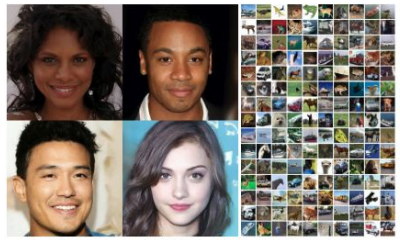


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Published as a conference paper at ICLR 2021

SCORE-BASED GENERATIVE MODELING THROUGH STOCHASTIC DIFFERENTIAL EQUATIONS

Yang Song*
Stanford University
yangsong@cs.stanford.edu

Jascha Sohl-Dickstein
Google Brain
jaschasd@google.com

Diederik P. Kingma
Google Brain
durk@google.com

Abhishek Kumar
Google Brain
abhishk@google.com

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Ben Poole
Google Brain
pooleb@google.com

ABSTRACT

Creating noise from data is easy; creating data from noise is generative modeling. We present a stochastic differential equation (SDE) that smoothly transforms a complex data distribution to a known prior distribution by slowly injecting noise, and a corresponding reverse-time SDE that transforms the prior distribution back into the data distribution by slowly removing the noise. Crucially, the reverse-time SDE depends only on the time-dependent gradient field (a.k.a., score) of the perturbed data distribution. By leveraging advances in score-based generative modeling, we can accurately estimate these scores with neural networks, and use numerical SDE solvers to generate samples. We show that this framework encapsulates previous approaches in score-based generative modeling and diffusion probabilistic modeling, allowing for new sampling procedures and new modeling capabilities. In particular, we introduce a predictor-corrector framework to correct errors in the evolution of the discretized reverse-time SDE. We also derive an equivalent neural ODE that samples from the same distribution as the SDE, but additionally enables exact likelihood computation, and improved sampling efficiency. In addition, we provide a new way to solve inverse problems with score-based models, as demonstrated with experiments on class-conditional generation, image inpainting, and colorization. Combined with multiple architectural improvements, we achieve record-breaking performance for unconditional image generation on CIFAR-10 with an Inception score of 9.89 and FID of 2.20, a competitive likelihood of 2.99 bits/dim, and demonstrate high fidelity generation of 1024 × 1024 images for the first time from a score-based generative model.

1 INTRODUCTION

Two successful classes of probabilistic generative models involve sequentially corrupting training data with slowly increasing noise, and then learning to reverse this corruption in order to form a generative model of the data. *Score matching with Langevin dynamics* (SMLD) (Song & Ermon, 2019) estimates the *score* (*i.e.*, the gradient of the log probability density with respect to data) at each noise scale, and then uses Langevin dynamics to sample from a sequence of decreasing noise scales during generation. *Denoising diffusion probabilistic modeling* (DDPM) (Sohl-Dickstein et al., 2015; Ho et al., 2020) trains a sequence of probabilistic models to reverse each step of the noise corruption, using knowledge of the functional form of the reverse distributions to make training tractable. For continuous state spaces, the DDPM training objective implicitly computes scores at each noise scale. We therefore refer to these two model classes together as *score-based generative models*.

Score-based generative models, and related techniques (Bordes et al., 2017; Goyal et al., 2017; Du & Mordatch, 2019), have proven effective at generation of images (Song & Ermon, 2019; 2020; Ho et al., 2020), audio (Chen et al., 2020; Kong et al., 2020), graphs (Niu et al., 2020), and shapes (Cai

*Work partially done during an internship at Google Brain.

First Diffusion Model (ICML 2015)

Score-based/Diffusion Generative Models

Score-based generative models, and diffusion generative models:

Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

Eric A. Weiss EAWEISS@BERKELEY.EDU
University of California, Berkeley

Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

Abstract

A central problem in machine learning involves modeling complex data-sets using highly flexible families of probability distributions in which learning, sampling, inference, and evaluation are still analytically or computationally tractable. Here, we develop an approach that simultaneously achieves both flexibility and tractability. The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. This approach allows us to rapidly learn, sample from, and evaluate probabilities in deep generative models with thousands of layers or time steps, as well as to compute conditional and posterior probabilities under the learned model. We additionally release an open source reference implementation of the algorithm.

1. Introduction

Historically, probabilistic models suffer from a tradeoff between two conflicting objectives: *tractability* and *flexibility*. Models that are *tractable* can be analytically evaluated and easily fit to data (e.g. a Gaussian or Laplace). However, these models are unable to aptly describe structure in rich datasets. On the other hand, models that are *flexible* can be molded to fit structure in arbitrary data. For example, we can define models in terms of any (non-negative) function $\phi(\mathbf{x})$ yielding the flexible distribution $p(\mathbf{x}) = \frac{\phi(\mathbf{x})}{Z}$, where Z is a normalization constant. However, computing this normalization constant is generally intractable. Evaluating, training, or drawing samples from such flexible models typically requires a very expensive Monte Carlo process.

A variety of analytic approximations exist which ameliorate, but do not remove, this tradeoff–for instance mean field theory and its expansions (T, 1982; Tanaka, 1998), variational Bayes (Jordan et al., 1999), contrastive divergence (Welling & Hinton, 2002; Hinton, 2002), minimum probability flow (Sohl-Dickstein et al., 2011b;a), minimum KL contraction (Lyu, 2011), proper scoring rules (Gneiting & Raftery, 2007; Parry et al., 2012), score matching (Hyvärinen, 2005), pseudolikelihood (Besag, 1975), loopy belief propagation (Murphy et al., 1999), and many, many more. Non-parametric methods (Gershman & Blei, 2012) can also be very effective[1].

1.1. Diffusion probabilistic models

We present a novel way to define probabilistic models that allows:

1. extreme flexibility in model structure,
2. exact sampling,

[1]Non-parametric methods can be seen as transitioning smoothly between tractable and flexible models. For instance, a non-parametric Gaussian mixture model will represent a small amount of data using a single Gaussian, but may represent infinite data as a mixture of an infinite number of Gaussians.

Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

Generative Modeling by Estimating Gradients of the Data Distribution

Yang Song
Stanford University
yangsong@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Abstract

We introduce a new generative model where samples are produced via Langevin dynamics using gradients of the data distribution estimated with score matching. Because gradients can be ill-defined and hard to estimate when the data resides on low-dimensional manifolds, we perturb the data with different levels of Gaussian noise, and jointly estimate the corresponding scores, *i.e.*, the vector fields of gradients of the perturbed data distribution for all noise levels. For sampling, we propose an annealed Langevin dynamics where we use gradients corresponding to gradually decreasing noise levels as the sampling process gets closer to the data manifold. Our framework allows flexible model architectures, requires no sampling during training or the use of adversarial methods, and provides a learning objective that can be used for principled model comparisons. Our models produce samples comparable to GANs on MNIST, CelebA and CIFAR-10 datasets, achieving a new state-of-the-art inception score of 8.87 on CIFAR-10. Additionally, we demonstrate that our models learn effective representations via image inpainting experiments.

1 Introduction

Generative models have many applications in machine learning. To list a few, they have been used to generate high-fidelity images [26, 6], synthesize realistic speech and music fragments [58], improve the performance of semi-supervised learning [28, 10], detect adversarial examples and other anomalous data [54], imitation learning [22], and explore promising states in reinforcement learning [41]. Recent progress is mainly driven by two approaches: likelihood-based methods [17, 29, 11, 60] and generative adversarial networks (GAN [15]). The former uses log-likelihood (or a suitable surrogate) as the training objective, while the latter uses adversarial training to minimize f-divergences [40] or integral probability metrics [2, 55] between model and data distributions.

Although likelihood-based models and GANs have achieved great success, they have some intrinsic limitations. For example, likelihood-based models either have to use specialized architectures to build a normalized probability model (*e.g.*, autoregressive models, flow models), or use surrogate losses (*e.g.*, the evidence lower bound used in variational auto-encoders [29], contrastive divergence in energy-based models [21]) for training. GANs avoid some of the limitations of likelihood-based models, but their training can be unstable due to the adversarial training procedure. In addition, the GAN objective is not suitable for evaluating and comparing different GAN models. While other objectives exist for generative modeling, such as noise contrastive estimation [19] and minimum probability flow [50], these methods typically only work well for low-dimensional data.

In this paper, we explore a new principle for generative modeling based on estimating and sampling from the *(Stein) score* [33] of the logarithmic data density, which is the gradient of the log-density function at the input data point. This is a vector field pointing in the direction where the log data density grows the most. We use a neural network trained with score matching [24] to learn this vector field from data. We then produce samples using Langevin dynamics, which approximately

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

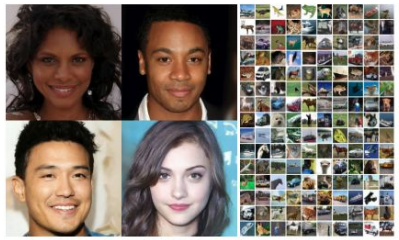


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Published as a conference paper at ICLR 2021

SCORE-BASED GENERATIVE MODELING THROUGH STOCHASTIC DIFFERENTIAL EQUATIONS

Yang Song*
Stanford University
yangsong@cs.stanford.edu

Jascha Sohl-Dickstein
Google Brain
jaschasd@google.com

Diederik P. Kingma
Google Brain
durk@google.com

Abhishek Kumar
Google Brain
abhishk@google.com

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Ben Poole
Google Brain
pooleb@google.com

ABSTRACT

Creating noise from data is easy; creating data from noise is generative modeling. We present a stochastic differential equation (SDE) that smoothly transforms a complex data distribution to a known prior distribution by slowly injecting noise, and a corresponding reverse-time SDE that transforms the prior distribution back into the data distribution by slowly removing the noise. Crucially, the reverse-time SDE depends only on the time-dependent gradient field (a.k.a., score) of the perturbed data distribution. By leveraging advances in score-based generative modeling, we can accurately estimate these scores with neural networks, and use numerical SDE solvers to generate samples. We show that this framework encapsulates previous approaches in score-based generative modeling and diffusion probabilistic modeling, allowing for new sampling procedures and new modeling capabilities. In particular, we introduce a predictor-corrector framework to correct errors in the evolution of the discretized reverse-time SDE. We also derive an equivalent neural ODE that samples from the same distribution as the SDE, but additionally enables exact likelihood computation, and improved sampling efficiency. In addition, we provide a new way to solve inverse problems with score-based models, as demonstrated with experiments on class-conditional generation, image inpainting, and colorization. Combined with multiple architectural improvements, we achieve record-breaking performance for unconditional image generation on CIFAR-10 with an Inception score of 9.89 and FID of 2.20, a competitive likelihood of 2.99 bits/dim, and demonstrate high fidelity generation of 1024 × 1024 images for the first time from a score-based generative model.

1 INTRODUCTION

Two successful classes of probabilistic generative models involve sequentially corrupting training data with slowly increasing noise, and then learning to reverse this corruption in order to form a generative model of the data. *Score matching with Langevin dynamics* (SMLD) (Song & Ermon, 2019) estimates the *score* (*i.e.*, the gradient of the log probability density with respect to data) at each noise scale, and then uses Langevin dynamics to sample from a sequence of decreasing noise scales during generation. *Denoising diffusion probabilistic modeling* (DDPM) (Sohl-Dickstein et al., 2015; Ho et al., 2020) trains a sequence of probabilistic models to reverse each step of the noise corruption, using knowledge of the functional form of the reverse distributions to make training tractable. For continuous state spaces, the DDPM training objective implicitly computes scores at each noise scale. We therefore refer to these two model classes together as *score-based generative models*.

Score-based generative models, and related techniques (Bordes et al., 2017; Goyal et al., 2017; Du & Mordatch, 2019), have proven effective at generation of images (Song & Ermon, 2019; 2020; Ho et al., 2020), audio (Chen et al., 2020; Kong et al., 2020), graphs (Niu et al., 2020), and shapes (Cai

*Work partially done during an internship at Google Brain.

Score-based Generative Model (NeurIPS 2019)

Score-based/Diffusion Generative Models

Score-based generative models, and diffusion generative models:

Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

Eric A. Weiss EAWEISS@BERKELEY.EDU
University of California, Berkeley

Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

Abstract

A central problem in machine learning involves modeling complex data-sets using highly flexible families of probability distributions in which learning, sampling, inference, and evaluation are still analytically or computationally tractable. Here, we develop an approach that simultaneously achieves both flexibility and tractability. The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. This approach allows us to rapidly learn, sample from, and evaluate probabilities in deep generative models with thousands of layers or time steps, as well as to compute conditional and posterior probabilities under the learned model. We additionally release an open source reference implementation of the algorithm.

1. Introduction

Historically, probabilistic models suffer from a tradeoff between two conflicting objectives: *tractability* and *flexibility*. Models that are *tractable* can be analytically evaluated and easily fit to data (e.g. a Gaussian or Laplace). However, these models are unable to aptly describe structure in rich datasets. On the other hand, models that are *flexible* can be molded to fit structure in arbitrary data. For example, we can define models in terms of any (non-negative) function $\phi(\mathbf{x})$ yielding the flexible distribution $p(\mathbf{x}) = \frac{\phi(\mathbf{x})}{Z}$, where Z is a normalization constant. However, computing this normalization constant is generally intractable. Evaluating, training, or drawing samples from such flexible models typically requires a very expensive Monte Carlo process.

A variety of analytic approximations exist which ameliorate, but do not remove, this tradeoff–for instance mean field theory and its expansions (T, 1982; Tanaka, 1998), variational Bayes (Jordan et al., 1999), contrastive divergence (Welling & Hinton, 2002; Hinton, 2002), minimum probability flow (Sohl-Dickstein et al., 2011b;a), minimum KL contraction (Lyu, 2011), proper scoring rules (Gneiting & Raftery, 2007; Parry et al., 2012), score matching (Hyvärinen, 2005), pseudolikelihood (Besag, 1975), loopy belief propagation (Murphy et al., 1999), and many, many more. Non-parametric methods (Gershman & Blei, 2012) can also be very effective[1].

1.1. Diffusion probabilistic models

We present a novel way to define probabilistic models that allows:

1. extreme flexibility in model structure,
2. exact sampling,

Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s).

[1]Non-parametric methods can be seen as transitioning smoothly between tractable and flexible models. For instance, a non-parametric Gaussian mixture model will represent a small amount of data using a single Gaussian, but may represent infinite data as a mixture of an infinite number of Gaussians.

Generative Modeling by Estimating Gradients of the Data Distribution

Yang Song Stanford University yangsong@cs.stanford.edu
Stefano Ermon Stanford University ermon@cs.stanford.edu

Abstract

We introduce a new generative model where samples are produced via Langevin dynamics using gradients of the data distribution estimated with score matching. Because gradients can be ill-defined and hard to estimate when the data resides on low-dimensional manifolds, we perturb the data with different levels of Gaussian noise, and jointly estimate the corresponding scores, *i.e.*, the vector fields of gradients of the perturbed data distribution for all noise levels. For sampling, we propose an annealed Langevin dynamics where we use gradients corresponding to gradually decreasing noise levels as the sampling process gets closer to the data manifold. Our framework allows flexible model architectures, requires no sampling during training or the use of adversarial methods, and provides a learning objective that can be used for principled model comparisons. Our models produce samples comparable to GANs on MNIST, CelebA and CIFAR-10 datasets, achieving a new state-of-the-art inception score of 8.87 on CIFAR-10. Additionally, we demonstrate that our models learn effective representations via image inpainting experiments.

1 Introduction

Generative models have many applications in machine learning. To list a few, they have been used to generate high-fidelity images [26, 6], synthesize realistic speech and music fragments [58], improve the performance of semi-supervised learning [28, 10], detect adversarial examples and other anomalous data [54], imitation learning [22], and explore promising states in reinforcement learning [41]. Recent progress is mainly driven by two approaches: likelihood-based methods [17, 29, 11, 60] and generative adversarial networks (GAN [15]). The former uses log-likelihood (or a suitable surrogate) as the training objective, while the latter uses adversarial training to minimize f-divergences [40] or integral probability metrics [2, 55] between model and data distributions.

Although likelihood-based models and GANs have achieved great success, they have some intrinsic limitations. For example, likelihood-based models either have to use specialized architectures to build a normalized probability model (*e.g.*, autoregressive models, flow models), or use surrogate losses (*e.g.*, the evidence lower bound used in variational auto-encoders [29], contrastive divergence in energy-based models [21]) for training. GANs avoid some of the limitations of likelihood-based models, but their training can be unstable due to the adversarial training procedure. In addition, the GAN objective is not suitable for evaluating and comparing different GAN models. While other objectives exist for generative modeling, such as noise contrastive estimation [19] and minimum probability flow [50], these methods typically only work well for low-dimensional data.

In this paper, we explore a new principle for generative modeling based on estimating and sampling from the *(Stein) score* [33] of the logarithmic data density, which is the gradient of the log-density function at the input data point. This is a vector field pointing in the direction where the log data density grows the most. We use a neural network trained with score matching [24] to learn this vector field from data. We then produce samples using Langevin dynamics, which approximately

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Denoising Diffusion Probabilistic Models

Jonathan Ho UC Berkeley jonathanho@berkeley.edu
Ajay Jain UC Berkeley ajayj@berkeley.edu
Pieter Abbeel UC Berkeley pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

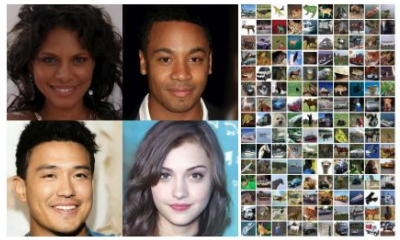


Figure 1: Generated samples on CelebA-HQ 256×256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Published as a conference paper at ICLR 2021

SCORE-BASED GENERATIVE MODELING THROUGH STOCHASTIC DIFFERENTIAL EQUATIONS

Yang Song* Stanford University yangsong@cs.stanford.edu
Jascha Sohl-Dickstein Google Brain jaschasd@google.com
Diederik P. Kingma Google Brain durk@google.com
Abhishek Kumar Google Brain abhishk@google.com
Stefano Ermon Stanford University ermon@cs.stanford.edu
Ben Poole Google Brain pooleb@google.com

ABSTRACT

Creating noise from data is easy; creating data from noise is generative modeling. We present a stochastic differential equation (SDE) that smoothly transforms a complex data distribution to a known prior distribution by slowly injecting noise, and a corresponding reverse-time SDE that transforms the prior distribution back into the data distribution by slowly removing the noise. Crucially, the reverse-time SDE depends only on the time-dependent gradient field (a.k.a., score) of the perturbed data distribution. By leveraging advances in score-based generative modeling, we can accurately estimate these scores with neural networks, and use numerical SDE solvers to generate samples. We show that this framework encapsulates previous approaches in score-based generative modeling and diffusion probabilistic modeling, allowing for new sampling procedures and new modeling capabilities. In particular, we introduce a predictor-corrector framework to correct errors in the evolution of the discretized reverse-time SDE. We also derive an equivalent neural ODE that samples from the same distribution as the SDE, but additionally enables exact likelihood computation, and improved sampling efficiency. In addition, we provide a new way to solve inverse problems with score-based models, as demonstrated with experiments on class-conditional generation, image inpainting, and colorization. Combined with multiple architectural improvements, we achieve record-breaking performance for unconditional image generation on CIFAR-10 with an Inception score of 9.89 and FID of 2.20, a competitive likelihood of 2.99 bits/dim, and demonstrate high fidelity generation of 1024×1024 images for the first time from a score-based generative model.

1 INTRODUCTION

Two successful classes of probabilistic generative models involve sequentially corrupting training data with slowly increasing noise, and then learning to reverse this corruption in order to form a generative model of the data. *Score matching with Langevin dynamics* (SMLD) (Song & Ermon, 2019) estimates the *score* (*i.e.*, the gradient of the log probability density with respect to data) at each noise scale, and then uses Langevin dynamics to sample from a sequence of decreasing noise scales during generation. *Denoising diffusion probabilistic modeling* (DDPM) (Sohl-Dickstein et al., 2015; Ho et al., 2020) trains a sequence of probabilistic models to reverse each step of the noise corruption, using knowledge of the functional form of the reverse distributions to make training tractable. For continuous state spaces, the DDPM training objective implicitly computes scores at each noise scale. We therefore refer to these two model classes together as *score-based generative models*.

Score-based generative models, and related techniques (Bordes et al., 2017; Goyal et al., 2017; Du & Mordatch, 2019), have proven effective at generation of images (Song & Ermon, 2019; 2020; Ho et al., 2020), audio (Chen et al., 2020; Kong et al., 2020), graphs (Niu et al., 2020), and shapes (Cai

*Work partially done during an internship at Google Brain.

Denoising Diffusion Model (NeurIPS 2020)

Score-based/Diffusion Generative Models

Score-based generative models, and diffusion generative models:

Deep Unsupervised Learning using Nonequilibrium Thermodynamics

Jascha Sohl-Dickstein JASCHA@STANFORD.EDU
Stanford University

Eric A. Weiss EAWEISS@BERKELEY.EDU
University of California, Berkeley

Niru Maheswaranathan NIRUM@STANFORD.EDU
Stanford University

Surya Ganguli SGANGULI@STANFORD.EDU
Stanford University

Abstract

A central problem in machine learning involves modeling complex data-sets using highly flexible families of probability distributions in which learning, sampling, inference, and evaluation are still analytically or computationally tractable. Here, we develop an approach that simultaneously achieves both flexibility and tractability. The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. This approach allows us to rapidly learn, sample from, and evaluate probabilities in deep generative models with thousands of layers or time steps, as well as to compute conditional and posterior probabilities under the learned model. We additionally release an open source reference implementation of the algorithm.

1. Introduction

Historically, probabilistic models suffer from a tradeoff between two conflicting objectives: *tractability* and *flexibility*. Models that are *tractable* can be analytically evaluated and easily fit to data (e.g. a Gaussian or Laplace). However, these models are unable to aptly describe structure in rich datasets. On the other hand, models that are *flexible* can be molded to fit structure in arbitrary data. For example, we can define models in terms of any (non-negative) function $\phi(\mathbf{x})$ yielding the flexible distribution $p(\mathbf{x}) = \frac{\phi(\mathbf{x})}{Z}$, where Z is a normalization constant. However, computing this normalization constant is generally intractable. Evaluating, training, or drawing samples from such flexible models typically requires a very expensive Monte Carlo process.

A variety of analytic approximations exist which ameliorate, but do not remove, this tradeoff–for instance mean field theory and its expansions (T, 1982; Tanaka, 1998), variational Bayes (Jordan et al., 1999), contrastive divergence (Welling & Hinton, 2002; Hinton, 2002), minimum probability flow (Sohl-Dickstein et al., 2011b;a), minimum KL contraction (Lyu, 2011), proper scoring rules (Gneiting & Raftery, 2007; Parry et al., 2012), score matching (Hyvärinen, 2005), pseudolikelihood (Besag, 1975), loopy belief propagation (Murphy et al., 1999), and many, many more. Non-parametric methods (Gershman & Blei, 2012) can also be very effective[1].

1.1. Diffusion probabilistic models

We present a novel way to define probabilistic models that allows:

1. extreme flexibility in model structure,
2. exact sampling,

[1]Non-parametric methods can be seen as transitioning smoothly between tractable and flexible models. For instance, a non-parametric Gaussian mixture model will represent a small amount of data using a single Gaussian, but may represent infinite data as a mixture of an infinite number of Gaussians.

Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

Generative Modeling by Estimating Gradients of the Data Distribution

Yang Song Stanford University yangsong@cs.stanford.edu
Stefano Ermon Stanford University ermon@cs.stanford.edu

Abstract

We introduce a new generative model where samples are produced via Langevin dynamics using gradients of the data distribution estimated with score matching. Because gradients can be ill-defined and hard to estimate when the data resides on low-dimensional manifolds, we perturb the data with different levels of Gaussian noise, and jointly estimate the corresponding scores, *i.e.*, the vector fields of gradients of the perturbed data distribution for all noise levels. For sampling, we propose an annealed Langevin dynamics where we use gradients corresponding to gradually decreasing noise levels as the sampling process gets closer to the data manifold. Our framework allows flexible model architectures, requires no sampling during training or the use of adversarial methods, and provides a learning objective that can be used for principled model comparisons. Our models produce samples comparable to GANs on MNIST, CelebA and CIFAR-10 datasets, achieving a new state-of-the-art inception score of 8.87 on CIFAR-10. Additionally, we demonstrate that our models learn effective representations via image inpainting experiments.

1 Introduction

Generative models have many applications in machine learning. To list a few, they have been used to generate high-fidelity images [26, 6], synthesize realistic speech and music fragments [58], improve the performance of semi-supervised learning [28, 10], detect adversarial examples and other anomalous data [54], imitation learning [22], and explore promising states in reinforcement learning [41]. Recent progress is mainly driven by two approaches: likelihood-based methods [17, 29, 11, 60] and generative adversarial networks (GAN [15]). The former uses log-likelihood (or a suitable surrogate) as the training objective, while the latter uses adversarial training to minimize f-divergences [40] or integral probability metrics [2, 55] between model and data distributions.

Although likelihood-based models and GANs have achieved great success, they have some intrinsic limitations. For example, likelihood-based models either have to use specialized architectures to build a normalized probability model (*e.g.*, autoregressive models, flow models), or use surrogate losses (*e.g.*, the evidence lower bound used in variational auto-encoders [29], contrastive divergence in energy-based models [21]) for training. GANs avoid some of the limitations of likelihood-based models, but their training can be unstable due to the adversarial training procedure. In addition, the GAN objective is not suitable for evaluating and comparing different GAN models. While other objectives exist for generative modeling, such as noise contrastive estimation [19] and minimum probability flow [50], these methods typically only work well for low-dimensional data.

In this paper, we explore a new principle for generative modeling based on estimating and sampling from the *(Stein) score* [33] of the logarithmic data density, which is the gradient of the log-density function at the input data point. This is a vector field pointing in the direction where the log data density grows the most. We use a neural network trained with score matching [24] to learn this vector field from data. We then produce samples using Langevin dynamics, which approximately

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Denoising Diffusion Probabilistic Models

Jonathan Ho UC Berkeley jonathanho@berkeley.edu
Ajay Jain UC Berkeley ajayj@berkeley.edu
Pieter Abbeel UC Berkeley pabbeel@cs.berkeley.edu

Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].

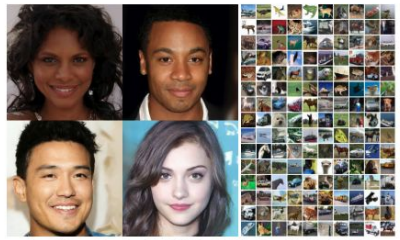


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Published as a conference paper at ICLR 2021

SCORE-BASED GENERATIVE MODELING THROUGH STOCHASTIC DIFFERENTIAL EQUATIONS

Yang Song* Stanford University yangsong@cs.stanford.edu
Jascha Sohl-Dickstein Google Brain jaschasd@google.com
Diederik P. Kingma Google Brain durk@google.com
Abhishek Kumar Google Brain abhishk@google.com
Stefano Ermon Stanford University ermon@cs.stanford.edu
Ben Poole Google Brain pooleb@google.com

ABSTRACT

Creating noise from data is easy; creating data from noise is generative modeling. We present a stochastic differential equation (SDE) that smoothly transforms a complex data distribution to a known prior distribution by slowly injecting noise, and a corresponding reverse-time SDE that transforms the prior distribution back into the data distribution by slowly removing the noise. Crucially, the reverse-time SDE depends only on the time-dependent gradient field (a.k.a., score) of the perturbed data distribution. By leveraging advances in score-based generative modeling, we can accurately estimate these scores with neural networks, and use numerical SDE solvers to generate samples. We show that this framework encapsulates previous approaches in score-based generative modeling and diffusion probabilistic modeling, allowing for new sampling procedures and new modeling capabilities. In particular, we introduce a predictor-corrector framework to correct errors in the evolution of the discretized reverse-time SDE. We also derive an equivalent neural ODE that samples from the same distribution as the SDE, but additionally enables exact likelihood computation, and improved sampling efficiency. In addition, we provide a new way to solve inverse problems with score-based models, as demonstrated with experiments on class-conditional generation, image inpainting, and colorization. Combined with multiple architectural improvements, we achieve record-breaking performance for unconditional image generation on CIFAR-10 with an Inception score of 9.89 and FID of 2.20, a competitive likelihood of 2.99 bits/dim, and demonstrate high fidelity generation of 1024 × 1024 images for the first time from a score-based generative model.

1 INTRODUCTION

Two successful classes of probabilistic generative models involve sequentially corrupting training data with slowly increasing noise, and then learning to reverse this corruption in order to form a generative model of the data. *Score matching with Langevin dynamics* (SMLD) (Song & Ermon, 2019) estimates the *score* (*i.e.*, the gradient of the log probability density with respect to data) at each noise scale, and then uses Langevin dynamics to sample from a sequence of decreasing noise scales during generation. *Denoising diffusion probabilistic modeling* (DDPM) (Sohl-Dickstein et al., 2015; Ho et al., 2020) trains a sequence of probabilistic models to reverse each step of the noise corruption, using knowledge of the functional form of the reverse distributions to make training tractable. For continuous state spaces, the DDPM training objective implicitly computes scores at each noise scale. We therefore refer to these two model classes together as *score-based generative models*.

Score-based generative models, and related techniques (Bordes et al., 2017; Goyal et al., 2017; Du & Mordatch, 2019), have proven effective at generation of images (Song & Ermon, 2019; 2020; Ho et al., 2020), audio (Chen et al., 2020; Kong et al., 2020), graphs (Niu et al., 2020), and shapes (Cai

*Work partially done during an internship at Google Brain.

Score-based Models via SDEs (ICLR 2021)

Score-based/Diffusion Generative Models

Score-based and diffusion generative models take advantage of the approximate inference algorithms we have recently learned!

Score-based/Diffusion Generative Models

Score-based and diffusion generative models take advantage of the approximate inference algorithms we have recently learned!

Score-based generative models → [today]

- Involve Langevin Monte Carlo (LMC) with an annealed proposal distribution.

Score-based/Diffusion Generative Models

Score-based and diffusion generative models take advantage of the approximate inference algorithms we have recently learned!

Score-based generative models → [today]

- Involve Langevin Monte Carlo (LMC) with an annealed proposal distribution.

Denoising diffusion generative models → [upcoming]

- Involve variational inference (VI) via ELBO maximization.

Score-based Generative Models — Setup

Score-based Generative Models — Setup

Suppose that we have a dataset: $D = \{x_1, x_2, \ldots, x_n\}$

Where each data point is assumed to be drawn from some unknown distribution, *i.e.*,

$$x_i \sim p(x)$$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Setup

Suppose that we have a dataset: $D = \{x_1, x_2, \ldots, x_n\}$

Where each data point is assumed to be drawn from some unknown distribution, *i.e.*,

$$x_i \sim p(x)$$

Suppose we want to model this distribution using a model $p_\theta(x)$, where we assume this distribution is parameterized by a $\theta \in \mathbb{R}^d$.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Setup

We can write this model $p_\theta(x)$ in a general way as:

Score-based Generative Models — Setup

We can write this model $p_\theta(x)$ in a general way as:

$$p_\theta(x) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x) = \frac{1}{Z(\theta)} e^{-f_\theta(x)}$$

Where, as before,

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Setup

We can write this model $p_\theta(x)$ in a general way as:

$$p_\theta(x) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x) = \frac{1}{Z(\theta)} e^{-f_\theta(x)}$$

Where, as before, $Z(\theta)$ is the normalization constant.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Setup

We can write this model $p_\theta(x)$ in a general way as:

$$p_\theta(x) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x) = \frac{1}{Z(\theta)} e^{-f_\theta(x)}$$

Where, as before, $Z(\theta)$ is the normalization constant.

And $\tilde{p}_\theta(x)$ is an unnormalized PDF.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Setup

We can write this model $p_\theta(x)$ in a general way as:

$$p_\theta(x) = \frac{1}{Z(\theta)} \tilde{p}_\theta(x) = \frac{1}{Z(\theta)} e^{-f_\theta(x)}$$

Where, as before, $Z(\theta)$ is the normalization constant.

And $\tilde{p}_\theta(x)$ is an unnormalized PDF.

And $f_\theta(x)$ is equal to the negative log of the unnormalized PDF, *i.e.*,

$$f_\theta(x) = -\log \tilde{p}_\theta(x)$$

Sometimes called an *energy-based model*.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Difficulties with MLE

Previous strategies involve trying to learning parameters θ via maximum likelihood:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} \log p_\theta(x_i)$$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Difficulties with MLE

Previous strategies involve trying to learning parameters θ via maximum likelihood:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} \log p_\theta(x_i)$$

However:

- To optimize (or evaluate) $p_\theta(x)$, we need the normalization constant $Z(\theta)$.
- …which is intractable in general for most $\tilde{p}_\theta(x)$.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — Main Strategy

So instead, we will take the following strategy:

Score-based Generative Models — Main Strategy

So instead, we will take the following strategy:

(1) **First**, we will aim to learn what is referred to as the **(Stein) score function** of the log data density.

(2) **Second**, Given this score function, we will be able to draw samples from $p_\theta(x)$ using Langevin Monte Carlo.

Score-based Generative Models — Main Strategy

So instead, we will take the following strategy:

(1) **First**, we will aim to learn what is referred to as the **(Stein) score function** of the log data density.

(2) **Second**, Given this score function, we will be able to draw samples from $p_\theta(x)$ using Langevin Monte Carlo.

Going through these two steps…

Score-based Generative Models

First, we will aim to learn the **(Stein) score function** of the log data density, *i.e.*,

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

Score-based Generative Models

First, we will aim to learn the **(Stein) score function** of the log data density, *i.e.*,

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

Note that:

- The term *score* is also sometimes used to refer to the gradient of the log-likelihood function with respect to the parameter θ.
- We've seen this term before! (Recall in LMC/HMC…).

Score-based Generative Models

First, we will aim to learn the **(Stein) score function** of the log data density, *i.e.*,

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

This score function can also be written:

$$s_\theta(x) = \nabla_x \log p_\theta(x) = \nabla_x \log \left(\frac{1}{Z(\theta)} \tilde{p}_\theta(x) \right)$$
$$= -\nabla_x f_\theta(x) - \underbrace{\nabla_x \log Z(\theta)}_{=0} = -\nabla_x f_\theta(x)$$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models

First, we will aim to learn the **(Stein) score function** of the log data density, *i.e.*,

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

This score function can also be written:

$$s_\theta(x) = \nabla_x \log p_\theta(x) = \nabla_x \log \left(\frac{1}{Z(\theta)} \tilde{p}_\theta(x) \right)$$

Definition of unnormalized PDF

$$= -\nabla_x f_\theta(x) - \underbrace{\nabla_x \log Z(\theta)}_{=0} = -\nabla_x f_\theta(x)$$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models

First, we will aim to learn the **(Stein) score function** of the log data density, *i.e.*,

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

This score function can also be written:

$$s_\theta(x) = \nabla_x \log p_\theta(x) = \nabla_x \log \left(\frac{1}{Z(\theta)} \tilde{p}_\theta(x) \right)$$

Definition of unnormalized PDF

$$= -\nabla_x f_\theta(x) - \underbrace{\nabla_x \log Z(\theta)}_{=0} = -\nabla_x f_\theta(x)$$

Definition of energy-based model

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models

First, we will aim to learn the **(Stein) score function** of the log data density, *i.e.*,

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

This score function can also be written:

$$s_\theta(x) = \nabla_x \log p_\theta(x) = \nabla_x \log \left(\frac{1}{Z(\theta)} \tilde{p}_\theta(x) \right)$$

Definition of unnormalized PDF

$$= -\nabla_x f_\theta(x) - \underbrace{\nabla_x \log Z(\theta)}_{=0} = -\nabla_x f_\theta(x)$$

Definition of energy-based model

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models

Suppose we have learned an approximation $s_\theta(x)$ to the true score function:

$$s_\theta(x) \approx \nabla_x \log p(x) = \nabla_x \log \tilde{p}(x)$$

I.e., it approximates the score function of the distribution that generated our data

Score-based Generative Models

Suppose we have learned an approximation $s_\theta(x)$ to the true score function:

$$s_\theta(x) \approx \nabla_x \log p(x) = \nabla_x \log \tilde{p}(x)$$

I.e., it approximates the score function of the distribution that generated our data

Recall that in Langevin Monte Carlo (SMC):

- In order to sample from a PDF, you just need access to the gradient of the unnormalized PDF!

Score-based Generative Models

Quick LMC Recap:

Gradient Based MCMC

Suppose we have a given PDF $p(x), \quad x \in \mathbb{R}^n$.

Consider an objective function: $f(x) = -\log p(x)\,.$

The gradient of this objective function is:

$$\nabla_x f(x) = \left[\frac{d}{dx_1} f(x), \ldots, \frac{d}{dx_n} f(x)\right]^\top$$

Score-based Generative Models

Quick LMC Recap:

Gradient Based MCMC — Langevin Monte Carlo (LMC)

Langevin Monte Carlo (LMC) is very similar!

Each step in LMC can be written:

$$x_t = x_{t-1} - \frac{\epsilon_t}{2}\nabla_x f(x_{t-1}) + \sqrt{\epsilon_t}\mathcal{N}(0, I_n)$$

Main difference is an extra "stochastic term"

$$x_t = x_{t-1} - \eta_t \nabla_x f(x_{t-1})$$

Can view this as "*gradient descent plus noise*"

Compare with gradient descent

Score-based Generative Models

Therefore (after learning the score function):

Score-based Generative Models

Therefore (after learning the score function):

Second, we can simply run LMC with our learned score function $s_\theta(x)$:

$$x_{i+1} \leftarrow x_i + \epsilon \nabla_x s_\theta(x) + \sqrt{2\epsilon}\,\mathcal{N}(0, I), \quad i = 0, 1, \cdots, K,$$

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

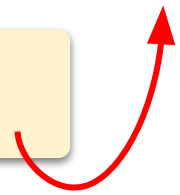
Score-based Generative Models

Therefore (after learning the score function):

Second, we can simply run LMC with our learned score function $s_\theta(x)$:

$$x_{i+1} \leftarrow x_i + \epsilon \nabla_x s_\theta(x) + \sqrt{2\epsilon}\,\mathcal{N}(0, I), \quad i = 0, 1, \cdots, K,$$

Note the learned score function is swapped in.



Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models

Therefore (after learning the score function):

Second, we can simply run LMC with our learned score function $s_\theta(x)$:

$$x_{i+1} \leftarrow x_i + \epsilon \nabla_x s_\theta(x) + \sqrt{2\epsilon}\,\mathcal{N}(0, I), \quad i = 0, 1, \cdots, K,$$

Note the learned score function is swapped in.

This is equivalent to previous LMC, but step size is written slightly different here.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — An illustration

An illustration:

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models — An illustration

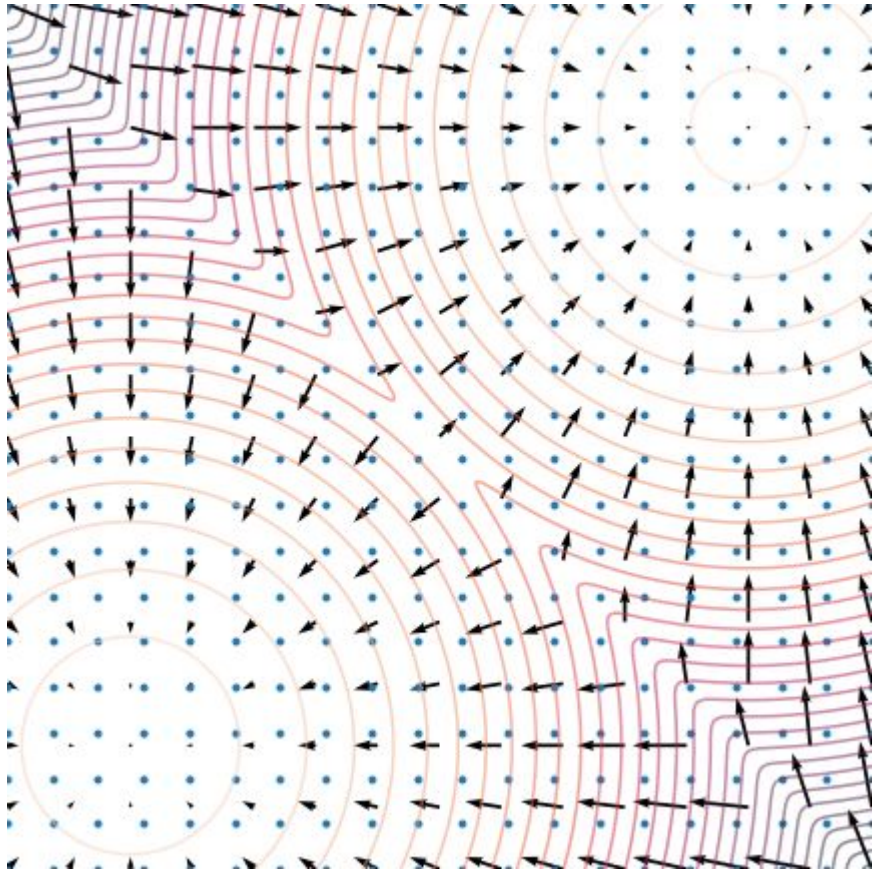
An illustration:

Contour lines denote the PDF $p_\theta(x)$.

Black arrows denote the score function $s_\theta(x)$.

Blue dots denote the samples produced via LMC.

(After random initialization).



Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

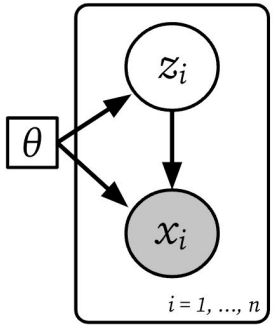
Score-based Generative Models — Intuition on Learning

What are we learning here (*e.g.*, with a neural network) versus in VAEs?

Score-based Generative Models — Intuition on Learning

What are we learning here (*e.g.*, with a neural network) versus in VAEs?

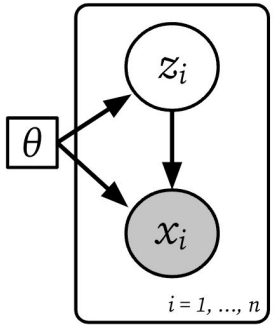
- In VAEs, we aim to learn a **direct map** from noise to data.
  - *I.e.*, directly learn the PDF of interest.
  - ⇒ a neural network that generates a sample from this PDF.



Score-based Generative Models — Intuition on Learning

What are we learning here (*e.g.*, with a neural network) versus in VAEs?

- In VAEs, we aim to learn a **direct map** from noise to data.
  - *I.e.*, directly learn the PDF of interest.
  - ⇒ a neural network that generates a sample from this PDF.



- In score-based models, we aim to learn a **score function**
  - Which we can view as pointing us in the direction of steepest ascent of PDF.
    ⇒ Use neural network *repeatedly* to take gradient steps to sample from this PDF.

Score-based Generative Models

So the main question is: How do we learn an approximation $s_\phi(x)$ to the score function $s_\theta(x)$?

Score-based Generative Models

So the main question is: How do we learn an approximation $s_\phi(x)$ to the score function $s_\theta(x)$?

⇒ Answer: we will minimize the (so called) *Fisher divergence* between the model and data distributions, written as:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - s_\theta(x)\|_2^2\right]$$

Intuitively: this compares the squared L2 distance between the ground-truth data score and the score-based model.

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models

So the main question is: How do we learn an approximation $s_\phi(x)$ to the score function $s_\theta(x)$?

⇒ Answer: we will minimize the (so called) *Fisher divergence* between the model and data distributions, written as:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\|\nabla_x \log p(x) - s_\theta(x)\|_2^2\right]$$

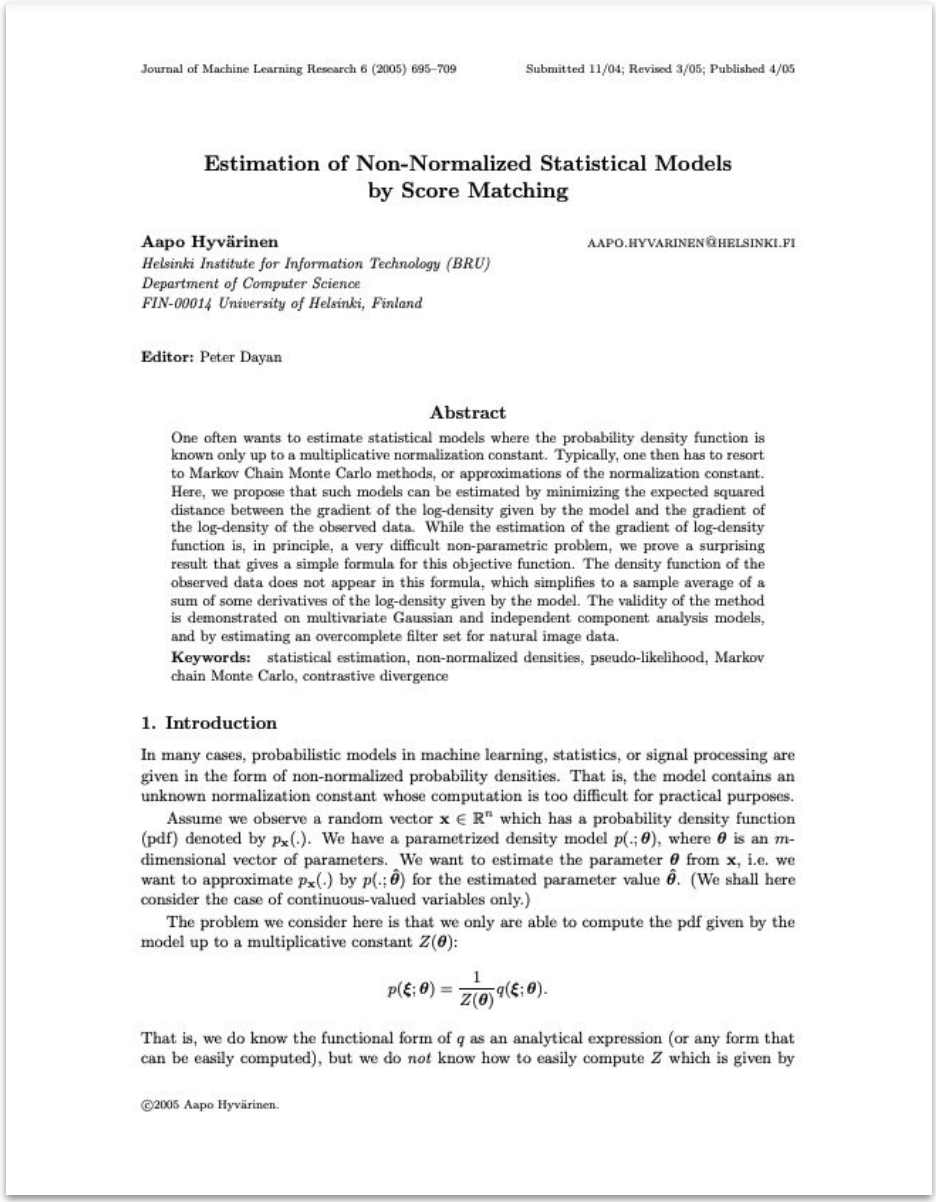
However, there is an issue: we don't know the ground truth PDF $p(x)$ and ground truth data score $\nabla_x \log p(x)$).

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Score-based Generative Models

Solution: Use a technique called **Score Matching.**

This will let us minimize the Fisher divergence without knowledge of the ground-truth data score.



Journal of Machine Learning Research 6 (2005) 695–709 Submitted 11/04; Revised 3/05; Published 4/05

Estimation of Non-Normalized Statistical Models by Score Matching

Aapo Hyvärinen AAPO.HYVARINEN@HELSINKI.FI
Helsinki Institute for Information Technology (BRU)
Department of Computer Science
FIN-00014 University of Helsinki, Finland

Editor: Peter Dayan

Abstract

One often wants to estimate statistical models where the probability density function is known only up to a multiplicative normalization constant. Typically, one then has to resort to Markov Chain Monte Carlo methods, or approximations of the normalization constant. Here, we propose that such models can be estimated by minimizing the expected squared distance between the gradient of the log-density given by the model and the gradient of the log-density of the observed data. While the estimation of the gradient of log-density function is, in principle, a very difficult non-parametric problem, we prove a surprising result that gives a simple formula for this objective function. The density function of the observed data does not appear in this formula, which simplifies to a sample average of a sum of some derivatives of the log-density given by the model. The validity of the method is demonstrated on multivariate Gaussian and independent component analysis models, and by estimating an overcomplete filter set for natural image data.

Keywords: statistical estimation, non-normalized densities, pseudo-likelihood, Markov chain Monte Carlo, contrastive divergence

1. Introduction

In many cases, probabilistic models in machine learning, statistics, or signal processing are given in the form of non-normalized probability densities. That is, the model contains an unknown normalization constant whose computation is too difficult for practical purposes.

Assume we observe a random vector $\mathbf{x} \in \mathbb{R}^n$ which has a probability density function (pdf) denoted by $p_{\mathbf{x}}(.)$. We have a parametrized density model $p(.;\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is an m-dimensional vector of parameters. We want to estimate the parameter $\boldsymbol{\theta}$ from $\mathbf{x}$, i.e. we want to approximate $p_{\mathbf{x}}(.)$ by $p(.;\hat{\boldsymbol{\theta}})$ for the estimated parameter value $\hat{\boldsymbol{\theta}}$. (We shall here consider the case of continuous-valued variables only.)

The problem we consider here is that we only are able to compute the pdf given by the model up to a multiplicative constant $Z(\boldsymbol{\theta})$:

$$p(\boldsymbol{\xi};\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} q(\boldsymbol{\xi};\boldsymbol{\theta}).$$

That is, we do know the functional form of q as an analytical expression (or any form that can be easily computed), but we do *not* know how to easily compute Z which is given by

Source: Yang Song, Generative Modeling by Estimating Gradients of the Data Distribution

Next Class

Next Class

Next class, we will cover:

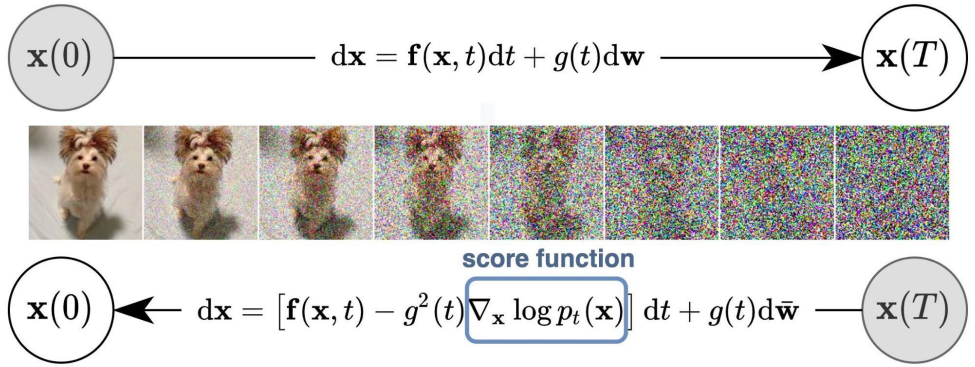
Next Class

Lecture: (Continued) score-based generative models and diffusion models.

Next Class

Lecture: (Continued) score-based generative models and diffusion models.

- The (Stein) score function, score matching.
- Score-based generative models.
- Sampling with Langevin Monte Carlo.
- Denoising diffusion probabilistic models.
- Learning with variational inference.

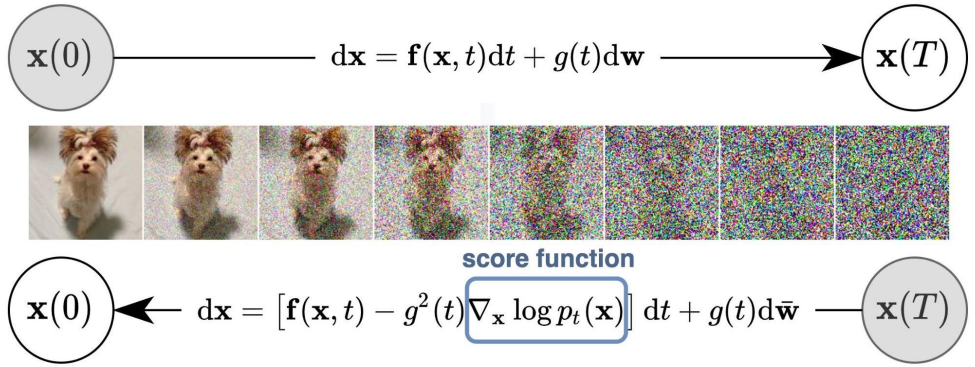


Source: Yang Song, "Generative Modeling by Estimating Gradients of the Data Distribution"

Next Class

Lecture: (Continued) score-based generative models and diffusion models.

- The (Stein) score function, score matching.
- Score-based generative models.
- Sampling with Langevin Monte Carlo.
- Denoising diffusion probabilistic models.
- Learning with variational inference.



Source: Yang Song, "Generative Modeling by Estimating Gradients of the Data Distribution"

After:

- Paper presentations from students.