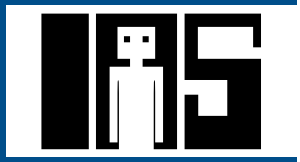
Imitation learning with energy based model

Imitationslernen mit energiebasiertem Modell
Master thesis by Jianhang He
Date of submission: March 17, 2021

1. Review: Prof. Dr. Jan Peters
Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT



Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 APB TU Darmstadt

Hiermit versichere ich, Jianhang He, die vorliegende Masterarbeit gemäß §22 Abs. 7 APB der TU Darmstadt ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, March 17, 2021

J. He

Contents

1 Introduction

1.1 Background

Artificial Intelligence has been always attracting lots of interest in the public and researchers, in particular since the invention of electronic computers. It is desired that a computer program achieves what a biological entity can do: sense the world, analyze information in (possibly) various modality and then take (re-)action based on internal rules, with the possibility to update its rules to increase the performance of some (pre-)defined goals.

In the sense of algorithm realization, machine learning, in particular, reinforcement learning targets directly learning this sense/planning/action behavior. With the current advance in computation ability, deep learning based RL approach demonstrates that an RL algorithm can surpass human-level performance, for example in Go [1].

However, it is known by researchers that RL algorithm is very sample inefficient [2] and that it depends on reaction with physical processes or simulator to generate samples. This feature makes requires one thing to gather a huge amount of data for training, and at each iteration the sample gathered from the previous steps may not be relevant, depending on the type of RL algorithm; for another, generating samples from physical processes or simulator would cost a huge amount of time and money, which out-weights time for training algorithm.

To overcome such a sampling problem, one particular idea is to provide a good starting strategy and then learn to (re-)act with the process. For example, a basis policy is learned from demonstration, on top of this basis policy, a reinforcement learning algorithm tries to modify the action produced by basis policy to improve or generalize the demonstrated policy. Such a basis policy learned from demonstration is what imitation learning focuses on.

1.2 Motivation

This work is motivated by the paper from Mordatch [3]. The paper demonstrates an application of energy based model by composing a joint probability distribution from several individual probability distributions. These probability distributions can be represented in a uniform framework, which is a Boltzmann distribution conditioned by various concepts embedded by a fixed length of code.

In one of the experiments in the paper, the author demonstrates that a random variable could move its position to fulfill a spatial relationship between relevant objects, given their positions and the encoding of a particular concept (for example “in-between”).

One particularly interesting aspect is that a trajectory could be generated by doing gradient ascent w.r.t. input variables of an energy based model. After several iterations, the variable configuration is shifted such that it acquires the least energy: achieving maximum probability for a set of variable configurations.

It then comes to the question of whether a energy based model could be trained for trajectory generation. Such a model could be used for example in robotic such that a just-in-time trajectory planner could be implemented with low computation cost by doing gradient ascent on that energy based model.

1.3 Contribution

In the master thesis, energy based model is applied for imitation learning. A model architecture is proposed for learning a policy by behavior cloning method. The training dynamic for an energy based model, as well as the application of SGLD sampler is reviewed, discussed and tested. An algorithm for generating trajectory on a energy based model is proposed. Some metrics for diagnosing probability distribution convergence, MCMC chain convergence, are experimented with. The model is applied to four dataset to demonstrate its ability to learn state-action mapping.

Structure of the thesis Chapter 2 introduces energy based model, sampling methods on probability distributions as well as imitation learning. Chapter 3 introduces an energy model for generating trajectory using stochastic gradient Langevin dynamics. Chapter 4 gives the experiment results and discussions. Lastly chapter 5 concludes the thesis.

2 Foundations

2.1 Probabilistic inference

The term *probabilistic inference* [4] usually refers to inference on the probabilistic graphical model. But the idea of querying on the model should be transferable to other types of models. An energy based model supports querying the probability given the configuration of input variable.

Another type of inference task is relevant to trajectory sampling: given some observed variable, find out the most probable value of those unobserved variables. In terms of trajectory sampling, the observed variable could be the state of the current system, and the action is the variable to be queried.

The criterion for *most probable* could be reformulated as finding an optimal configuration of input variables that maximize some kinds of criterion.

The following point estimator gives an optimal solution in terms of probability:

$$^{\mathrm{ML}}a = \mathrm{ML}(p, x) = \arg\max_a p(a|x)$$

This estimator gives the most probable input configuration given current state.

Another type of point estimator considers also the a prior probability distribution:

$$^{\mathrm{MAP}}a = \mathrm{MAP}(p, x) = \arg\max_a p(a|x)p_0(a)$$

If the à prior probability distribution evolves over time, the action represented by an energy based model distribution $p_\theta(x, a)$ could be modified without tuning the model parameter. For example a Bayesian filter could be written in this form.

Lastly, a point estimator could also consider the distribution of the probability. The expectation estimator in section 3.3 shows an application of such point estimator.

$$^{\mathrm{EXP}}a = \mathrm{EXP}(p, x) = \mathbb{E}_a\{p(a|x)\}$$

The expectation could be computed by sampling on the distribution, an efficient method is SGLD sampler, which is discussed in section 2.4

In terms of energy based model, $^{\mathrm{ML}}a$ requires fewer computation resources than $^{\mathrm{MAP}}a$ because the partition function doesn't influence the estimation result and thus could be skipped. For input variable that is usually high dimensional, this partition function is computationally intractable. The expectation estimator minimizes a Bayes risk function implicitly [5].

2.2 Energy based model

2.2.1 Compatibility

An energy based model [6] describes the degree of compatibility given a particular configuration of a set of variables. The term "compatibility" means that an energy based model itself is not constrained to modelling a probability distribution.

As an example, given an image and its duplicate with additive noise, the energy of such an image pair could be defined as highly compatible because they represent the same object. And an image pair that consists of a car, and a human, can be defined as not compatible as they describe different classes of objects.

The "compatibility" itself is not limited to one definition. Suppose in the previous scenario, in which different objects are captured in the image pair. If we define the "compatibility" to be that some spatial property, for example, we can define two objects are compatible because both of them are standing on the earth, or both of them is in between objects that belong to their same category.

To be more concrete: if in the first picture, a car is in between its leading car and following car, and in the second picture, a human is sitting in front of and behind other people, then an energy could be structured such that they fulfill the spatial property that these two objects are compatible.

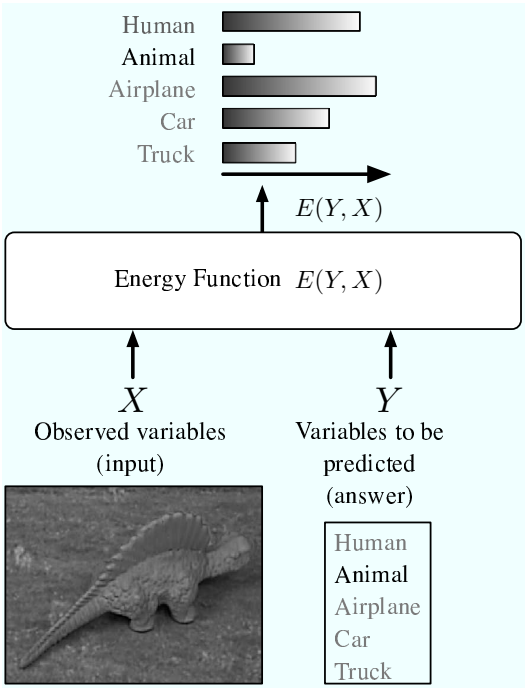


Figure 2.1: Compatibility of an image and a label. The compatibility is not necessarily a probability, for the purpose of classification, only the relative value makes a difference. The picture is taken from [6].

Speaking of "compatibility", it may remind the reader of the compatibility in the sense of electrical/mechanical/software interface. One may even think that compatibility is not limited to object, but could be generalized to also include time-series information. Actually, if a time-series could be embedded as a vector of real number (an object), then an energy base model could express the compatibility of this object with some other objects.

2.2.2 Mathematical formulation

From the above discussion, it is obvious that energy based model is a very general framework to describe the relationship between objects. Next, we introduce a formal description of energy based model.

Mathematically, an energy based model has the following form, given two sets of variables $X \in \mathbb{R}^m$ and $Y \in \mathbb{R}^n$:

$$E : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$$

Note that in the above formulation, the energy can be mapped to a negative number as there's no constraint of the image domain of such a function. This seems to be counterintuitive because it contradicts common sense that negative energy does not exist in the reality.

It could be justified as that the energy level is below the baseline energy level. Another possible interpretation is that the model is predicting the logarithm of energy, not the energy itself.

The latter interpretation is applied in the following chapter and in the implementation for the following reasons:

- when interpreted in a probabilistic context, a logarithm number represents a larger dynamical range than a decimal number.
- it guarantees the model produces a strict positive number, enforces less constraint on model output, making it easier to produces a feasible solution
- it reduces the requirements for defining a feasible starting solution for the problem to be solved, makes no special assumption on model architecture, etc.

As stated in section 2.2.1, an energy based model models the degree of "compatibility" given a variable configuration. When the energy is normalized, this compatibility could be interpreted as a probability measure.

A probability distribution described by an energy based model has the following form.

$$p_\theta(x) = \frac{1}{Z_\theta} \exp\{-\frac{1}{T} E_\theta(x)\} \tag{2.1}$$

Such type of distribution is first applied in the physic with the name "Maxwell-Boltzmann distribution" to describe the particle state distribution.

The negative sign in the exponential term indicates that the more energy a state x has, the more unlikely such a state would appear. The variable T is interpreted as *temperature*, but is mostly set to one for the sake of simplicity in the field of machine learning research. Lastly, the denominator Z, also called *partition function*, ensures that the energy exponential sums up to one. It is defined as follows:

$$Z_\theta = \int_{x' \in \mathcal{X}} \exp\{-\frac{1}{T} E_\theta(x')\} \mathrm{dx}'$$

One obstacle to apply energy based model as a tool for describing probability density is to calculated the denominator. The reason is that it involves integration over all possible state, which makes the computation intractable if the set of the possible state is high dimensional. The phenomenon that the computation expense grows exponentially w.r.t. dimension is well known as *curse of dimensionality*[7].

To compute the probability density, several methods are proposed, either exactly or approximately. Such methods are discussed in the following section 2.1.

2.2.3 Architecture

Energy based model could be represented variously. In the equation 2.1, the energy based model is parameterized by a vector $\theta \in \mathbb{R}^p$, and is commonly be represented by a (deep) neural network [8, 9] recently.

A neural network is a general purpose function approximator that is capable to approximate arbitrary complex functions given sufficient parameters. Depending on the type of

activation function, the number of hidden layers as well as the number of nodes in each layer, the capacity of a neural network varies [10].

Another type of architecture to represent an energy based model is the *Boltzmann machine* [11, 12]. A Boltzmann machine represents the energy function by a linear combination of input variables and their quadratic combinations. An extension of such type of model includes Markov random field. In this master thesis, we are only interested in an energy based model that is represented by a (deep) neural network.

2.3 Training

Training an energy model is known to be quite challenging. In this section, we would provide an overview of the existing method for training energy based model.

2.3.1 Overview

Recently, [13] provides a tutorial for the current state-of-art methods for training an energy model. It could be categorized into three categories:

- an objective function that optimizes maximum likelihood approximately.
- an auxiliary objective function that is based on necessarily condition, for example, Stein's identity.
- methods that involve external sampling networks, for example, adversarial training.

We would describe the related works in these categories in the following sub-sections.

[14] argues that every classifier could be treated as an energy based model, which provides a way to workaround converting an energy based model from an existing classifier. In appendix H, the author provides valuable review and experience on the recent approach to handle the challenge of training energy based model with MCMC methods.

Energy based model is widely adopted in image denoising, image generation, etc. However, applying it for regression is not until recently being researched. [15] evaluates different training approaches for 1d regression task as well as object tracking. Our method of training an energy based model could also be categorized into a regression task.

[16] shares tips and failures for training energy based model in the appendix A.12. The last tip comments on SGLD noise level matches the result of section 4.4 but each dataset could have a different noise level and a fixed step length.

Instead of training the energy based model to approximate the data distribution, [17] provides a new perspective to model the SGLD transition kernel is dependent on the parameter of an energy based model. Instead of learning a static model, the author proposes to learn a dynamical sampler.

2.3.2 Maximum likelihood estimation

As a starting point, training energy based model could be seen as a problem of parameter estimation. The classical maximum likelihood method formulates the problem as follows:

$$\theta = \arg\min_{\theta} {}^{\mathcal{MLE}}\mathcal{L}(\theta) = \arg\min_{\theta} -\Pi_{i=0}^{N} \log p_\theta({}^{\text{demo}}x^i, {}^{\text{demo}}a^i)$$

where ${}^{\text{demo}}x^i, {}^{\text{demo}}a^i$ are samples from expert trajectories.

Apply the gradient operator on each data point, we got the following expression to evaluate the gradient:

$$\nabla_\theta \log p_\theta(x, a) = -\nabla_\theta E_\theta(x, a) - \nabla_\theta \log Z_\theta$$

While the first term involves a forward pass of the neural network on the data point, the second term does not depend on data. However, it is the second term that makes the computation intractable for high dimensional model.

By exploiting the definition of Boltzmann distribution, the gradient of partition function could be expressed as the following

$$\log Z_\theta = \mathbb{E}_{\hat{x},\hat{a}\sim p_\theta(\cdot,\cdot)} \{-\nabla_\theta E_\theta(\hat{x}, \hat{a})\}$$

Most approaches that work on MLE tries to find an efficient sampling method, for example, MCMC. Discussion about MCMC, in particular stochastic gradient Langevin dynamic is discussed in section 2.4.

Contrastive divergence & noise initialization An approach to estimate this expectation is *contrastive divergence* by running a fixed step of MCMC that is initialized at the sampled data. [13]

This approach is straightforward and easy to follow. But as discussed in section 3.3, it is more desired to initialize the MCMC chain with as little à prior information as needed to increase its validity region that the policy distribution could sample action from.

Such a method that initializes MCMC from noise is called *noise initialization*. In [18] the authors propose the first ConvNet that could generate realistic images by training a model using noise initialization.

The author gives another reason for using noise initialization: since the starting position of a Markov chain is the stationary distribution, a trivial distribution such as $p_\theta = \text{const.}$ could also produce sample distribution that is close to the data distribution if the MCMC doesn't transient at all.

By estimating the gradient using MCMC sampling, the objective function of contrastive divergence could now be written as:

$$^{\mathcal{AML}}\mathcal{L}(\theta) = \mathbb{E}_{x,a\sim^{\text{demo}}\mathcal{D}}\left\{E_\theta(x,a)\right\} - \mathbb{E}_{x,a\sim p_\theta(\cdot,\cdot)}\left\{E_\theta(x,a)\right\}$$

Training convergence By using such objective function, the loss curve should be interpreted in a way that is different from MLE loss.

[18] states that the loss reaches 0, if the following conditions hold:

- the model distribution p_θ learns the data distribution $^{\text{demo}}\mathcal{D}$
- MCMC converges to its stationary distribution before sampling starts

When the loss oscillates around 0, [18] thinks that the training process stabilized, however, it could be a result that p_θ learns how to adjust its distribution to adapt to the sample, such that the sampler could produce samples that are close to the data distribution.

It is stated that when p_θ converges to data distribution, then doing long run MCMC, the sampled distribution should still close to the data distribution. This condition could be used to check the convergence of p_θ.

Variational maximum likelihood [19] reexamines the logarithm of partition function, then proposes that this quantity maximizes an optimization problem. An additional sampler is required, and it is trained in a GAN-like scheme which may cause difficulty in convergence. Thirdly, in the training objective, an entropy regularization is added which could be hard to estimate. However, the author states that such a method contains the advantages of MCMC method, score matching method as well as noise contrastive estimation. The expense is to train an additional data sampler. To be concrete, it shows faster and stabler convergence during training.

Diffusion recovery likelihood [20] presents an idea of learning a conditional distribution that is depending on a perturbed dataset. Multi-level noise makes the dataset gradually diffuses in a Gaussian shape distribution.

It is argued that learning a energy based model that approximates a lower variance noised data distribution, conditioned by a higher variance data, could be easier to train.

It demonstrates the ability that an energy based model could generate sample from a distribution with discontinuity, by progressively generating a sample, starting from a distribution conditioned by a large noise variance, in which case the conditioned variable approximates to Gaussian noise, and iteratively reducing the noise variance, in which case is a distribution that approximates to the data distribution.

f-Divergence minimization [21] reviews the problem of learning distribution. In the paper, both the I-projection and M-projection are discussed and demonstrated why one particular projection is easier to compute than another. The author provides an in-depth theoretical analysis about training convergence.

Improved contrastive divergence training Recently, [22] argues that the contrastive divergence objective is missing a term that could improve training stability. They propose that the KL divergence of finite step MCMC distribution to data distribution should also be considered in the training objective.

It was found that this new loss term would incorporate self-attention and normalization. Spectral normalization is needed for a stabler training dynamic if this loss term is absent.

2.3.3 Auxiliary objective function

There's another method that uses identity matching itself as an objective function, which will only be satisfied when two distributions are identical, to train the energy based model.

Score matching The first order gradient of a distribution is call *score* function. This method uses score function equivalence to train the energy based model. It minimizes the Fisher divergence of two distributions. [23] It requires computing the trace of second-order derivative, which is still computational heavy. But the advantage is that the partition function or its higher-order derivatives are not involved in training.

[9] provides experiments of training a deep energy based model with score matching objective. The author reinterprets this kind of method as a one-step denoising procedure.

Denoising score mathing [24] is another approach to evaluate the equivalence. It does not require one to compute second order derivative. It evaluates the Fisher divergence by using a noise perturbed dataset.

[25] provides an idea to use multi-level noise to learn a series of energy based models that is considering the noise level as an additional input variable. A similar idea of perturbing the dataset could be seen in *diffusion recovery likelihood*.

Slicde score matching [26] recently propose to evaluate the Fisher divergence approximately by evaluating the equivalence of score function with randomly selected vectors. It is more computationally efficient, in the sense that the complexity for evaluating the projected trace of second-order derivative could be simplified to $\mathcal{O}(n^2)$, where n is the dimension of input variable. [13]

Stein's discrepancy [27] proposed Stein's identity that evaluates an unnormalized distribution on a dataset. The distribution function inside the expectation is replaced with the distribution of the energy based model. This identity is evaluated in a set of function and is expected that the identity holds when energy base model approximates the data distribution. [28]

[28] proposes to describe the set of functions with a parameterized neural network. It gives a discussion on what kinds of functional space such a network should represent.

The advantage of using Stein's discrepancy is it does not require sampling on the energy based model. This method has relation to GAN-like model and could suffer from training instability.

[29] uses a special type of kernel functions to describe the functional space, such that the trace term in the Stein's discrepancy becomes a constant.

[30] uses stein's variational gradient descent method to guide the update of a generator, which generates samples from energy based model. The energy based model itself is trained with the contrastive divergence objective.

2.3.4 Noise contrastive estimation

Noise contrastive estimation [31] minimizes the KL divergence between two conditional distributions. The distribution of energy based model, as well as data distribution, is conditioned by a noise distribution, possibly a Gaussian distribution.

The advantage of learning the noised version of conditional distribution could be that it learns the normalizing constant of the energy based model as a by-product. [13]

Classifiers could be seen as an energy based model as state by [14]. In their experiment it was shown that the classifiers are more robust to adversarial input, when the model is trained using a perturbed dataset. It could be further investigated to see what result to be expected if using a perturbed dataset to train the noise contrastive estimation.

2.4 Stochastic gradient Langevin dynamic

Popular MCMC methods have high computational complexity when applied to high dimensional distributions. Recently [32] applies stochastic gradient Langevin dynamic sampling method to probability distribution represented by machine learning models.

Applying SGLD on energy based model avoids calculating the partition function, making inference on high dimensional less computational demanding. It thus makes energy based model applied to many applications: for example synthesizing a molecular structure [33].

Sampling a trajectory on an energy based model could be express by the following equation:

$$a^{l+1} \leftarrow a^l - \frac{\alpha^2}{2}\nabla_a E_\theta(x, a^l) + \alpha\epsilon, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Two parameters could be tuned for such a sampling operator:

- ${}^{\text{SGLD}}K$ gives the step length of a MCMC chain, for a sufficiently large K the MCMC chain converges to its stationary distribution, the distribution of the energy based model. [34]
- ${}^{\text{SGLD}}\alpha$ defines the noise level of a MCMC chain. It turns a stochastic gradient method into a MCMC chain, the higher is α, the more randomness it gives to this MCMC chain. Correct implementation requires that α be annealing, but in practice, it was found out that a constant noise level could generate samples that are close to the dataset. [16, 35, 36]

Noise level Giving an incorrect noise level would get a sampled distribution that does not match the distribution of the model. [35] gives a theoretical analysis of SGLD algorithm, and it proposes that the noise level could be chosen as

$${}^{\text{SGLD}}\alpha \leftarrow \frac{\eta}{N}, \eta \in (0, \frac{1}{2\tilde{L}}]$$

by assuming that the energy function E_θ is $\tilde{L}$-gradient Lipschitz.

Chapter 4 discusses the assumptions made in modeling the policy distribution. The choice for noise level should be oriented by the range of feasible action instead. But it could be enforced as a constraint in the objective function, such that the sampler could faithfully generate samples from the energy based model.

Average MCMC chain length It is not easy to find out whether a MCMC chain converges to its stationary distribution, nor is it easy to find out whether an energy based model converges to its target distribution. [18] proposes an indicator for judging which phase a training process is currently in.

The average MCMC chain length could be defined as the following:

$$R := {}^{\text{SGLD}}K^{-1} \sum_{l=0}^{{}^{\text{SGLD}}K} v^l = {}^{\text{SGLD}}K^{-1} \sum_{l=0}^{{}^{\text{SGLD}}K} \left\| a^{l+1} - a^l \right\|$$

Figure 2.2 shows the curve of a converging and non-converging MCMC chain. It is evident that there is a rising phase for a converging MCMC chain.

It is expected that R has the following properties, which is useful for diagnosing training dynamic. Chapter 4 gives an application on tuning model architecture, noise level with this metric.

- when MCMC chain starts at normal distribution, during training process, it is expected that R increases until it converges to a value L.
- if the distribution of energy based model converges to the data distribution, then by increasing ${}^{\text{SGLD}}K \to {}^{\text{SGLD}}K + S$, it is expected that $L \approx L + Sf({}^{\text{SGLD}})$, where $f(\sigma)$ is a function that calculates the average length of a MCMC chain on Gaussian distribution with variance σ^2.

Mixing rate & burn in Before MCMC chain converges to its stationary distribution, there's a transient phase, in which the sample distribution is not reflecting the true distribution. Those samples should be ignored when doing MCMC sampling. This process of waiting for convergence is sometimes referred to as the burn-in phase in the literature. [16, 18]

The mixing rate gives an indicator of how many MCMC iterations are required for a MCMC chain to converge to its stationary distribution. It depends on which type of sampler is used, in which distribution the MCMC chain starts from, and the complexity of the stationary distribution.

In *contrastive divergence* the MCMC chain starts at the data distribution, which would make the exploration more difficult because it requires that the model generate a large enough gradient such that the area around the mode could be explored, otherwise the learned distribution could be flat or a constant, given a very small noise level and short-run MCMC. This phenomenon is discussed in chapter 4.

It would also make the curve of figure 2.2 a bit different from the MCMC chain that starts from a normal distribution. But the difference is not investigated in this master thesis.

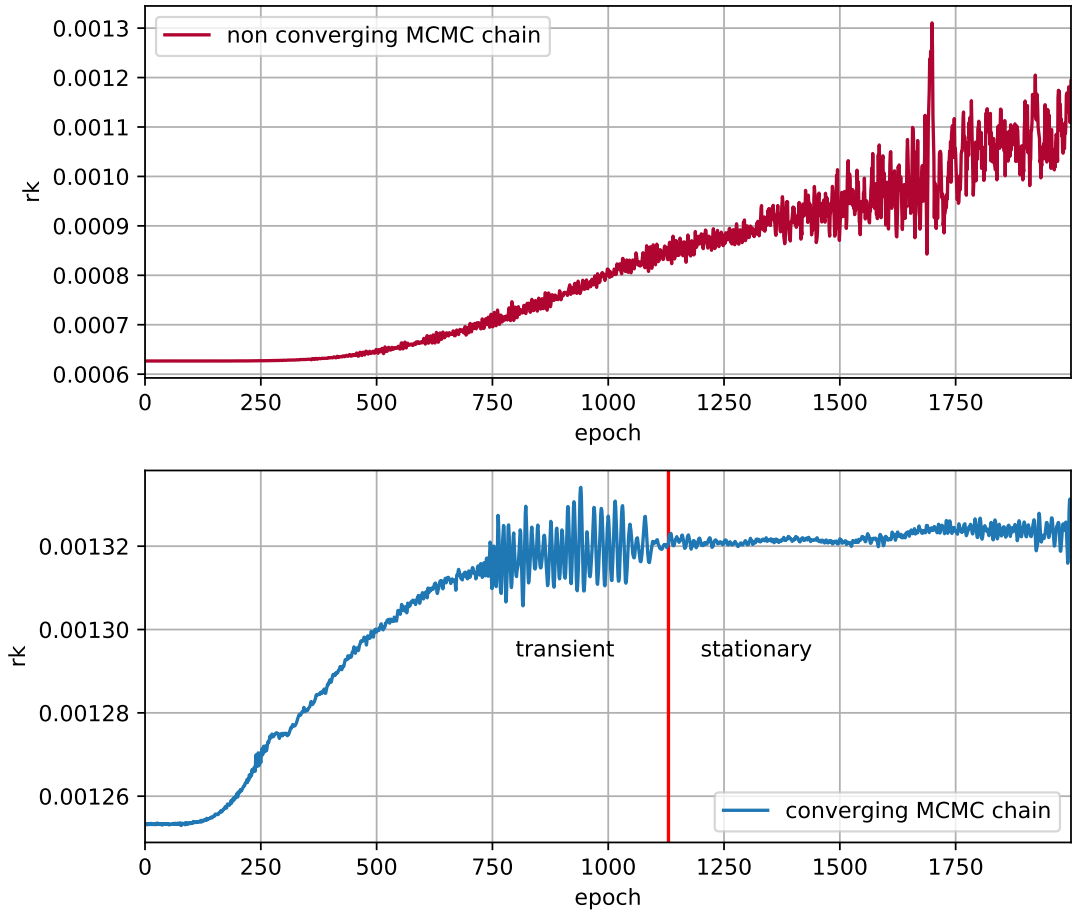


Figure 2.2: Average MCMC chain length R curve of converging and non-converging MCMC chain

Persistence contrastive divergence starts MCMC chain from the previous training step. It is hoped that the distribution of the model doesn't change much. And PCD starts like random initialization. It helps to accelerate the MCMC chain convergence [18].

Replay buffer In reinforcement learning, a replay buffer keeps the state, action from the previous steps for training the model. A replay buffer could also be applied in MCMC chain [16] such that the persistence contrastive divergence could have more chance to switch between different modes of the distribution. For example with a probability ε, an MCMC chain in the replay buffer is resetting to its staring distribution, the normal distribution.

2.5 Imitation Learning

A general setting for imitation is that the expert produces demonstrated trajectories $\{^{\text{demo}}\tau_i\}_{i=1}^{N}$ and no other information about the reward function, etc. This setting is adopted in this master thesis.

There are two categories for imitation learning, behavior cloning, and inverse optimal control. [37] provides a thorough review of existing methods. Each type of algorithm is briefly mentioned here.

Behaviour cloning It's behaviour cloning goal to learn a policy given that a trajectory consists of a sequence of state-action pair $\{^{\text{demo}}(x_i^j, a_i^j)\}_{j=1}^{T_i}{}_{i=1}^{N}$. Compared to the inverse optimal control method, behavior cloning method treats the problem as doing nonlinear regression in action-state space.

Inverse optimal control In inverse optimal control (inverse reinforcement learning), a reward function would be recovered from demonstrated trajectories. The problem is known as an ill-posed problem. Various types of regularization techniques are proposed to impose the solution uniqueness, for example, maximum margin by [38] and maximum entropy [39].

3 Imitation learning with energy based model

In this chapter, a model for imitation learning using energy based model is proposed. First, the problem of imitation learning is described. It raises three questions that would be answered by the following sections: model design for representing a probability distribution; sampling method for trajectory; lastly the method for training the model.

3.1 Problem statement

We are interested in designing an energy based model to learn a state-action mapping demonstrated by an expert, in the form of a set of trajectories $\{^{\text{demo}}\tau^i\}_{i=1}^N$.

A trajectories $^{\text{demo}}\tau^i$ consists of a series of states: $^{\text{demo}}\tau^i = ((x_0^i, a_0^i), (x_1^i, a_1^i), \dots, (x_{n_i}^i, a_{n_i}^i))$.

1. each trajectory does not have necessarily the same length: $n_i \not\equiv \text{const.}$
2. the distance between adjacent states is not necessarily equidistant: $\|x_j^i - x_{j+1}^i\| \not\equiv \text{const.}$
3. each trajectory is normalized to the same scale: $|\text{length}(^{\text{demo}}\tau^i) - \text{mean}(\text{length}(^{\text{demo}}\tau^i))| < \delta$, where $\text{length}(^{\text{demo}}\tau^i) := \sum_j \|x_j^i - x_{j+1}^i\|$
4. all trajectories starts at coordinate orgin $x_0^i = (0, 0), i = 1, \dots, N$

This means the trajectories in figure 3.1 are valid trajectories in a demo set.

It is desired that an energy based model learns a policy $\pi : x \mapsto a$ from demonstrated trajectories, such that when a trajectory is sampled using this policy starting at origin $(0, 0)$, the sampled trajectory looks similar to the demonstrated trajectory. The procedure for sampling trajectory is described in algorithm 1.

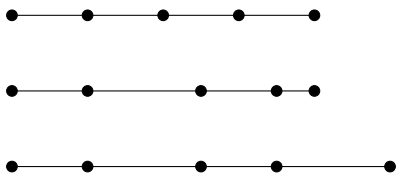


Figure 3.1: Possible demo trajectories

This algorithm serves as a framework for the following discussions. The problems to be focused on includes:

1. how to use energy based model to express a policy?
2. how to sample an action given the current state?
3. how to learn the distribution from demonstrated trajectories?

Data: Policy expressed by an energy based model $\pi(a|x)$, system dynamic $\mathcal{T}(x_{j+1}|x_j, a_j)$
Input: Trajectory length t
Output: Sampled trajectory τ
$\tau_0 \leftarrow \mathbf{0}$;
for $j \leftarrow 0$ *to* $t-1$ **do**
  $a_j \sim \pi(\cdot|x_j)$;
  $x_{j+1} \leftarrow \mathcal{T}(x_j, a_j)$;
  $\tau_{j+1} \leftarrow x_{j+1}$;
end

Algorithm 1: Meta algorithm for sampling a trajectory given policy $\pi(a|x)$

3.1.1 Energy based model for behavior cloning

The energy based model in this master thesis is used for behavior cloning.

When ML estimation is applied to sample a trajectory on the energy based model, then such a method is a greedy algorithm because it looks for the best solution given the current state. In section 3.3, it would be demonstrated that such a method produces a trajectory that deviates from the demonstrated trajectory in the long run. The same problem is discussed in [40], which states that given enough dataset and small horizon, a series

of policies could be trained such that the accumulated error in during the sequential decision could be corrected by these policies. The action keeps the state that is close to the demonstrated trajectories.

The reason the trajectory is not most probably could be that the algorithm is short-sighted: it doesn't consider the accumulated joint probability of a trajectory.

For a trajectory τ that is consisted of a series of random variables $(x^0, a^0), (x^1, a^1), ..., (x^t, a^t)$, the joint probability is

$$p({}^{\text{demo}}\tau) = p(x^0) \prod_{j=1}^{T} \mathcal{T}(x^j \mid x^{j-1}, a^{j-1}) {}^{\text{demo}}\pi(a^{j-1} \mid x^{j-1})$$

We now apply the ML estimator to sample a trajectory with the same starting position x^0:

$$p({}^{\text{ML}}\tau) = p(x^0) \prod_{j=1}^{T} \mathcal{T}(x^j \mid x^{j-1}, {}^{\text{ML}}a^{j-1}) \text{ML}(E_\theta, x^{j-1}) \tag{3.1}$$

- equation 3.1 shows that ML estimator represents a deterministic policy
- a stochastic policy could be implemented by applying SGLD on the energy based model for querying an action

3.2 Modelling policy

A policy is a function that maps a state to action. In the control engineering domain, a policy is also called a controller that outputs a control signal given the current state of a system. Such a control signal is deterministic.

But we would like to model a policy to be stochastic, for the reason that an energy based model expresses a probability distribution that per se contains stochastic property. Additionally, a stochastic policy is more general and also frequently discussed in the reinforcement learning algorithm. By learning a stochastic policy, it would be beneficial to see such a policy could be assigned as an initialization policy for a reinforcement learning algorithm.

We reformulate formulate the input variable x in formula 2.1 to (x, a), and model the function as a conditional probability distribution:

$$\pi_\theta(a|x) := p_\theta(a|x) = \frac{1}{Z_\theta}\exp\{-\frac{1}{T}E_\theta(x,a)\} \tag{3.2}$$

It makes no difference whether we use the energy function to model a joint distribution $p_\theta(x,a)$ or a conditional distribution $p_\theta(a|x)$ for the sake of sampling an action, which would be discussed in more detail in 3.3.

From the assumptions on trajectories in section 3.1, and the sampling algorithm 1 we can expect that action is finite and has its mode within the region defined the maximum length of adjacent states, a.k.a. step size.

$$B_\varepsilon(0) := \{a : \|a - 0\| < \varepsilon\} \tag{3.3}$$

From the viewpoint of geometry, the set of action defined above is a sphere in high dimensional space. Its radius is defined as:

$$\varepsilon := 2\max_{i,j}\|x_j^i - x_{j+1}^i\|$$

It is double the size of the maximum step size in the dataset. The scale factor 2 in defining the radius from step size has no particular reason but could be further discussed to make a reasonable statement.

The range of most possible actions could help design model architecture and tune the model by inspecting training diagnostic information (diagram, loss curve, gradient norm, etc.).

Additional assumption on demonstrated trajectories In the following discussion, we impose an additional constraint on the trajectory, in the hope that the distribution of the policy function is uni-modal:

The shape of the demonstrated trajectories looks similar

This statement is ambiguous but could be assumed as long as the trajectories are generated by a human expert that repeats its demonstration multiple times.

Modelling action implicitly In the formula 3.2, the action modelled as an input vector of a probability distribution. It raises the question of whether there exists another way for generating the action, for example, use the gradient information?

$$a = \nabla_x E_\theta(x) \tag{3.4}$$

In the case above, the probability distribution the energy based model represents becomes harder to interpret. One cannot say that one state is more likely to be visited than another by comparing its energy level. This thesis argues that because in the reality, the feasible region constrains where a state could be reached.

On the other hand, formula 3.4 has the same form as a *control lyapunov function* which describes a control law that stabilize some specific function.

Borrowing the idea from control theory, we think that this represents a vector field, in which the expert trajectories could be generated by applying formula 3.4 as the control law. The energy function here could be interpreted as a value function in reinforcement learning.

In conclusion, modelling action with energy function implicitly using formula 3.4 poses an inverse optimal control problem. We think that there exists a number of literature on using energy based model to solve inverse optimal control or inverse reinforcement learning that are more involved than representing the action as the gradient of the energy function. Thus we won't further discuss the method of implicitly modeling the action.

3.2.1 Model architecture

Next we're going to describe the architecture of the energy based model. First to simplify the notation, we denote $-T^{-1}E_\theta(x, a)$ as $f_\theta(x, a)$ such that the probability distribution could be expressed as the following equation:

$$p_\theta(x, a) = \frac{1}{Z_\theta} \exp\{f_\theta(x, a)\}$$

The model consists of two networks. The `FE` network extract features from x and a to extend its dimension. The `FCN` network evaluates the energy given a set of feature configurations from the previous network output.

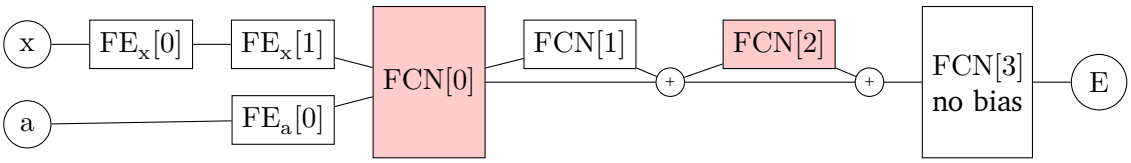


Figure 3.2: Model architecture of the energy based model. The red filled blocks are wrapped with spectral normalization. Both FE_x, FE_a and FCN are fully connected network. The activation function in FE_x network is LeakyReLU. The activation function in FE_a network is ELU. The activation function in FCN network is ReLU. The output layer has no bias term.

The FE network contains a dedicated extractor for each input vector because the input range for x and a could be very different: while a mostly lies in the region defined by equation 3.3, the range of x is unspecified. It is thus reasonable to first transform each of the inputs separately into a higher dimensional embedded space.

We denote each of the network to be $\mathrm{FE}_a : \mathcal{A} \to \mathbb{R}^{\mathrm{zdim}}$ and $\mathrm{FE}_x : \mathcal{X} \to \mathbb{R}^{\mathrm{zdim}}$.

The FCN network could be a 4 layer fully connected network that consists of an input layer, two hidden layers, and an output layer. The output layer predicts the logarithm of negative energy to simplify the notation of the energy based model expression. This network could be denoted as $\mathrm{FCN} : \mathbb{R}^{2\mathrm{zdim}} \to \mathbb{R}$. The dimension of the hidden layer is parameterized by a variable hdim.

The policy could be expressed by the following expression:

$$f_\theta(x, a) = \mathrm{FCN}(\mathrm{FE}_x(x), \mathrm{FE}_a(a))$$

Spectral normalization It is mentioned in [16] that spectral normalization helps improve training stability when contrastive divergence objective is used. During the experiment, it is found that applying such a normalization makes it difficult to train the energy based model, which could be observed by comparing the sampled action distribution between the network with/without applying spectral normalization at the same epoch. We consider applying it only at the first and third layer of FCN network.

$$\mathrm{FCN}[0] \leftarrow \mathrm{SPECTRAL\_NORMALIZATION}(\mathrm{FCN}[0])$$
$$\mathrm{FCN}[2] \leftarrow \mathrm{SPECTRAL\_NORMALIZATION}(\mathrm{FCN}[2])$$

SPECTRAL_NORMALIZATION is implemented by `pytorch` using power iteration to calculate spectral norm.

Residual layer Residual layer structure is applied in two hidden layers of FCN network, as did in [15].

$$\mathrm{out}[i] \leftarrow \mathrm{FCN}[i](\mathrm{out}[i-1]) + \mathrm{out}[i-1], i \in \{1, 2\}$$

The advantage for residual layer could be that the gradient pass through those hidden layers, thus reduces the effect of vanishing gradient. This could also be applied here for a shallow network.

According to the discussion on training dynamic of contrastive divergence objective in section 4.3.2, the loss level should be somehow balanced when converged, oscillating around zero. It causes a problem for updating the parameter of each network layer when the loss is small.

Although a lot of sample is generated from the model, because the majority of negative data points overlap to positive data points, the distribution divergence between the model and the data needs far more iteration than needed to accumulate enough gradient magnitude to update the parameter of the layer to adjust the shape of its distribution where few or even no data is sampled there.

3.3 Sampling action

Doing probabilistic inference on energy based model is required to find a suitable action given the current state.

By assumption, we know that the conditional distribution $p_\theta(\cdot|x)$ is a uni-modal distribution with its peak lies in the set $B_\varepsilon(0)$. We want to find the most probable action a_t that an expert would take given the current state x_t.

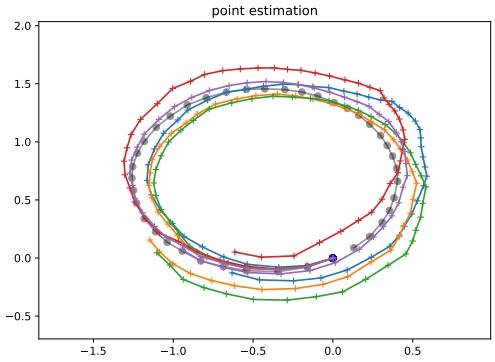


Figure 3.3: Action point estimation

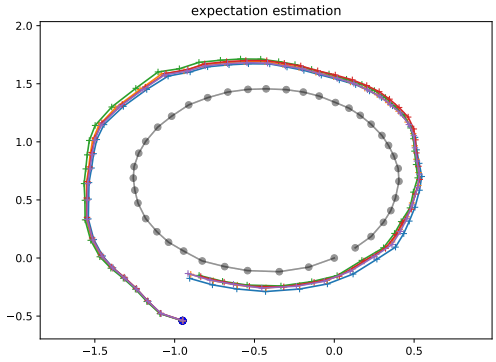


Figure 3.4: Action expectation estimation

Point estimation trajectory sampler That we compute a point estimation w.r.t. an input variable, and $\exp(\cdot)$ is a monotonically increasing function, we can neglect the partition function Z_θ and $\exp(\cdot)$ during the inference.

We got the first operator to acquire an action from energy based model:

$$a_t = \arg\max_a p_\theta(a|x_t) = \arg\max_a \frac{1}{Z_\theta} \exp\{f_\theta(x_t, a)\} = \arg\max_a f_\theta(x_t, a)$$

Note if we use the energy based model to represent a joint distribution, the inference equation remains the same. This justified that learning the joint distribution achieves the same result as learning the conditioned distribution.

By taking the most probable action, we could sample a trajectory that starts at the origin $(0, 0)$. During the experiment, it is found out that the trajectories sampled by algorithm 1 quickly diverge from the expert trajectory, as seen in 3.3. Discussion on such phenomenon could be seen in chapter 4.

Expectation estimation trajectory sampler To improve the quality of the sampled trajectory, we consider computing the expected action using SGLD instead of using the point estimation directly, the resulting trajectory could be seen in figure 3.4.

$$a_t = \mathbb{E}_{a \sim p_\theta(\cdot|x_t)}\{a\}$$

The modified algorithm for sampling trajectory is described in algorithm 2.

Data: Policy expressed by an energy based model $\pi(a|x)$, system dynamic $\mathcal{T}(x_{j+1}|x_j, a_j)$
Input: Trajectory length t, estimate population p, SGLD step K, SGLD noise level α
Output: Sampled trajectory τ

$\tau_0 \leftarrow \mathbf{0}$;
/* initialize candidate state */
$\hat{x}_0^l \leftarrow \varepsilon^l, \varepsilon \sim \mathcal{N}(\mathbf{0}, \alpha\mathbf{I}), l = 1, \dots, p$;
for $j \leftarrow 0$ *to* $t-1$ **do**
  /* initialize MCMC chain with random noise */
  $a_j^l \sim \mathcal{N}(\mathbf{0}, \alpha\mathbf{I}), l = 1, \dots, p$;
  /* perform SGLD steps */
  for $k \leftarrow 1$ *to* K **do**
    $a_j^l \leftarrow \mathrm{SGLD}(\pi, \hat{x}_j^l, a_j^l)$;
  end
  /* estimate expected action */
  $a_j \leftarrow p^{-1} \sum_{l=1}^{p} a_j^l$;
  /* sample state at next time step */
  $x_{j+1} \leftarrow \mathcal{T}(x_j, a_j)$;
  $\tau_{j+1} \leftarrow x_{j+1}$;
  /* update state population */
  $\hat{x}_{j+1}^l \leftarrow \mathcal{T}(x_j, a_j) + \varepsilon^l, \varepsilon \sim \mathcal{N}(\mathbf{0}, \alpha\mathbf{I}), l = 1, \dots, p$;
end

Algorithm 2: Sampling a trajectory using expectation estimation

Qualitatively it shows that the trajectory tends to converge to the expert trajectory when the state is not far from the demonstrated trajectories. A more detailed analysis of this sampler applied on different demonstrated trajectories are discussed in section 2.1.

Trajectory sampler for multi-modal distribution The method of expectation estimation could only applies to the dataset where trajectories holds the assumption given in section 3.1 and paragraph 3.2. If the uni-modal distribution on the action is violated, it's reasonable to conclude that the expected estimation won't give an action that matches the demonstrated policy. A sampling operator that supports sampling on a multi-modal distribution could be reserve for further investigation.

Such a trajectory sampling operator should fulfill certain properties. For example:

- it could determine on which mode it is currently in, based on the states from the previous time steps.
- it could be parameterized such that it chooses the mode based on various criterions, such as
  - most applied actions in the expert demonstrations
  - action that considers external reward function, e.g. take the action that minimizes risk / increases information gain when the entropy of the action distribution is above some threshold

3.4 Learning policy

In this section, the objective function for energy based model will be derivated. We start by minimizing the KL divergence between the expert policy and policy of energy based model.

$$\begin{aligned}\mathcal{L}(\theta) &= \mathrm{KL}(\pi^{\mathrm{demo}} \| \pi_\theta) \\ &= \mathbb{E}_{\pi^{\mathrm{demo}}}\{\log \pi^{\mathrm{demo}}(a|x)\} - \mathbb{E}_{\pi^{\mathrm{demo}}}\{\log \pi_\theta(a|x)\}\end{aligned}$$

Apply gradient operator ∇_θ

$$\begin{aligned}\nabla_\theta \mathcal{L}(\theta) &= -\mathbb{E}_{\pi^{\mathrm{demo}}}\{\nabla_\theta \log \pi_\theta(a|x)\} \\ &\approx -\frac{1}{N}\sum_{i=1}^{N} \sum_{x,a \in {}^{\mathrm{demo}}\tau^i} \nabla_\theta \log \pi_\theta(a|x) \\ &= -\frac{1}{N}\sum_{i=1}^{N} \sum_{x,a \in {}^{\mathrm{demo}}\tau^i} \nabla_\theta f_\theta(x,a) - \nabla_\theta \log Z_\theta \\ &= -\frac{1}{N}\sum_{i=1}^{N} \sum_{x,a \in {}^{\mathrm{demo}}\tau^i} \underbrace{\nabla_\theta f_\theta(x,a) - \nabla_\theta \mathbb{E}_{\pi_\theta}\{f_\theta(x,a)\}}_{\nabla_\theta \mathcal{L}(\theta;x,a)}\end{aligned}$$

We could rewrite the last equation by evaluating gradient on each data point in the dataset

$$\nabla_\theta \mathcal{L}(\theta; x, a) \approx \underbrace{\nabla_\theta f_\theta(x, a)}_{\text{positive gradient}} \underbrace{- \frac{1}{M} \sum_{\hat{x}, \hat{a} \in \text{SGLD}(\pi_\theta)} f_\theta(\hat{x}, \hat{a})}_{\text{negative gradient}} \tag{3.5}$$

The loss in equation 3.5 decreases the energy level on negative data point, while increasing the energy level on positive level. [18]

Sampling method Estimating the parameter θ by minimizing this loss function is equivalent to approximately maximizing the likelihood function.

The difficulty lies in sampling data points on the energy based model efficiently. Section 2.4 discusses techniques for efficient sampling. One particularly popular method in recent years is MCMC with Langevin dynamics because it could be applied to sample a high dimensional probability distribution. When the MCMC chain initializes at data points, then such a method is called *contrastive divergence* in the literature. [6]

MCMC initialization Because the equation 3.5 adjust energy level based on sampled location, it is desired to sample actions in regions where expert demonstration locates.

For sampling trajectory, it is desired to start the MCMC chain at the origin to minimize the requirements for à prior information when applying the imitated policy in regions where no expert demonstration exists.

If the assumption holds that the mode of the action is very likely to locate in the set $B_\varepsilon(0)$, then MCMC should generate a chain inside this region. In summary, we could safely generate initial actions that are within the set $B_\varepsilon(0)$, for example, sampling from a Gaussian distribution.

$$^0a_j^i \sim \mathcal{N}(\mathbf{0}, \frac{\alpha^2}{\|^{\text{demo}}a\|^2}\mathbf{I})$$

L1 and L2 regularization The output of the energy based model is not constraint by the objective function. On the other hand, the sampling method depends only on the energy level in the local neighborhood. It means that an energy model can be offset by a constant but still fulfills the optimal condition.

To ensure the numerical stability, as well as to reduce the local optimal solution, the bias parameter in the last layer of FCN is removed, and secondly, an L1 and L2 regularization is enforced on the energy level to keep the network prediction with the range that a `float32` could represent.

It is observed from the gradient norm that the bias parameter of $\mathrm{FCN}[3]$ is within the range of 1×10^{-4}, which could be neglected compared to the gradient norm in other layers, for instance as high as 1×10^{3}.

4 Experiment

4.1 Environment setup

The model architecture is described in figure 3.2. It is evaluated on four datasets, which could be seen in section 4.4. A list of description on hyperparmeter is given in table 4.1.

hypterpamater	description
`lr`	learning rate of Adam optimzer
`z_dim`	embedding space dimension
`h_dim`	dimension of hidden layer in FCN network
`reg_l1`	L1 regularization weight factor
`reg_l2`	L2 regularization weight factor
`sgld_K`	SGLD step length
`sgld_al`	SGLD noise level
`sgld_dim`	SGLD negative sample total number
`dataset_sigma`	variance of additive noise added to SVG datasets

Table 4.1: Hyperparmater description

The model is implemented using `pytorch` 1.8.0, and `pytorch-lightning` 1.2.3. The model is trained on Nvidia RTX 2080.

4.2 Model architecture tuning

Several experiments are conducted on `circle` dataset, to test if the spectral normalization makes a difference on training stability, as mentioned in [16]. It was found that applying this normalization on all layers of the FCN and FE_x network would cause the gradient

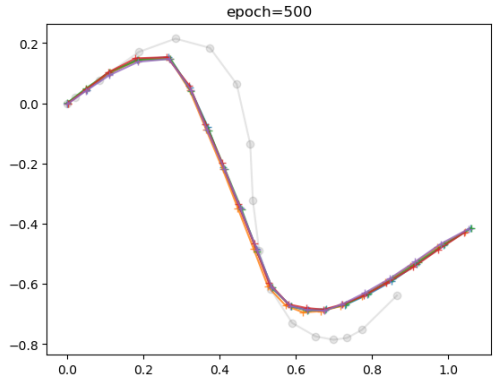


Figure 4.1: Sampled trajectory on `mouse1` dataset at 500 epoch, $\mathrm{hdim} = 16$, $\mathrm{zdim} = 128$

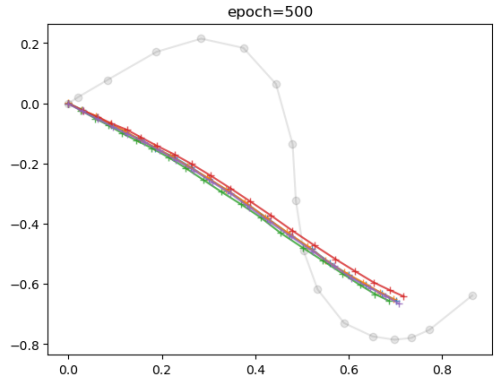


Figure 4.2: Sampled trajectory on `mouse1` dataset at 500 epoch, $\mathrm{hdim} = 8$, $\mathrm{zdim} = 64$

norm in each layer as small as 1×10^{-4}. The loss is thus kept in a small magnitude, making the training stagnate. The average MCMC chain length is within the range of the noise level throughout each epoch, supporting this argument.

The $\tanh$ activation keeps the magnitude of the loss function within a range of $\pm 1 \times 10^3$ when it was applied after the first layer of a three layer fully connected neural network when other layers apply ReLU as the activation function.

Training a three layer network on `path1` and `circle` requires up to 3×10^3 iteration, and the sampled trajectories still diverges from the expert demonstration. In comparison, the model in figure 3.2 produces trajectories on `circle` dataset after three epoch. It could be argued that embedding the state and action into a higher dimensional space is necessary.

The dimension of embedding space zdim and the dimension of the hidden layer hdim plays a role in convergence rate. Figure 4.2 and 4.1 shows the intermediate sampling result at 500 epoch. Clearly that the model with a higher dimension produces already trajectories that are similar to the expert demonstration from the point of view of a human. On the other hand, the model with half of the number of dimensions doesn't produce a comparable result when its epoch reaches 1000.

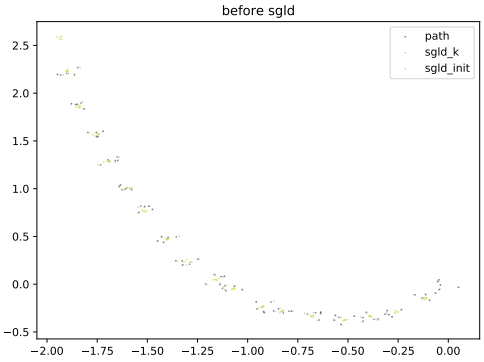


Figure 4.3: MCMC chain initialization

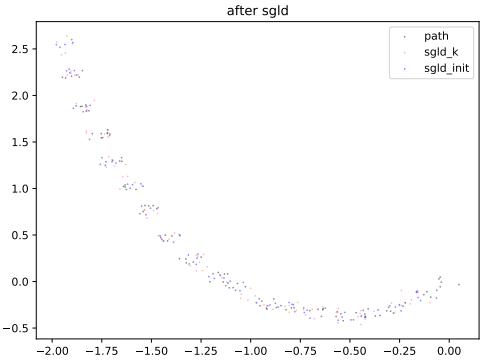


Figure 4.4: SGLD 100 step iteration

4.3 Training diagnostic

Energy based model is known to be hard to train. In this master thesis, we investigate some clues in the hope that these indicators serve as guidance for tuning the algorithm parameter in the early stage to avoid wasting time only to figure out that the model won't converge.

4.3.1 Tuning SGLD noise level

Tuning parameter is a challenge to deal with when applying SGLD sampler. Without a suitable value of noise level, it is not possible to acquire realistic sample data points.

When the noise level is too high, the gradient magnitude is overcome by the diffuse magnitude. The Markov chain tends to escape from its mode even it is initialized at the location of demonstrated actions and is unlikely stepped back given finite time and memory. The model could not get enough sample around the mode, but instead, the data distribution reflects more randomness, thus getting a biased density estimation.

The above mentioned problem could be found out by doing a sanity check. Figure 4.3 and figure 4.4 shows the SGLD iteration process.

In figure 4.3, `path` indicates the state in an expert trajectory. `sgld_init` shows the action of the expert policy takes. And `sgld_k` indicates that the action distribution

before running 100 steps of SGLD, when the MCMC chain is initialized using a Gaussian distribution.

When that the initial actions sampled from this Gaussian distribution satisfies the following criterion, meaning that the actions are not far away from current state x_j^i, where ${}^{\mathrm{SGLD}}\alpha$ is the SGLD noise level:

$$ {}^{0}a_j^i \in B_r(x_j^i), r := 10^{\mathrm{SGLD}}\alpha \tag{4.1} $$

After 100 steps of SGLD, the action population diffuses. The end result is shown in figure 4.4. It could be seen that `sgld_k` shifts towards the target distribution `sgld_init`. While some point cluster in the upper side of the diagram overlaps, the point clusters in the button seem to be not yet fully shifts to its target distribution.

It could be seen from 4.3 that the noise level satisfies equation 4.1. And secondly, for the point clusters in the upper side of the image, which could be assumed that the SGLD mixed to its target distribution, the following assumption is also satisfied:

$$ {}^{100}a_j^i \in B_\varepsilon(x_j^i) $$

When the means of the model and target distribution are close, the variance of sampled actions is expected to be small. If this statement is not observed for a sufficient enough number of epochs, then we could conclude that the noise level is too high.

4.3.2 Training dynamic

In all models that converge to the target distribution, the loss curves are similar to the shape of figure 4.5. Its worth to be analyzed to diagnose whether would converge or not.

Such loss curve decreases monotonically for the first half number of epochs but starts at zero loss. It starts at zero loss because the parameters of network layers are initialized in such a way that given random input, it gives outputs that are at the same energy level.

The maximum likelihood loss decreases the energy level at sampled data point, increases energy level at expert data point, After some iteration, the negative energy stop increasing.

This could be explained by inspecting the gradient norm of each layer. As the magnitude of the loss increases, so are the gradient norms. It means that it takes now less iteration

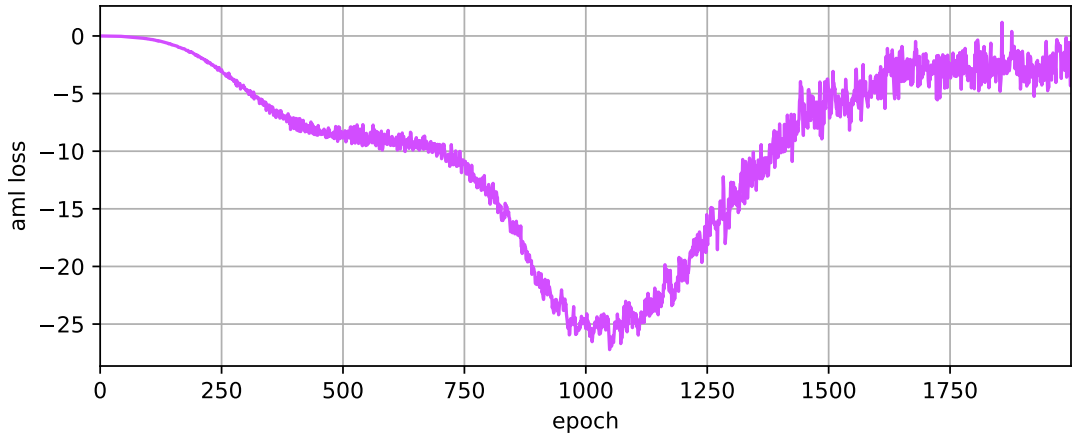


Figure 4.5: Maximum likelihood loss curve. A typical curve that increases its magnitude such that the network parameters get updated proportionally large, requiring less epoch to accumulate enough gradient to correct the distribution landscape represented by the energy based model. As the model gets close to converge, it requires more epoch again to accumulate gradient.

to correct the energy level at each negative sample location. As soon as it reaches some threshold, the random process of diffusion would be dominated by the gradient direction at the region between the starting location of the MCMC chain and the location of the mode.

4.3.3 Non-converging MCMC

We hope that there are some indicators for telling

1. whether an MCMC chain converges to its stationary distribution.
2. whether an energy based model converges to its target distribution.

Indicator for converging to sampling distribution There's no way to tell whether an MCMC chain converges to its target distribution, in general. But the samples acquired on transient distribution need to be ignored, otherwise, the estimation about the model distribution is biased.

All models in section 4.4 are trained with a fixed SGLD step parameter ${}^{\mathrm{SGLD}}K = 100$. It would be unavoidable for the first few epoch that the MCMC chain does not converge to the stationary distribution, but we could expect that the objective function would scale the magnitude such that the following relation holds if the number of epoch continues to increase, and the average MCMC chain length converges. [18]

$$ {}^{\mathrm{SGLD}}K\|\mathbf{W}\| \approx r_k $$

From figure 4.6 we can see the average MCMC chain length when a model is trained on `circle` dataset and ${}^{\mathrm{SGLD}}K = 100$. We could tell from this figure that the average MCMC chain length indicates the sampling distribution converges to the distribution of the dataset (the distribution represented by the energy based model).

We show figure 2.2 again here for a comparison of converging and non-converging MCMC chain.

Indicator for converging to target distribution However, the indicator r_k in figure 4.6 does not tell us whether the energy based model converges to its target distribution or it is merely over fitting the sampler such that the energy based model distribution, when sampled by this particular sampler, can generate a distribution that is close to the target distribution.

In other words, this energy based model is not *sampler invariant*. This problem could be demonstrated by figure 4.8 to 4.11.

We see that the sampling distribution with ${}^{\mathrm{SGLD}}K = 100$ converges to the target distribution. But if we continues to increase ${}^{\mathrm{SGLD}}K = 150$, the sampling distribution shifts.

It could be concluded that the underlying energy based model has a distribution that is different from the ground truth. However, because of the loss objective function, the energy based model learns to trick this particular sampler, such that when applying the sampling operator with a particular parameter, here ${}^{\mathrm{SGLD}}K$, the sampling distribution corresponds to the target distribution. This phenomenon is observed also in [18].

It was proposed to increase the ${}^{\mathrm{SGLD}}K$ to allow more involved mixing, but it was stated that the step length is 10 times the step length to ensure that the energy based model is not sensitive to the sampler parameter.

4.4 Sampled trajectories

4.4.1 path1

hyperpameter	value
lr	1×10^{-4}
z_dim	64
h_dim	8
reg_l1	0.0
reg_l2	1×10^{-5}
sgld_K	100
sgld_al	5×10^{-3}
sgld_dim	1×10^{6}
dataset_sigma	0.1
epoch	851

Table 4.2: Training parameter for path1 dataset

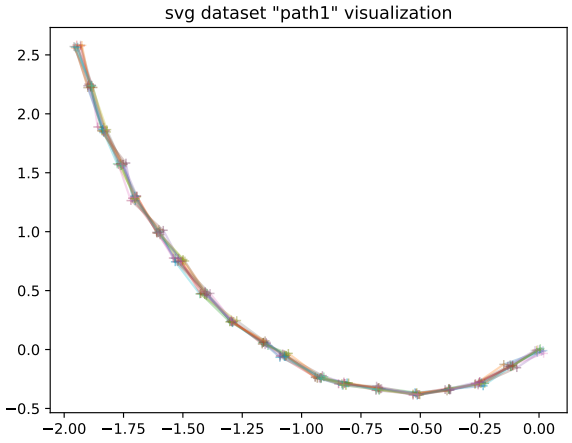


Figure 4.12: path1 dataset

path1 dataset acts as a reference dataset to tune model architecture, inspect training dynamic, etc. The state-action mapping is added with a Gaussian noise with variance dataset_sigma. Figure 4.14 to 4.18 shows trajectories starting from different position. The model learns that the action to be identical in the position orthogonal to the demonstrated trajectory.

4.4.2 circle

hyperpameter	value
lr	1×10^{-4}
z_dim	128
h_dim	10
reg_l1	0.0
reg_l2	1×10^{-5}
sgld_K	100
sgld_al	1×10^{-3}
sgld_dim	1×10^{6}
dataset_sigma	0.1
epoch	834

Table 4.3: Training parameter for `circle` dataset

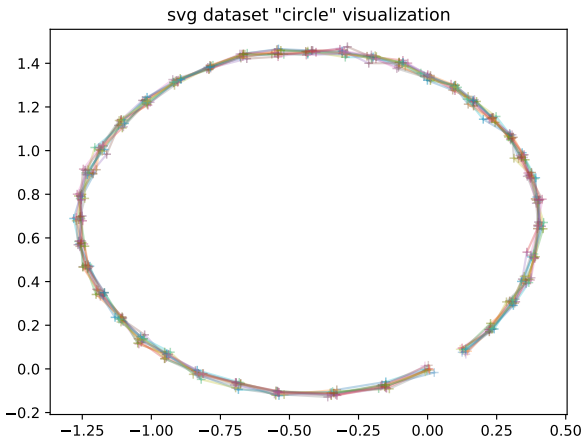


Figure 4.13: `circle` dataset

`circle` dataset is another dataset whose trajectories are generated by adding noise to a reference path. This dataset requires three times the epoch (3000 iteration) in a three layer MLP for training to generate trajectories that are similar to demonstrated trajectory. Using the proposed trajectory sampler, the trajectories stabilizes to the reference path inside the circle, and has region of convergence is roughly 1.5 the radius of the circle.

4.4.3 mouse1

hyperpameter	value
lr	1×10^{-3}
z_dim	128
h_dim	16
reg_l1	1×10^{-6}
reg_l2	0
sgld_K	256
sgld_al	1×10^{-3}
sgld_dim	1×10^{6}
epoch	1498

Table 4.4: Training parameter for mouse1 dataset

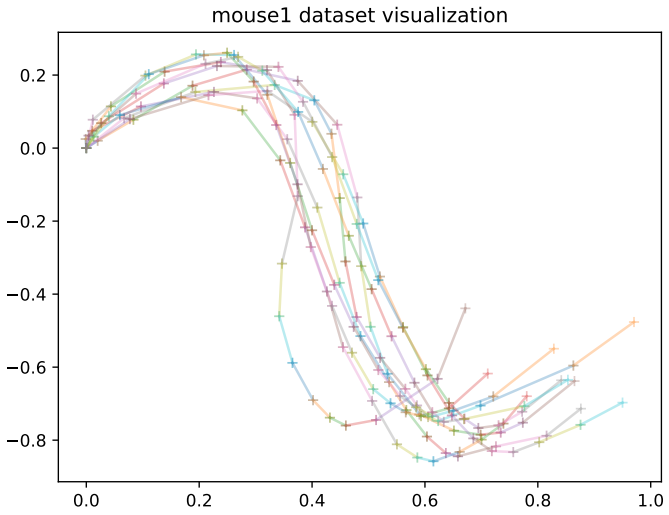


Figure 4.20: mouse1 dataset

Training an energy based model for mouse1 dataset requires tuning the SGLD step length. It is tested that with sgld_K equals to 256, the number of iteration required for MCMC chain to converge is around 1100, while for sgld_K equals to 100 the MCMC chain is still mixing. The curve one can refer to is figure 4.7.

This dataset exhibits large variance near the tail of the trajectory. The EXP sampler does not produces trajectories that reflect its multi-modality.

4.4.4 mouse2

hyperpameter	value
lr	1×10^{-3}
z_dim	128
h_dim	10
reg_l1	1×10^{-6}
reg_l2	0
sgld_K	100
sgld_al	5×10^{-4}
sgld_dim	1×10^{6}
epoch	1999

Table 4.5: Training parameter for mouse2 dataset

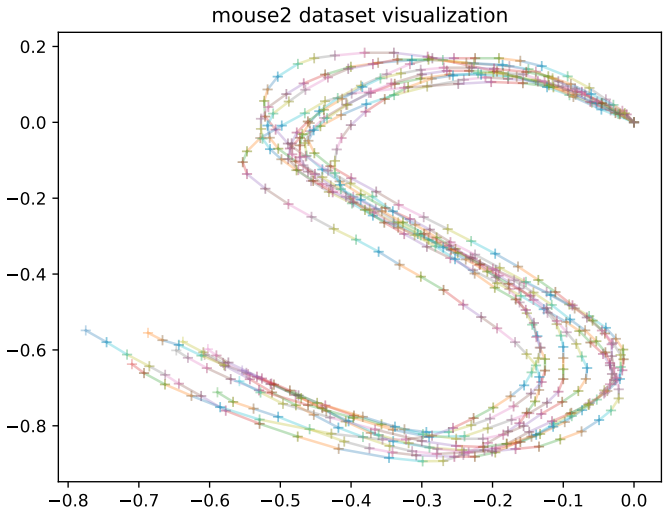


Figure 4.21: mouse2 dataset

mouse2 dataset demonstrates again that the energy based model tends to imitate the action that is not necessarily mimicking its closest demonstration in the orthogonal direction, at least measured by Euclidean distance. But it provides guidance on how we can further impose an additional constraint on the objective function, possibly be dataset dependent.

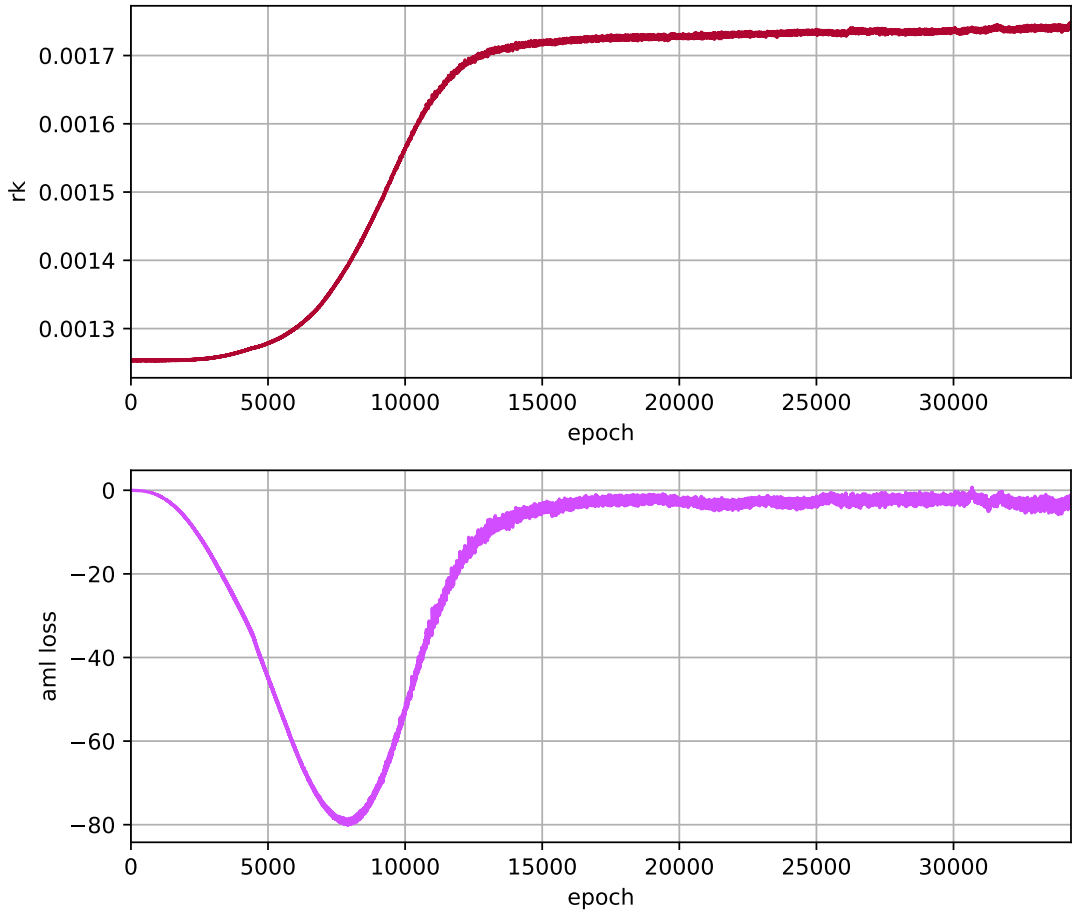


Figure 4.6: Average MCMC chain length on `circle` dataset with ${}^{\mathrm{SGLD}}K = 100$. It could be seen that the average MCMC chain length increases slightly, this indicates that the MCMC chain reaches its model before running up all the SGLD step. The slope of rk should be proportional to the noise level ${}^{\mathrm{SGLD}}\alpha$

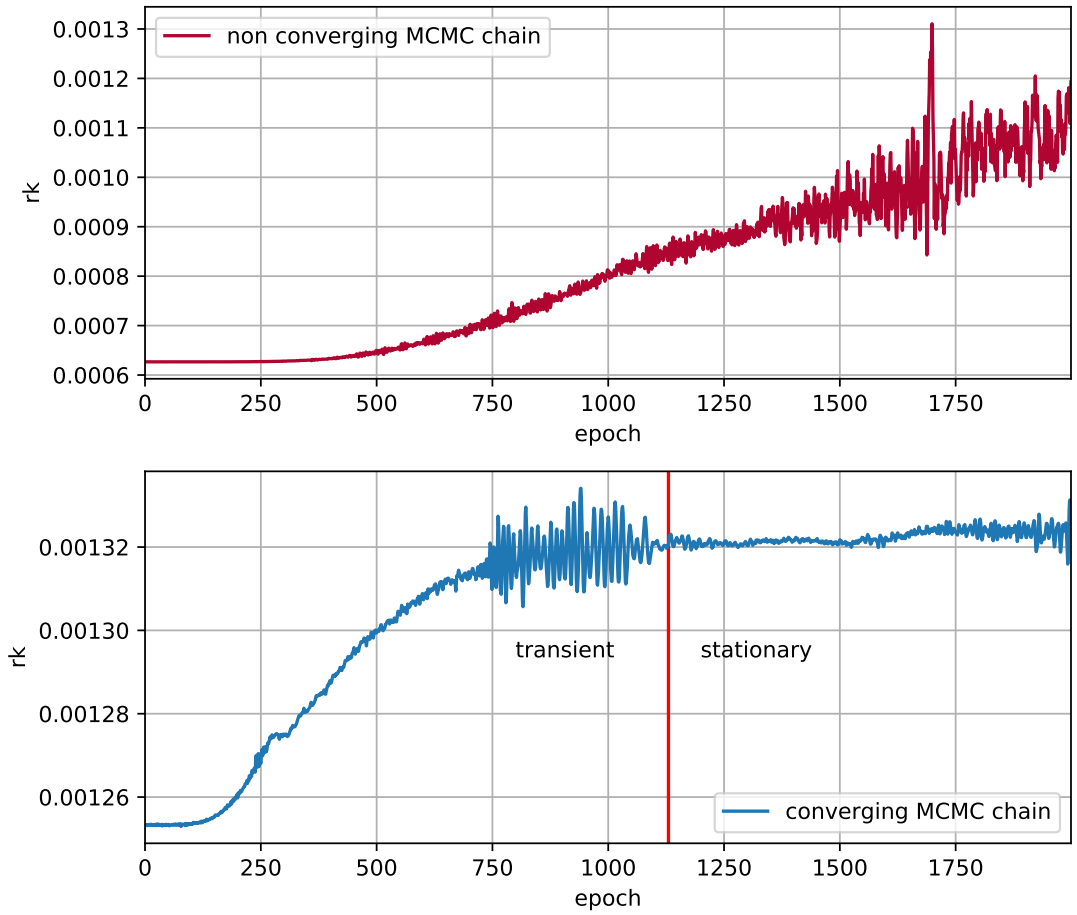


Figure 4.7: Average MCMC chain length R curve of converging and non-converging MCMC chain. Both models are trained on `mouse1` dataset. The model in the upper is using MCMC step length of 100. The model in the lower is using MCMC step length of 256. This comparison shows that MCMC step length could effect the rate of training the model, and is supported by inspecting the magnitude of gradient norm in each network layer. It could be seen clearly in the lower diagram that the MCMC chain converges to a value after a PT1 like a rising curve. It could be said that before epoch 1100, neither MCMC chain nor the distribution of energy based model converges. After this epoch, at least the sample distribution is stationary. Whether the MCMC converges, should be validated by running a large MCMC step length. Refer [18] for a more detailed investigation on this problem

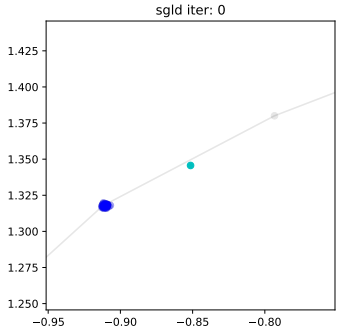


Figure 4.8: ${}^{\mathrm{SGLD}}K = 0$ on `circle` dataset. The blue particle indicates noise initialized MCMC chain.

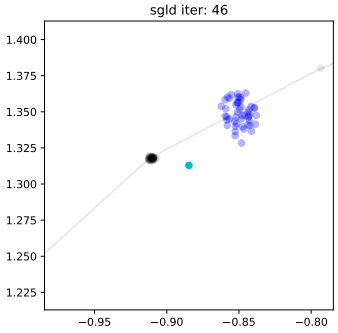


Figure 4.9: ${}^{\mathrm{SGLD}}K = 46$ on `circle` dataset. The blue particle moves from starting position towards its next state. The SGLD length is approximately proportional to the SGLD step.

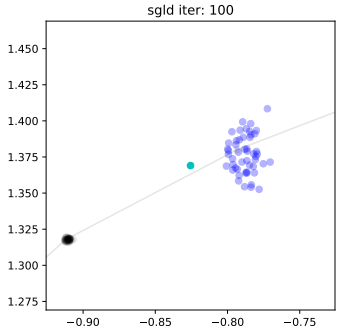


Figure 4.10: ${}^{\mathrm{SGLD}}K = 100$ on `circle` dataset. The particle reaches its target distribution exactly at 100 step.

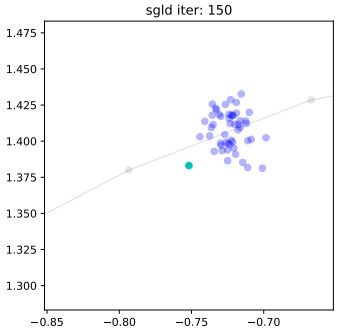


Figure 4.11: ${}^{\mathrm{SGLD}}K = 150$ on `circle` dataset. Continue running the MCMC chain, it could be shown that the particle deviates from its target distribution. This indicates that the MCMC chain doesn't converge to its stationary distribution. This phenomenon is also discussed in [17].

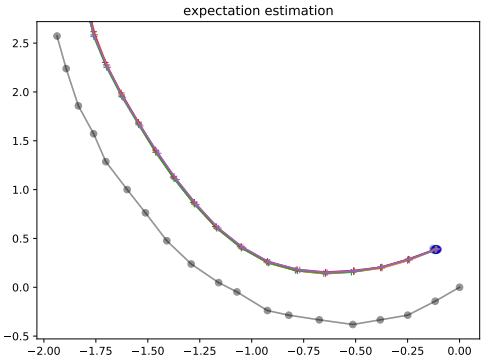


Figure 4.14: `path1` inner starting point

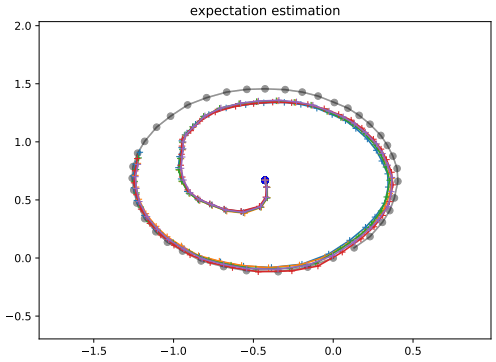


Figure 4.15: `circle` inner starting point

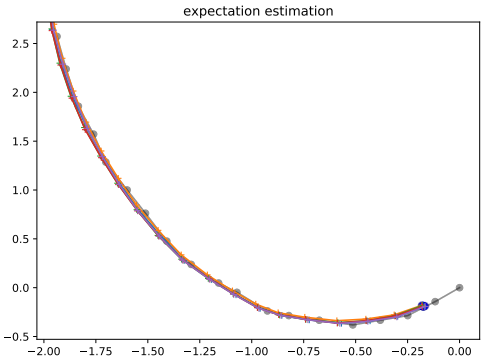


Figure 4.16: `path1` on the reference path

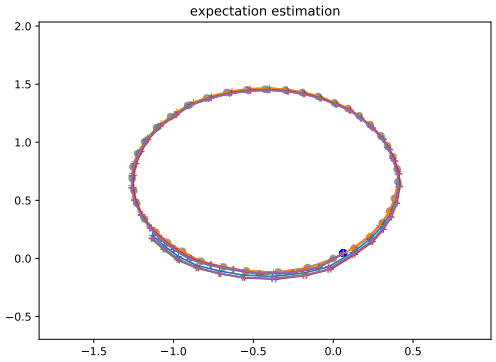


Figure 4.17: `circle` on the reference path

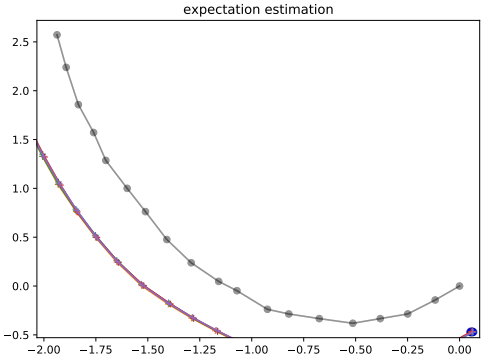


Figure 4.18: `path1` on the outer starting point

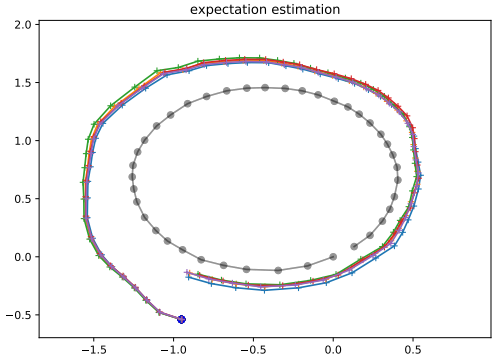


Figure 4.19: `circle` on the outer starting point

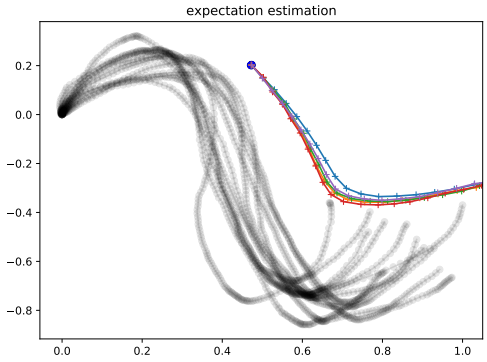


Figure 4.22: mouse1 inner starting point

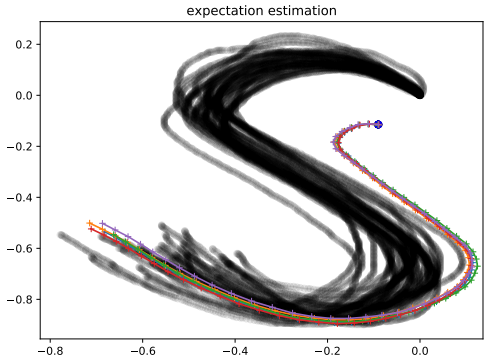


Figure 4.23: mouse2 inner starting point

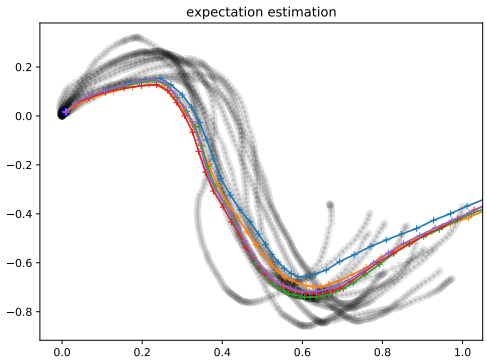


Figure 4.24: mouse1 on the reference path

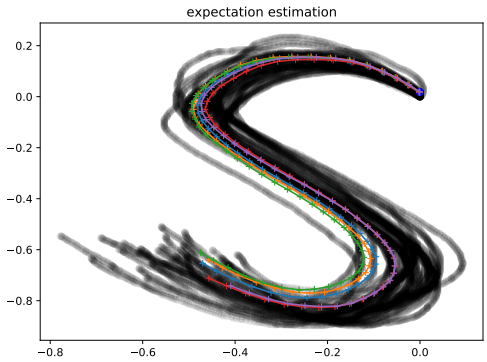


Figure 4.25: mouse2 on the reference path

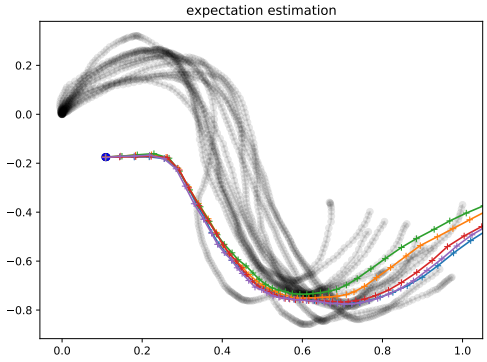


Figure 4.26: mouse1 on the outer starting point

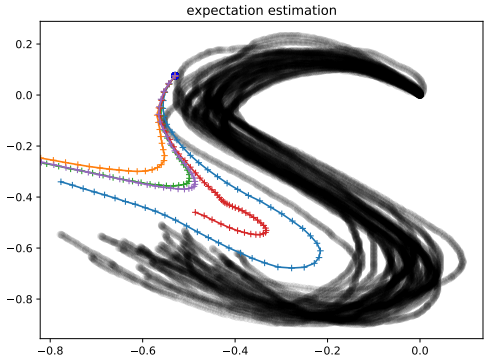


Figure 4.27: mouse2 on the outer starting point

5 Conclusion

This master thesis experiments with trajectory generation with energy based models.

An energy based model is proposed to learn the state action mapping given demonstrated trajectories in 2D space. An architecture that embeds state and action into a higher dimensional space is proposed. It is tested that applying spectral normalization makes the learning process requires far more epoch to update the network parameters. A sampler for generating trajectory on the energy based model is proposed. It is shown that the trajectory produced by the sampler has a smaller variance and more close to the demonstrated trajectory.

Recent methods for training energy based models with maximum likelihood estimation are reviewed. Metrics for diagnosing training convergence are applied and analyzed in the experiment. The training dynamic is discussed in the experiment, arguing that noise initialization together with average MCMC chain length could reflect the convergence of an MCMC chain.

The energy based model is evaluated on four datasets to demonstrate its ability to generate trajectory within a reasonable large neighborhood of the demonstrated trajectory.

Bibliography

[1] Steffen Hölldobler, Sibylle Möhle, and Anna Tigunova. “Lessons Learned from AlphaGo.” In: *YSIP*. 2017, pp. 92–101.

[2] Yang Yu. “Towards sample efficient reinforcement learning”. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2018, pp. 5739–5743.

[3] Igor Mordatch. *Concept Learning with Energy-Based Models*. 2018. arXiv: 1811.02486 [cs.AI].

[4] TL Griffiths and AL Yuille. *Technical introduction: A primer on probabilistic inference. UCLA. Department of Statistics Papers no. 2006010103. UCLA, Los Angeles, CA*. 2006.

[5] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York, 2001.

[6] Yann LeCun et al. “A tutorial on energy-based learning”. In: *Predicting structured data* 1.0 (2006).

[7] Yoshua Bengio, Yann LeCun, et al. “Scaling learning algorithms towards AI”. In: *Large-scale kernel machines* 34.5 (2007), pp. 1–41.

[8] Johannes N. Hendriks et al. *Deep Energy-Based NARX Models*. 2020. arXiv: 2012.04136 [cs.LG].

[9] Saeed Saremi et al. *Deep Energy Estimator Networks*. 2018. arXiv: 1805.08306 [stat.ML].

[10] Manoj Kumar and Neha Yadav. “Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: a survey”. In: *Computers & Mathematics with Applications* 62.10 (2011), pp. 3796–3811.

[11] Hugo Larochelle et al. “Learning algorithms for the classification restricted boltzmann machine”. In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 643–669.

[12] Takayuki Osogami. *Boltzmann machines and energy-based models*. 2019. arXiv: 1708.06008 [cs.NE].

[13] Yang Song and Diederik P. Kingma. *How to Train Your Energy-Based Models*. 2021. arXiv: 2101.03288 [cs.LG].

[14] Will Grathwohl et al. *Your Classifier is Secretly an Energy Based Model and You Should Treat it Like One*. 2020. arXiv: 1912.03263 [cs.LG].

[15] Fredrik K. Gustafsson et al. *How to Train Your Energy-Based Model for Regression*. 2020. arXiv: 2005.01698 [cs.CV].

[16] Yilun Du and Igor Mordatch. "Implicit generation and modeling with energy based models". In: (2019).

[17] Erik Nijkamp et al. *Learning Non-Convergent Non-Persistent Short-Run MCMC Toward Energy-Based Model*. 2019. arXiv: 1904.09770 [stat.ML].

[18] Erik Nijkamp et al. *On the Anatomy of MCMC-Based Maximum Likelihood Learning of Energy-Based Models*. 2019. arXiv: 1903.12370 [stat.ML].

[19] Will Grathwohl et al. *No MCMC for me: Amortized sampling for fast and stable training of energy-based models*. 2020. arXiv: 2010.04230 [cs.LG].

[20] Ruiqi Gao et al. *Learning Energy-Based Models by Diffusion Recovery Likelihood*. 2020. arXiv: 2012.08125 [cs.LG].

[21] Lantao Yu et al. *Training Deep Energy-Based Models with f-Divergence Minimization*. 2020. arXiv: 2003.03463 [cs.LG].

[22] Yilun Du et al. *Improved Contrastive Divergence Training of Energy Based Models*. 2020. arXiv: 2012.01316 [cs.LG].

[23] Aapo Hyvärinen and Peter Dayan. "Estimation of non-normalized statistical models by score matching." In: *Journal of Machine Learning Research* 6.4 (2005).

[24] Kevin Swersky et al. "On autoencoders and score matching for energy based models". In: *ICML*. 2011.

[25] Yang Song and Stefano Ermon. *Improved Techniques for Training Score-Based Generative Models*. 2020. arXiv: 2006.09011 [cs.LG].

[26] Yang Song et al. *Sliced Score Matching: A Scalable Approach to Density and Score Estimation*. 2019. arXiv: 1905.07088 [cs.LG].

[27] Charles Stein et al. "A bound for the error in the normal approximation to the distribution of a sum of dependent random variables". In: *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 2: Probability Theory*. The Regents of the University of California. 1972.

[28] Will Grathwohl et al. “Learning the stein discrepancy for training and evaluating energy-based models without sampling”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 3732–3747.

[29] Qiang Liu, Jason D. Lee, and Michael I. Jordan. *A Kernelized Stein Discrepancy for Goodness-of-fit Tests and Model Evaluation*. 2016. arXiv: 1602.03253 [stat.ML].

[30] Qiang Liu and Dilin Wang. *Learning Deep Energy Models: Contrastive Divergence vs. Amortized MLE*. 2017. arXiv: 1707.00797 [stat.ML].

[31] Michael Gutmann and Aapo Hyvärinen. “Noise-contrastive estimation: A new estimation principle for unnormalized statistical models”. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 297–304.

[32] Max Welling and Yee W Teh. “Bayesian learning via stochastic gradient Langevin dynamics”. In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. Citeseer. 2011, pp. 681–688.

[33] Ryuichiro Hataya, Hideki Nakayama, and Kazuki Yoshizoe. *Graph Energy-based Model for Substructure Preserving Molecular Design*. 2021. arXiv: 2102.04600 [physics.chem-ph].

[34] Dirk P Kroese, Thomas Taimre, and Zdravko I Botev. *Handbook of monte carlo methods*. Vol. 706. John Wiley & Sons, 2013.

[35] Nicolas Brosse, Alain Durmus, and Eric Moulines. *The promises and pitfalls of Stochastic Gradient Langevin Dynamics*. 2018. arXiv: 1811.10072 [stat.ML].

[36] Yang Song and Stefano Ermon. *Generative Modeling by Estimating Gradients of the Data Distribution*. 2020. arXiv: 1907.05600 [cs.LG].

[37] Takayuki Osa et al. “An Algorithmic Perspective on Imitation Learning”. In: *Foundations and Trends in Robotics* 7.1-2 (2018), pp. 1–179. ISSN: 1935-8261. DOI: 10.1561/2300000053. URL: http://dx.doi.org/10.1561/2300000053.

[38] Andrew Y Ng, Stuart J Russell, et al. “Algorithms for inverse reinforcement learning.” In: *Icml*. Vol. 1. 2000, p. 2.

[39] Brian D Ziebart et al. “Maximum entropy inverse reinforcement learning.” In: *Aaai*. Vol. 8. Chicago, IL, USA. 2008, pp. 1433–1438.

[40] J. Andrew (Drew) Bagnell. *An Invitation to Imitation*. Tech. rep. CMU-RI-TR-15-08. Pittsburgh, PA: Carnegie Mellon University, Mar. 2015.