UNIVERSITY OF CALIFORNIA
Los Angeles

Learning Energy-Based Prior Model
for Unsupervised Meta-Learning

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Statistics

by

Deqian Kong

2021

ABSTRACT OF THE THESIS

Learning Energy-Based Prior Model
for Unsupervised Meta-Learning

by

Deqian Kong
Master of Science in Statistics
University of California, Los Angeles, 2021
Professor Ying Nian Wu, Chair

In this thesis, we shall discuss the unsupervised meta-learning. Building a general-purpose AI demands an intelligent system capable of learning a broad range of knowledge with modest data and transferring the learned knowledge to the concrete case. Meta-learning is introduced to tackle this problem. Enabled by the common feature between meta-learning and unsupervised learning that they both learn a learning procedure that is more efficient and effective than learning from scratch, we propose the Symbolic Vector coupling Energy-Based Model (SVEBM) to implement unsupervised meta-learning by exploiting the structural difference between unsupervised and supervised meta-learning. From the probabilistic point of view, we illustrate these approaches as graphical models.

The thesis of Deqian Kong is approved.

Hongjing Lu

Tao Gao

Ying Nian Wu, Committee Chair

University of California, Los Angeles

2021

To my parents, grandparents and my sister

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Formost, I want to offer my special thanks to my advisors Dr. Ying Nian Wu and Dr. Song-Chun Zhu for their continuous support to my M.Sc. and Ph.D. research, for their patience, knowledge and enthusiasm.

I would like to offer thanks to Bo Pang for his patient guidance and insightful discussions about this thesis and many other projects, to Allen Kei who is always by my side and supports me, to Yixin Chen for his thoughtful ideas in computer vision, to Yifei Xiong whose optimistic mind has always encouraged me, and to Shi Feng for his vivid physical intuition, solid mathematical foundations and years of friendship.

I would also like to thank Laurie Leyden for her outstanding support in administrative work and the support and facilities offered by the Department of Statistics at UCLA.

CHAPTER 1

Meta-Learning

The state-of-the-art machine learning has seen tremendous success in broad range of areas, such as image recognition[HZR16], natural language processing[VSP17], reinforcement learning[SSS17] and so on. Take image recognition as an example. Supported by ImageNet[KSH12], the booming of data has established the route to data-hungry models: after training on thousands of labelled images for each class, the machine can 'beat' human in nearly every specific task. However, in the 'state-of-the-art' of human learning, a five-year-old child can quickly identify an image with only modest positive examples. In that sense, a computation-intensive system cannot even come close to achievements of a child, in sharp contrast to human cognition that can readily generalize reasonable concepts[Ten99].

Another dichotomy existing in human learning and current machine learning is that machine learning community favors an end-to-end schema for a single task from scratch, while human can perform multi-task learning at the same time and keep reusing the abstractions and concepts continuously[Fin18]. From the human learning perspective, the emergence of common sense from single task end-to-end learning with thousands of datapoints seems implausible.

These dichotomies demand an intelligent system capable of learning a broad range of knowledge with modest data. The meta-learning is introduced to tackle this problem. The core idea of meta-learning is treating each task as a training example instead of each datapoint, which is viewed to learn a general prior in the hierarchical Bayesian framework[Ten99].

While the large and diverse dataset will boost the performance, it doesn't acquire generalizability by nature. Therefore, meta-learning also emphasizes the knowledge transfer from the prior to the specific task.

In this chapter, we first define the meta-learning problem in Section 1.1 and then introduce the model-agnostic meta-learning in Section 1.2.1, then formulate it in a probabilistic view in Section 1.2.2 and illustrate it as a graphical model in Section 1.2.

1.1 Problem Statement

Meta-learning, or *learning how to learn*, mainly emphasize a learning procedure that learns the structures and concepts among previous seen tasks such that the model can quickly adapt to make generalizable inferences with limited amounts of new data. Meta-learning algorithm has wide range of applications including classification, regression, reinforcement learning et al. In this thesis, we only talk about the scenario of few-shot classification such that this problem can be sufficiently defined. A brief introduction of a task in few-shot classification is shown in Figure 1.1.

In the context of few-shot classification, an M-way S-shot classification task $\mathcal{T}$ consists of M classes, each of which has S support examples and labels $\{x_s, y_s\}_{s=1}^{S}$ and Q query examples and labels $\{x_q, y_q\}_{q=1}^{Q}$. For each task $\mathcal{T}$, it has total $(S+Q)M$ datapoints and we only use support examples to train a classifier and query examples to evaluate the learned classifier.

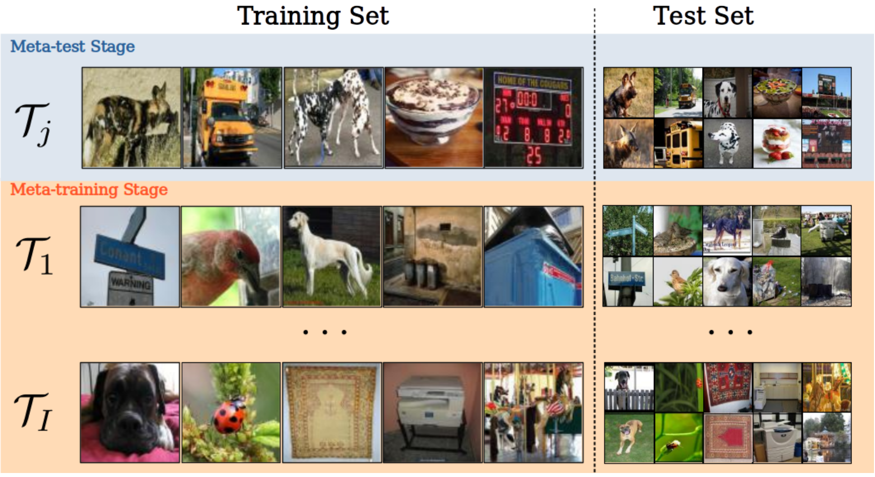


Figure 1.1: 5-way 1-shot 2-query classification task on MiniImagenet[RL16]. In each task, the training set (support set) has 5×1 images and the test set (query set) has 5×2 images. The meta-learning algorithm aims to accomplish any held-out task $\mathcal{T}_j$ in the meta-test stage by ingesting meta-training tasks $\{\mathcal{T}_i\}_{i=1}^{I}$ in the meta-training stage.

As shown in Figure 1.1, we assume a distribution over all tasks $p(\mathcal{T})$ that we want our model to be able to adapt to. In the supervised few-shot classification task, we first sample a set of

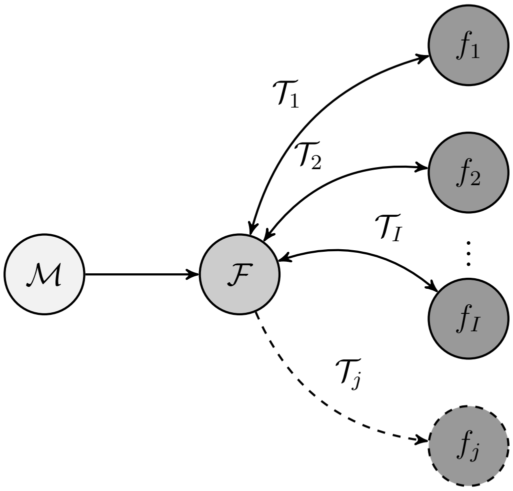


Figure 1.2: Graphical illustration of general meta-learning algorithm. A meta-learning algorithm $\mathcal{M}$ aims to find a learning procedure $\mathcal{F}$ such that in the meta-test stage (as dashed lines) it can accomplish a held-out task by producing a reasonable classifier f_j. In the meta-training stage, we use $\mathcal{F}$ to produce specific classifier f_i for any given meta-training task and in turn, fine-tune the learning procedure $\mathcal{F}$.

meta-training tasks $\{\mathcal{T}_i\}_{i=1}^{I} \sim p(\mathcal{T})$ with their corresponding datasets $\{\mathcal{D}_{\mathcal{T}_i}\}_{i=1}^{I}$, each of which can be further decomposed into training set (support set) and test set (query set) without overlapping $\mathcal{D}_{\mathcal{T}_i} = \{\mathcal{D}_{\mathcal{T}_i}^{\text{train}}, \mathcal{D}_{\mathcal{T}_i}^{\text{test}}\}$. To be concrete, $\mathcal{D}_{\mathcal{T}_i}$ has total $(S+Q)M$ datapoints.

In the meta-training stage, the meta-learning algorithm $\mathcal{M}(\cdot)$ takes as input a set of meta-training tasks $\{\mathcal{T}_i\}_{i=1}^{I}$, and produces a learning procedure $\mathcal{F}(\cdot)$. During the meta-training stage for each task, $\mathcal{F}(\cdot)$ is optimized on the training set $\mathcal{D}_{\mathcal{T}_i}^{\text{train}}$ to produce a classifier $f_i(\cdot)$ such that $f_i(\cdot)$ can perform well on $\mathcal{D}_{\mathcal{T}_i}^{\text{test}}$.

While in the meta-test stage, we first sample a set of meta-test tasks $\{\mathcal{T}_j\}_{j=1}^{J} \sim p(\mathcal{T})$ with their corresponding datasets $\{\mathcal{D}_{\mathcal{T}_j}\}_{j=1}^{J}$, which are held-out from the meta-training tasks $\{\mathcal{T}_i\}_{i=1}^{I}$. Then we use learned procedure $\mathcal{F}(\cdot)$ to produces $f_j(\cdot)$ on $\mathcal{D}_{\mathcal{T}_j}^{\text{train}}$ and perform well on $\mathcal{D}_{\mathcal{T}_j}^{\text{test}}$. In other words, the goal of $\mathcal{M}(\cdot)$ is to learn $\mathcal{F}(\cdot)$ such that when faced the held-out task $\{\mathcal{T}_j\}$, $\mathcal{F}(\cdot)$ can learn a classifier $f_j(\cdot)$ to accomplish $\mathcal{T}_j$.

Symbol	**Terminology**	**Examples in few-shot classification**
$\mathcal{T}$	task	The goal of a M-way S-shot classification task is learn a model on the support set $\mathcal{D}_{\mathcal{T}}^{\text{train}}$ to correctly predict the labels of query set $\mathcal{D}_{\mathcal{T}}^{\text{test}}$.
$p(\mathcal{T})$	task distribution	distributions of all tasks from which meta-training tasks and meta-test tasks are sampled
$\{\mathcal{T}_i\}_{i=1}^{I} \sim p(\mathcal{T})$	meta-training tasks	tasks sampled from $p(\mathcal{T})$ used for meta-training
$\{\mathcal{D}_{\mathcal{T}_i}\}_{i=1}^{I}$	meta-training set	datasets corresponding to the meta-training tasks, each of which consists of $(S+Q)M$ datapoints
$\{\mathcal{T}_j\}_{j=1}^{J} \sim p(\mathcal{T})$	meta-test tasks	tasks sampled from $p(\mathcal{T})$ used for meta-test, which are held out from the meta-training tasks $\{\mathcal{T}_i\}$
$\{\mathcal{D}_{\mathcal{T}_j}\}_{j=1}^{J}$	meta-test set	datasets corresponding to the meta-test tasks, each of which consists of $(S+Q)M$ datapoints
$\mathcal{D}_{\mathcal{T}}^{\text{train}}$	training set (support set)	training data for task $\mathcal{T}$ with $S \times M$ datapoints
$\mathcal{D}_{\mathcal{T}}^{\text{test}}$	test set (query set)	test data for task $\mathcal{T}$ with $Q \times M$ datapoints

Table 1.1: Summary of meta-learning terminology used in this thesis.

1.2 Learning to Learn with Gradients

After clearly defining the meta-learning problem in the few-shot classification scenario, we shall think about how to propose a general and model-agnostic algorithm solely powered by stochastic gradient descent and backpropagation, which could shed light on the intuitive design of the algorithm without instantiation of any architecture or model[FAL17].

1.2.1 Model Agnostic Meta-Learning Algorithm (MAML)

As discussed in Section 1.1, the goal of a meta-learning algorithm is to learn a learning procedure $\mathcal{F}$ such that it can easily adapt to a new held-out task. Here, we mainly focus on Neural Network type of model powered by stochastic gradient descent.

From the view of representation learning, $\mathcal{F}$ is supposed to be internal representations that make sense most across all the tasks sampled from $p(\mathcal{T})$ and $\mathcal{F}$ serves as a better initialization to learn a task-specific classifier f than a random start point from scratch. If the internal representations indeed make sense, when given a specific task, the model should be able to learn a good classifier within one or few gradient updates. Standing on this viewpoint, the learning procedure $\mathcal{F}$ is assumed to be sensitive to the loss function of the new task in a sense that a small local change of the parameters will induce large improvements with respect to the task loss. With these simple intuitions behind, we can formally define the algorithms as follows.

We consider a learning procedure $\mathcal{F}$ is instantiated by a Neural Network f_θ with parameter θ. f_θ is also known as meta-learner with meta-parameter θ. When adapting to a new task $\mathcal{T}_i$, meta-parameter θ becomes task-specific parameter ϕ_i according to the loss function $\mathcal{L}_\theta(\mathcal{D}_{\mathcal{T}_i}^{\text{train}})$ within one or few gradient updates. The loss function can be mean squared error (MSE) for regression or binary cross-entropy (BCE) loss for discrete classification. Suppose we only update gradient once and denote η_0 as step size,

$$\phi_i = \theta - \eta_0 \nabla_\theta \mathcal{L}(\mathcal{D}_{\mathcal{T}_i}^{\text{train}}, \theta) \tag{1.1}$$

The few-shot classification objective function aims to evaluate the task-specific parameter on the query set $\mathcal{D}^{\text{test}}_{\mathcal{T}_i}$. This objective function is also named as meta-objective function,

$$\sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}\left(\mathcal{D}^{\text{test}}_{\mathcal{T}_i}, \phi_i\right) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}\left(\mathcal{D}^{\text{test}}_{\mathcal{T}_i}, \theta - \eta_0 \nabla_\theta \mathcal{L}(\mathcal{D}^{\text{train}}_{\mathcal{T}_i}, \theta)\right) \tag{1.2}$$

Then the meta-parameter θ is updated by stochastic gradient descent with respect to θ. What's to mention is that the gradient with respect to θ requires additional gradient computation that can be accomplished by auto-differentiation in PyTorch. In conclusion, MAML is shown in Algorithm 1.

```
Algorithm 1: Supervised MAML
Input  : Total learning iterations $T$, initial meta-parameters $\theta_0$, learning rate for the inner
         loop $\eta_0$ to compute task-specific parameter $\phi$ and learning rate $\eta_1$ for outer loop to
         update $\theta$, a dataset that forms a distribution of task $p(\mathcal{T})$.
Output : Meta-parameter $\theta_T$
if Meta-training then
    for $t = 0 : T - 1$ do
        1. Sample meta-training tasks: $\{\mathcal{T}_i\}_{i=1}^{I} \sim p(\mathcal{T})$
        for $i = 1 : I$ do
            2. Sample dataset: $\mathcal{D}^{\text{train}}_{\mathcal{T}_i} \sim \mathcal{D}_{\mathcal{T}_i}$ and $\mathcal{D}^{\text{test}}_{\mathcal{T}_i} \sim \mathcal{D}_{\mathcal{T}_i} \setminus \mathcal{D}^{\text{train}}_{\mathcal{T}_i}$
            3. Compute task-specific parameter: $\phi_i = \theta_t - \eta_0 \nabla_\theta \mathcal{L}(\mathcal{D}^{\text{train}}_{\mathcal{T}_i}, \theta_t)$
        4. Update meta-parameter: $\theta_{t+1} = \theta_t + \eta_1 \frac{1}{I} \sum_{i=1}^{I} \mathcal{L}\left(\mathcal{D}^{\text{test}}_{\mathcal{T}_i}, \phi_i\right)$
if Meta-test then
    5. Sample meta-test tasks:
    $\{\mathcal{T}_j\}_{i=1}^{J} \sim p(\mathcal{T})$, each of which is held-out from meta-training tasks.
    for $j = 1 : J$ do
        6. Sample dataset: $\mathcal{D}^{\text{train}}_{\mathcal{T}_j} \sim \mathcal{D}_{\mathcal{T}_j}$ and $\mathcal{D}^{\text{test}}_{\mathcal{T}_j} \sim \mathcal{D}_{\mathcal{T}_i} \setminus \mathcal{D}^{\text{train}}_{\mathcal{T}_j}$
        7. Compute task-specific parameter: $\phi_j = \theta_T - \eta_0 \nabla_\theta \mathcal{L}(\mathcal{D}^{\text{train}}_{\mathcal{T}_j}, \theta_T)$
        8. Compute evaluation metric.
```

1.2.2 A Probabilistic View of MAML

In Section 1.2.1, MAML is fully detailed with simple but informative intuition. MAML learns a learning procedure $\mathcal{F}$ that can be further parameterized by meta-parameter θ. In meta-training stage, θ learns to optimize a specific task $\mathcal{T}_i$ as task-specific parameter ϕ_i with one or few gradient updates.

However, like most of black-box Neural Networks, MAML has great expressive power but the ability to explain uncertainty. Direct gradient updates in (1.1) are deterministic which is a slight violation in human sense. Given a learned learning procedure, human tends to produce a set of different solutions for one specific task. This set of different solutions form as a distribution over task-specific parameter that can be further used to model the uncertainty for each task in a statistical way. In the view of Bayes, finding task-specific parameter ϕ_i in (1.1) can be regarded as an inference problem and meta-parameter θ is viewed as a prior. In other words, we need to make inferences of joint distribution on $q(\theta, \phi_i)$ based on the evidence (training set). Each of the parameter can be inferred akin to hierarchical Bayesian framework as $q(\theta, \phi_i) = q(\theta)q(\phi_i|\theta)$, where θ serves as a prior to infer ϕ_i[FXL18].

One possible solution is using structured variational inference, where we recruit another inference network q_α parameterized by a Neural Network with parameter α to make inference over θ. To be more concrete, for a supervised few-shot classification task $\mathcal{T}_i$, we denote $\mathcal{D}_i^{\text{train}} = \left(\mathbf{x}_i^{\text{train}}, \mathbf{y}_i^{\text{train}}\right)$ and $\mathcal{D}_i^{\text{test}} = (\mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}})$, in which $\mathbf{x}_i^{\text{train}}, \mathbf{y}_i^{\text{train}}, \mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}}$ are a set of meta-training datapoints with $\mathbf{x}$ as examples and $\mathbf{y}$ as corresponding labels. With little abuse of concepts, we model the $q(\theta)$ as a posterior $q_\alpha(\theta|\mathbf{x}_i^{\text{train}}, \mathbf{y}_i^{\text{train}}, \mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}})$ and $q(\phi_i|\theta)$ as $q_\alpha(\phi_i|\theta, \mathbf{x}_i^{\text{train}}, \mathbf{y}_i^{\text{train}}, \mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}})$ so that $q(\theta, \phi_i)$ is $q(\theta, \phi_i|\mathbf{x}_i^{\text{train}}, \mathbf{y}_i^{\text{train}}, \mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}})$.

Recall that in (1.1), ϕ_i is obtained by one or few gradient updates with respect to θ on the training set in the essence that ϕ_i is obtained by optimizing,

$$\phi_i^\star = \max p(\phi_i|\theta, \mathbf{x}_i^{\text{train}}, \mathbf{y}_i^{\text{train}}) \tag{1.3}$$

which gives the probabilistic interpretation of the MAML. When performing empirical Bayes, the marginal likelihood function in the first line (1.4) is required to integrate the task-specific parameter

ϕ_i out, which is impractical. Instead, we use a point estimate to replace the integration. In this way, the marginal likelihood is approximated by a posterior $p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{train}}, \mathbf{y}_i^{\text{train}}, \mathbf{x}_i^{\text{test}}, \theta)$ in (1.4) with respect to θ which uses the maximum a posterior (MAP) estimate of ϕ_i.

$$\begin{aligned} p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{train}}, \mathbf{y}_i^{\text{train}}, \mathbf{x}_i^{\text{test}}, \theta) &= \int p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{test}}, \phi_i) p(\phi_i|\mathbf{x}_i^{\text{train}}, \mathbf{y}_i^{\text{train}}, \theta) d\phi_i \\ &\approx p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{test}}, \phi_i^\star) \end{aligned} \tag{1.4}$$

Thus, in the hierarchical Bayes framework, we only perform variational inference for θ only and make MAP inference on ϕ_i. The evidence lower-bound (ELBO) for the log-likelihood $p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{train}}, \mathbf{y}_i^{\text{train}}, \mathbf{x}_i^{\text{test}})$ is shown as

$$\begin{aligned} \text{ELBO} &= \mathbb{E}_{q_\alpha(\theta|\mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}})} \left[p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{train}}, \mathbf{y}_i^{\text{train}}, \mathbf{x}_i^{\text{test}}, \theta) \right] - \mathbb{D}_{\text{KL}} \left(p(\theta) \| q_\alpha(\theta|\mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}}) \right) \\ &\approx \mathbb{E}_{q_\alpha(\theta|\mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}})} \left[p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{test}}, \phi_i^\star) \right] - \mathbb{D}_{\text{KL}} \left(q_\alpha(\theta|\mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}}) \| p(\theta) \right) \end{aligned} \tag{1.5}$$

where $\mathbb{D}_{\text{KL}}$ denotes the Kullback-Leibler divergence.

The prior distribution can be chosen to be Gaussian with learned mean μ_θ and diagonal covariance σ_θ^2. The posterior sampled from the inference network is given by

$$q_\alpha(\theta|\mathbf{x}_i^{\text{test}}, \mathbf{y}_i^{\text{test}}) = N(\mu_\theta + \eta \nabla_{\mu_\theta} \log p(\mathbf{y}_i^{\text{test}}|\mathbf{x}_i^{\text{test}}, \mu_\theta), v^2) \tag{1.6}$$

The detailed algorithm is shown in Algorithm 2. The comparison between vanilla MAML and probabilistic MAML is shown in Figure 1.3.

Here, we only give a basic introduction of the probabilistic MAML, which has been extended in the current literature. However, whichever form of probabilistic MAML built upon hierarchical Bayes would arguably give principled but intricate inference procedure.

Since this thesis is more interested in unsupervised meta-learning, while variational inference plays a crucial role in that scenario, this probabilistic MAML type of idea can be a great starting point. General illustration of unsupervised MAML is shown in Figure 1.4[HLF18].

Standing on the view of empirical Bayes, MAML further emphasize the prior art that treating meta-parameter as a prior. Akin to this, we build an energy-based prior model to perform unsupervised meta-learning in Chapter 3.

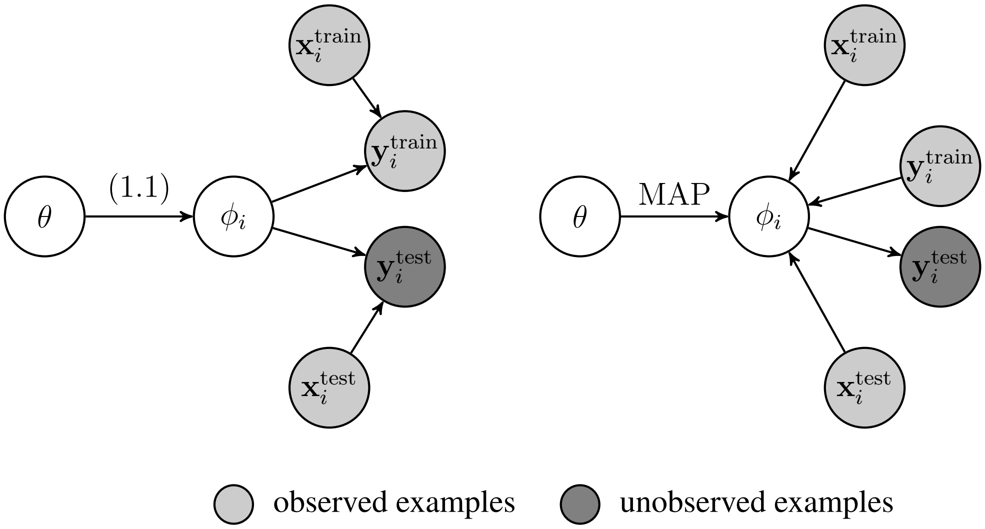

$\mathbf{x}_i^{\text{train}}$
$\mathbf{y}_i^{\text{train}}$
θ
(1.1)
ϕ_i
$\mathbf{y}_i^{\text{test}}$
$\mathbf{x}_i^{\text{test}}$
$\mathbf{x}_i^{\text{train}}$
$\mathbf{y}_i^{\text{train}}$
θ
MAP
ϕ_i
$\mathbf{y}_i^{\text{test}}$
$\mathbf{x}_i^{\text{test}}$
observed examples
unobserved examples

Figure 1.3: Graphical illustration of vanilla MAML in Algorithm 1 (left) and probabilistic MAML (right) in the supervised few-shot classification scenario. In vanilla MAML, ϕ_i is obtained by additional gradient updates with respect to θ, which is independent of $\mathbf{x}^{\text{train}}$ and $\mathbf{x}^{\text{test}}$. While in probabilistic MAML, ϕ_i is inferred by $p(\phi_i|\mathbf{x}_i^{\text{train}}, \mathbf{y}_i^{\text{train}}, \theta)$.

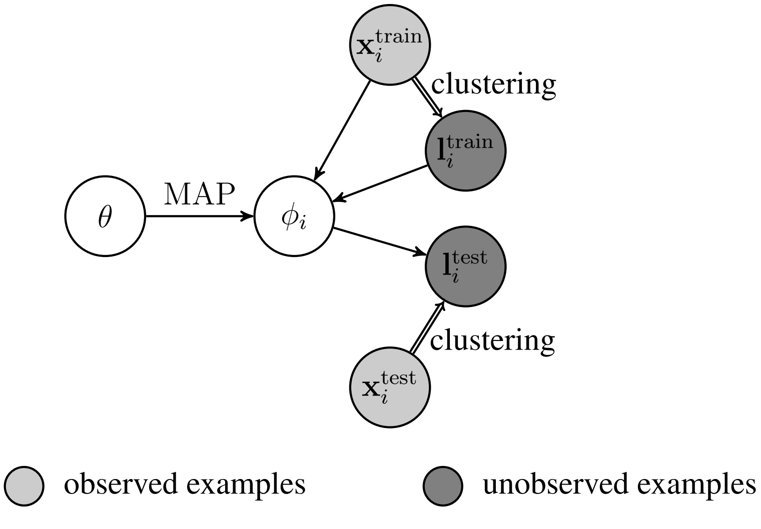

$\mathbf{x}_i^{\text{train}}$
clustering
$\mathbf{l}_i^{\text{train}}$
θ
MAP
ϕ_i
$\mathbf{l}_i^{\text{test}}$
clustering
$\mathbf{x}_i^{\text{test}}$
observed examples
unobserved examples

Figure 1.4: Graphical illustration of unsupervised MAML[HLF18]. During training, we first need to infer the pseudo-label $(\mathbf{l}_i^{\text{train}}, \mathbf{l}_i^{\text{test}})$ by K-means clustering, then generate the corresponding few-shot classification tasks and finally perform supervised MAML as before.

Algorithm 2: Supervised MAML via variational inference

```
Input  : Total learning iterations $T$, initial learnable parameters $\Theta = (\mu_\theta, \sigma_\theta^2, v^2, \gamma_0, \gamma_1)$,
         step size $(\eta_0, \eta_1)$, a dataset that forms a distribution of task $p(\mathcal{T})$.
Output : $\Theta_T$
if Meta-training then
    for $t = 0 : T - 1$ do
        1. Sample meta-training tasks: $\{\mathcal{T}_i\}_{i=1}^{I} \sim p(\mathcal{T})$
        for $i = 1 : I$ do
            2. Sample dataset: $\mathcal{D}_{\mathcal{T}_i}^{\text{train}} \sim \mathcal{D}_{\mathcal{T}_i}$ and $\mathcal{D}_{\mathcal{T}_i}^{\text{test}} \sim \mathcal{D}_{\mathcal{T}_i} \setminus \mathcal{D}_{\mathcal{T}_i}^{\text{train}}$
            3. Posterior sampling: $\theta \sim N(\mu_\theta - \gamma_0 \nabla_{\mu_\theta} \mathcal{L}(\mathcal{D}_i^{\text{test}}, \mu_\theta), v^2)$
            4. MAP of task-specific parameter: $\phi_i = \theta - \eta_0 \nabla_\theta \mathcal{L}(\mathcal{D}_{\mathcal{T}_i}^{\text{train}}, \theta)$
        5. Define prior: $p(\theta|\mathcal{D}^{\text{train}}) = N(\mu_\theta - \gamma_1 \nabla_{\mu_\theta} \mathcal{L}(\mathcal{D}_i^{\text{train}}, \mu_\theta), \sigma_\theta^2)$
        6. Update meta-parameter: $\Theta_{t+1} = \Theta_t + \eta_1 \nabla_\Theta \text{ELBO}$ using (1.5)
if Meta-test then
    7. Sample meta-test tasks:
    $\{\mathcal{T}_j\}_{i=1}^{J} \sim p(\mathcal{T})$, each of which is held-out from meta-training tasks.
    for $j = 1 : J$ do
        8. Sample dataset: $\mathcal{D}_{\mathcal{T}_j}^{\text{train}} \sim \mathcal{D}_{\mathcal{T}_j}$ and $\mathcal{D}_{\mathcal{T}_j}^{\text{test}} \sim \mathcal{D}_{\mathcal{T}_i} \setminus \mathcal{D}_{\mathcal{T}_j}^{\text{train}}$
        9. Sample prior: $\theta \sim p(\theta|\mathcal{D}^{\text{train}})$
        10. Compute task-specific parameter: $\phi_j = \theta - \eta_0 \nabla_\theta \mathcal{L}(\mathcal{D}_{\mathcal{T}_j}^{\text{train}}, \theta)$
```

CHAPTER 2

Latent Space Energy-Based Model

The main purpose of statistical modeling and machine learning is to encode dependencies between random variables. By capturing those dependencies, a model can be used to answer questions about the values of unknown variables using Bayesian framework. Energy-Based Model (EBM) captures dependencies between variables by associating a scalar energy to each configuration of the variables, where observed examples are assigned with low energy.

[LCH06] has defined the EBM in a non-probabilistic way such that it forms as a factor graph with much more flexibility than probabilistic approaches by circumventing the computation of intractable partition function. However, as to be shown in this chapter, the partition function can be used as a strength, which forms a principled way to model high-dimensional data and the learning process is directly accomplished by Maximum Likelihood Estimate (MLE). Both theoretical frameworks of EBM provide a common ground for many learning models, including traditional discriminative and generative approaches[XZL20], as well as conditional random fields[ZWM98].

Probabilistic models must be properly normalized. A powerful and efficient way is to use Markov Chain Monte Carlo (MCMC) to approximate the partition function. Powered by gradient-based method, we shall adapt Langevin dynamics as short-run MCMC[NPH20]. To overcome the computational consumption of MCMC, EBM is built in the low-dimensional latent space.

In this chapter, we shall build a probabilistic EBM in the latent space by treating it as a prior in Section 2.1[PHN20]. The intractable partition function is approximated by short-run MCMC. Then we further couple the latent variable with symbolic vector to perform classification in Section 2.2[PW21]. Both of the model are illustrated as graphical models.

2.1 Learning Latent Space Energy-Based Prior Model

Recently, deep generative models has flourished in the context of unsupervised learning and data generation. One of the simple but must-mentioned is the generator model, which learns to map the low-dimensional latent vector to the high-dimensional data space via a top-down network. The generator model is originated from the concepts of variational auto-encoder (VAE)[KW13] and generative adversarial network (GAN)[GPM20], in which the generator model is learned jointly with complementary models as the inference models in VAE and the discriminator models in GAN. In a more recent literature, the generator network can also be learned by MLE without resorting to a complementary model[PHN20], where the inference is guided by the MCMC such as Langevin dynamics[UO30].

2.1.1 Model: EBM as a prior

Let $x \in \mathbb{R}^D$ be an observed example and $z \in \mathbb{R}^d$ be the latent variable, where $D \gg d$. The joint distribution of (x, z) is

$$p_\theta(x, z) = p_\alpha(z) p_\beta(x|z) \tag{2.1}$$

where $p_\alpha(z)$ is the prior model with parameter α, $p_\beta(x|z)$ is the top-down generator model with parameters β, and $\theta = (\alpha, \beta)$.

The prior model $p_\alpha(z)$ is an EBM,

$$p_\alpha(z) = \frac{1}{Z(\alpha)} \exp\left(f_\alpha(z)\right) p_0(z) \tag{2.2}$$

where $p_0(z)$ is the reference distribution, assumed to be isotropic Gaussian. $f_\alpha(z)$ is the negative energy, which can be parameterized by a small multi-layer perceptron with the parameter α. $Z(\alpha)$ is known as the normalizing constant or the partition function that is computed by integrating z out.

$$Z(\alpha) = \int \exp\left(f_\alpha(z)\right) p_0(z) dz = \mathbb{E}_{p_0(z)}\left[\exp\left(f_\alpha(z)\right)\right] \tag{2.3}$$

The generator model is a top-down model with parameters β. For generation of images, it's

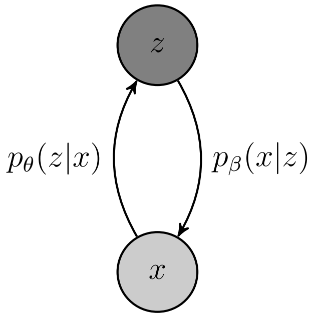


Figure 2.1: Graphical illustration of latent space EBM prior model. In the latent space, z follows a prior distribution $p_\alpha(z)$ in the form of EBM. Given z, the example x can be obtained from a generator model with parameters β. While the latent variable z is inferred from the posterior distribution $p_\theta(z|x)$ by Langevin dynamics.

always parameterized by a series of deconvolutional layers.

$$x = g_\beta(z) + \epsilon \tag{2.4}$$

where $\epsilon \sim N(0, \sigma^2 I_D)$ is random noise with assumed variance σ^2. Then $x \sim N(g_\beta(z), \sigma^2 I_D)$. Different from vanilla generator in VAE mapping the uni-modal prior $p_0(z)$ to be close to multi-modal data space, the generator $g_\beta(z)$ here refines $p_0(z)$ based on the observed examples so that it maps the more expressive $p_\alpha(z)$ to be closer to the data space.

The marginal distribution $p_\theta(x)$ used for MLE and the posterior distribution $p_\theta(z|x)$ for inference can be defined accordingly.

$$p_\theta(x) = \int p_\theta(x, z)dz = \int p_\alpha(z)p_\beta(x|z)dz \tag{2.5}$$

$$p_\theta(z|x) = \frac{p_\theta(x, z)}{p_\theta(x)} = \frac{p_\alpha(z)p_\beta(x|z)}{p_\theta(x)} \tag{2.6}$$

2.1.2 Unsupervised Learning via Short-run MCMC

Given observed training examples $\{x_i\}_{i=1}^n$, the log-likelihood function is

$$L(\theta) = \sum_{i=1}^{n} \log p_\theta(x_i) \tag{2.7}$$

For an observed example x, the learning gradient is

$$\begin{aligned} \nabla_\theta \log p_\theta(x) &= \mathbb{E}_{p_\theta(z|x)} \left[\nabla_\theta \log p_\theta(x, z)\right] \\ &= \mathbb{E}_{p_\theta(z|x)} \left[\nabla_\theta \left(\log p_\alpha(z) + \log p_\beta(x|z)\right)\right] \end{aligned} \tag{2.8}$$

which is a consequence of the simple identity $\mathbb{E}_{p_\theta(x)} \left[\nabla_\theta \log p_\theta(x)\right] = 0$. The proof is shown as follows,

$$\begin{aligned} \mathbb{E}_{p_\theta(z|x)} \left[\nabla_\theta \log p_\theta(x, z)\right] &= \mathbb{E}_{p_\theta(z|x)} \left[\nabla_\theta \log p_\theta(x) + \nabla_\theta \log p_\theta(z|x)\right] \\ &= \mathbb{E}_{p_\theta(z|x)} \left[\nabla_\theta \log p_\theta(x) + 0\right] \\ &= \nabla_\theta \log p_\theta(x) \end{aligned} \tag{2.9}$$

Specifically, for the prior model, $\nabla_\alpha \log p_\alpha(z) = \nabla_\alpha f_\alpha(z) - \mathbb{E}_{p_\alpha(z)} \left[\nabla_\alpha f_\alpha(z)\right]$. The learning gradient for an example x is

$$\delta_\alpha(x) = \nabla_\alpha \log p_\theta(x) = \mathbb{E}_{p_\theta(z|x)} \left[\nabla_\alpha f_\alpha(z)\right] - \mathbb{E}_{p_\alpha(z)} \left[\nabla_\alpha f_\alpha(z)\right] \tag{2.10}$$

which shows the nature of the empirical Bayes. The parameter α for the EBM is updated by comparing the difference between z inferred from the empirical observation x and z sampled from the current prior $p_\alpha(z)$.

As for the generator model,

$$\delta_\beta(x) = \nabla_\beta \log p_\theta(x) = \mathbb{E}_{p_\theta(z|x)} \left[\nabla_\beta \log p_\beta(x|z)\right] \tag{2.11}$$

which is also named as reconstruction error in practice.

The computation of above two gradient updates in (2.10) and (2.11) involves sampling the prior distribution $p_\alpha(z)$ and the posterior distribution $p_\theta(z|x)$. Here, we use Langevin dynamics to perform the sampling procedure. For a target distribution $\pi(z)$, the dynamic iterates

$$z_{k+1} = z_k + s\nabla_z \log \pi(z_k) + \sqrt{2s}\epsilon_k \tag{2.12}$$

where k denotes the time step of the sampling procedure, s is the small step size, and $\epsilon_k \sim N(0, I_d)$ is white noise as its analogy to Brownian motion.

Convergence of Langevin dynamics to the target distribution requires infinite steps with infinitesimal step size. Hence, we propose to use short-run MCMC for approximation.

The short-run MCMC is initialized from the fixed initial distribution and only run a fixed number of K steps in each sampling trial. In that sense, we only sample from an approximate of target distribution as $\tilde{\pi}(z)$. The short-run MCMC is defined as

$$z_0 \sim p_0(z), z_{k+1} = z_k + s\nabla_z \log \pi(z_k) + \sqrt{2s}\epsilon_k, k = 1, \ldots, K \tag{2.13}$$

where $z_K \sim \tilde{\pi}(z)$.

The full algorithm with unsupervised scheme is shown in Algorithm 3.

```
Algorithm 3: Unsupervised learning latent space EBM prior model via short-run MCMC
  Input  : Total learning iterations $T$, initial parameters $\theta_0 = (\alpha_0, \beta_0)$ for the EBM and the
           generator model, learning rate $(eta_\alpha, \eta_\beta)$, observed training examples $\{x_i\}_{i=1}^n$,
           batch size $m$, number of prior sampling $K_\alpha$ and posterior sampling $K_\theta$, step size
           in prior sampling $s_\alpha$ and posterior sampling $s_\theta$.
  Output : $\theta_T = (\alpha_T, \beta_T)$
  for $t = 0 : T - 1$ do
      1. Mini-batch: Sample observed examples $\{x_i\}_{i=1}^m$.
      2. Prior sampling: For each $x_i$, sample $z_i^- \sim \tilde{p}_{\alpha_t}(z)$ using (2.13), where
         $\pi(z) = p_{\alpha_t}(z)$, $s = s_\alpha$ and $K = K_\alpha$.
      3. Posterior sampling: For each $x_i$, sample $z_i^+ \sim \tilde{p}_{\theta_t}(z|x_i)$ using (2.13), where
         $\pi(z) = p_{\theta_t}(z|x_i)$, $s = s_\theta$ and $K = K_\theta$.
      4. Learning prior model: $\alpha_{t+1} = \alpha_t + \eta_\alpha \frac{1}{m} \sum_{i=1}^m [\nabla_\alpha f_{\alpha_t}(z_i^+) - \nabla_\alpha f_{\alpha_t}(z_i^-)]$
      5. Learning generator model: $\beta_{t+1} = \beta_t + \eta_\beta \frac{1}{m} \sum_{i=1}^m \log p_{\beta_t}(x_i|z_i^+)$
```

2.2 Coupling Symbolic Vectors For Classification

2.2.1 Model: symbolic-vector coupling EBM (SVEBM)

Let $x \in \mathbb{R}^D$ be an observed example, $z \in \mathbb{R}^d$ be the dense latent vector and $y \in \{0, 1\}^K$ be the symbolic one-hot vector indicating its belonging in total K categories. With the assumption that given z, y and x are independent, the joint distribution of (x, y, z) is

$$p_\theta(y, z, x) = p_\alpha(y, z) p_\beta(x|z) \tag{2.14}$$

where $p_\alpha(y, z)$ is the EBM prior model as mentioned in Section 2.1 with parameters α. $p_\beta(x|z)$ is the top-down generator model with parameters β and $\theta = (\alpha, \beta)$.

The prior model is defined as

$$p_\alpha(y, z) = \frac{1}{Z_\alpha} \exp\left(\langle y, f_\alpha(z)\rangle\right) p_0(z) \tag{2.15}$$

where similar to (2.2), $p_0(z)$ is the reference distribution as isotropic Gaussian, $f_\alpha(z)$ is a small multi-layer perceptron and Z_α is the normalizing constant or partition function.

The negative energy term $\langle y, f_\alpha(z)\rangle$ couples the dense vector z and symbolic vector y. The inference of symbolic vector y can be achieved from latent vector z using a softmax classifier.

$$p_\alpha(y|z) \propto \exp\left(\langle y, f_\alpha(z)\rangle\right) \tag{2.16}$$

Therefore, $f_\alpha(z)$ maps a latent vector in $\mathbb{R}^d$ to logit scores in $\mathbb{R}^K$.

The marginal distribution of latent variable z is computed by summation over y,

$$p_\alpha(z) = \frac{1}{Z_\alpha} \exp\left(F_\alpha(z)\right) p_0(z) \tag{2.17}$$

where $F_\alpha(z)$ denotes marginal energy as the form of log-sum-exponential.

$$F_\alpha(z) = \log \sum_y \exp\left(\langle y, f_\alpha(z)\rangle\right) \tag{2.18}$$

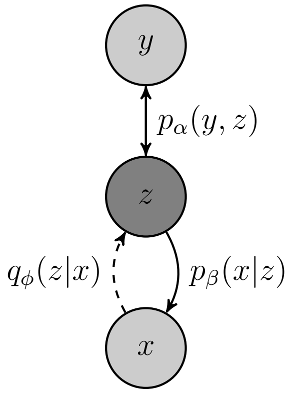


Figure 2.2: Graphical illustration of SVEBM. x is observed example. y is a symbolic one-hot vector and z is a dense vector. They are coupled through an EBM, $p_\alpha(y, z)$ in the latent space and y can be sufficiently inferred by z. The example x can be reconstructed from the generator model $p_\beta(x|z)$. The intractable posterior $p_\theta(z|x)$ is approximated by an inference network $q_\phi(z|x)$. Dashed lines denote variational inference.

The generator model is a top-down model with parameters β as Section 2.1. For image modeling,

$$x = g_\beta(z) + \epsilon \tag{2.19}$$

where $\epsilon \sim N(0, \sigma^2 I_D)$ is random noise with assumed variance σ^2. Then $x \sim N(g_\beta(z), \sigma^2 I_D)$.

The posterior distribution $p_\theta(z|x)$ for inference can be defined accordingly.

$$p_\theta(z|x) = \frac{p_\theta(x, z)}{p_\theta(x)} = \frac{p_\alpha(z)p_\beta(x|z)}{p_\theta(x)} \tag{2.20}$$

Here, instead of directly sampling the posterior $p_\theta(z|x)$ accomplished by short-run MCMC in Section 2.1, we tend to recruit another inference network $q_\phi(z|x)$ to approximate the true posterior $p_\theta(z|x)$ as VAE, and this inference model is jointly learned with the prior model and the generator model shown in Section 2.2.2.

2.2.2 Semi-supervised Learning via Variational Inference

In Section 2.1, we aim to use short-run MCMC to sample both the prior and the posterior. Due to the low-dimensionality in the latent space, short-run MCMC is affordable and mixes well. However, its additional back-propagation through whole generator model is undesirable. Hence, we adopt the same idea as VAE that we recruit another inference network $q_\phi(z|x)$ to approximate the true posterior $p_\theta(z|x)$. Following VAE, we tend to learn the inference model $q_\phi(z|x)$, the generator model $p_\beta(x|z)$ jointly.

For unlabelled example x, we have the unsupervised learning scheme and the log-likelihood $p_\theta(x)$ is lower bounded by evidence lower bound (ELBO).

$$\begin{aligned} \text{ELBO}(\theta, \phi) &= \log p_\theta(x) - \mathbb{D}_{\text{KL}}\left(q_\phi(z|x) \| p_\theta(z|x)\right) \\ &= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\beta(x|z)\right] - \mathbb{D}_{\text{KL}}\left(q_\phi(z|x) \| p_\alpha(z)\right) \end{aligned} \tag{2.21}$$

where $\mathbb{D}_{\text{KL}}$ denotes the Kullback-Leibler divergence.

For the prior model, the learning gradient for an example x is

$$\delta_\alpha = \nabla_\alpha \text{ELBO} = \mathbb{E}_{q_\phi(z|x)}\left[\nabla_\alpha F_\alpha(z)\right] - \mathbb{E}_{p_\alpha(z)}\left[\nabla_\alpha F_\alpha(z)\right] \tag{2.22}$$

where $\mathbb{E}_{q_\phi(z|x)}$ is approximated by samples from the inference network, while $\mathbb{E}_{p_\alpha(z)}$ is approximated by short-run MCMC from the prior.

Let $\psi = (\phi, \beta)$ be the placeholder for the parameters of the inference network ϕ and the generator network β. The learning gradient for ψ is

$$\delta_\psi = \nabla_\psi \text{ELBO} = \nabla_\psi \mathbb{E}_{q_\phi(z|x)}\left[\log p_\beta(x|z)\right] - \nabla_\psi \mathbb{D}_{\text{KL}}\left(q_\phi(z|x) \| p_0(z)\right) + \nabla_\psi \mathbb{E}_{q_\phi(z|x)}\left[F_\alpha(z)\right] \tag{2.23}$$

where $\mathbb{E}_{q_\phi(z|x)}$ involved in the two terms is approximated by samples from the inference network with reparametrization trick and the $\mathbb{D}_{\text{KL}}$ term is analytical tractable.

For labelled example x, the log-likelihood can be decomposed into

$$\log p_\theta(x, y) = \log p_\theta(x) + \log p_\alpha(y|x) \tag{2.24}$$

The first term can be optimized as the unsupervised learning scheme above and the second term can also be approximated by samples from the inference network,

$$\log p_\theta(y|x) = \log \mathbb{E}_{p_\theta(z|x)} \left[p_\alpha(y|z)\right] \approx \log \mathbb{E}_{q_\phi(z|x)} \left[p_\alpha(y|z)\right] \tag{2.25}$$

The learning gradients for the inference and prior model are computed accordingly,

$$\nabla_{\alpha,\phi} \log p_\theta(y|x) \approx \nabla_{\alpha,\phi} \log \mathbb{E}_{q_\phi(z|x)} \left[p_\alpha(y|z)\right] \tag{2.26}$$

As in (2.16), $p_\alpha(y|z)$ is a softmax classifier. In the setting of VAE, the inference network produces $\mu_\phi(x)$ and $\sigma^2_\phi(x)$ and latent variable z is sampled by reparametrization trick as

$$z = \mu_\phi(x) + \sigma_\phi(x)\epsilon, \epsilon \sim N(0, I_d) \tag{2.27}$$

In the supervised scheme, we only need to use mean without variance to perform classification, so that the (2.25) can further be refined as

$$\log p_\theta(y|x) \approx \log \mathbb{E}_{q_\phi(z|x)} \left[p_\alpha(y|z = \mu_\phi(x))\right] \tag{2.28}$$

The full algorithm with semi-supervised scheme is detailed in Algorithm 4.

Algorithm 4: Semi-supervised learning of SVEBM via variational inference

```
Input  : Total learning iterations $T$, initial parameters $\theta_0 = (\alpha_0, \beta_0, \phi_0)$ for the EBM, the
         generator model and the inference model respectively; learning rate for three
         learning procedures $(\eta_0, \eta_1, \eta_2)$, observed unlabelled training examples $\{x_i\}_{i=1}^{M}$
         and labelled examples $\{x_i\}_{i=M+1}^{M+N}$, batch size for unlabelled and labelled examples
         $(m, n)$, number of prior sampling $K_\alpha$, step size in prior sampling $s_\alpha$.
Output : $\theta_T = (\alpha_T, \beta_T, \phi_T)$
for $t = 0 : T-1$ do
    1. Mini-batch: Sample unlabelled examples $\{x_i\}_{i=1}^{m}$ and labelled examples $\{x_i\}_{i=1}^{n}$.
    2. Prior sampling: For each $x_i$, sample $z_i^- \sim \tilde{p}_{\alpha_t}(z)$ using (2.13), where
     $\pi(z) = p_{\alpha_t}(z)$, $s = s_\alpha$ and $K = K_\alpha$.
    3. Posterior sampling: For each $x_i$, sample $z_i^+ \sim q_\phi(z|x_i)$ using (2.13) from the
     inference network with reparametrization trick.
    if Unsupervised learning then
        4. Learning prior model: $\alpha_{t+1} = \alpha_t + \eta_0 \frac{1}{m} \sum_{i=1}^{m} [\nabla_\alpha F_{\alpha_t}(z_i^+) - \nabla_\alpha F_{\alpha_t}(z_i^-)]$
        5. Learning inference and generator models:
         $\psi_{t+1} = \psi_t + \eta_1 \frac{1}{m} \sum_{i=1}^{m} \nabla_\psi \left[\log p_\beta(x_i|z_i^+) - \mathbb{D}_{\mathrm{KL}}\left(q_\phi(z|x) \| p_0(z)\right) + F_\alpha(z_i^+)\right]$
    if Supervised learning then
        6. Learning prior and inference models: Let $\gamma = (\alpha, \phi)$.
         $\gamma_{t+1} = \gamma_t + \eta_2 \frac{1}{n} \sum_{i=m+1}^{m+n} \nabla_\gamma \log p_{\alpha_t}(y_i | z_i^+ = \mu_{\phi_t}(x_i))$
```

CHAPTER 3

SVEBM for Unsupervised Meta-Learning

Meta-learning is a broad concept to learn meta-knowledge across diverse tasks, such that the learned meta-knowledge can be further adapted to novel tasks. Unsupervised learning is a fundamental, unsolved problem and has witnessed promising results in wide range of domains. The core idea of unsupervised learning is to enable a proxy objective function to learn internal representations such that they can be used or fine-tuned to a downstream supervised task.

If an ultimate goal of unsupervised learning is to learn useful representations, can we derive an unsupervised learning objective that explicitly takes into account how the representation will be used? In that sense, meta-learning shares the spirit of unsupervised learning in that they attempt to discover a learning procedure that is more efficient and effective than learning from scratch.

However, the biggest obstacle ahead is that in meta-learning, there are two types of supervision from human: one is supervised labeled data in each task, the other is human-crafted task distribution. The former is easy to overcome in unsupervised case, while the latter is extremely hard in unsupervised learning schema, which results in failed learning.

To perform unsupervised meta-learning, we have to bear in mind that it differs from the supervised meta-learning in that the meta-training task space might be different from the meta-test task space. Since random generated tasks contain insufficient regularity to enable useful meta-learning, the unsupervised clustering is performed to get pseudo-labels followed by a general supervised MAML. Another approach is to learn the multi-modal task-specific prior from irregular tasks and then requires additional knowledge trick when adapting from the meta-training task space to meta-test task space. In this chapter, we follow the second line and introduce the Meta-SVEBM.

3.1 Unsupervised Meta-Learning

In Section 1.2, we have given a systemic sketch of supervised meta-learning in the few-shot classification scenario as MAML, where we further illustrate the unsupervised version of probabilistic MAML in Figure 1.4. However, MAML is not the end nor the beginning of the end of the meta-learning. It's probably a great end of the beginning. Especially for current unsupervised meta-learning literature, the task-specific parameter ϕ_i can be optimized implicitly from meta-parameter θ instead of explicit gradient updates as a MAP estimate. Moreover, rather than a two-stage framework as first unsupervised clustering for task generation, followed by MAML, we can directly use the multi-modal prior to represent latent class labels as shown in Section 3.2. Here we illustrate the general form of unsupervised meta-learning without any constraints in Figure 3.1.

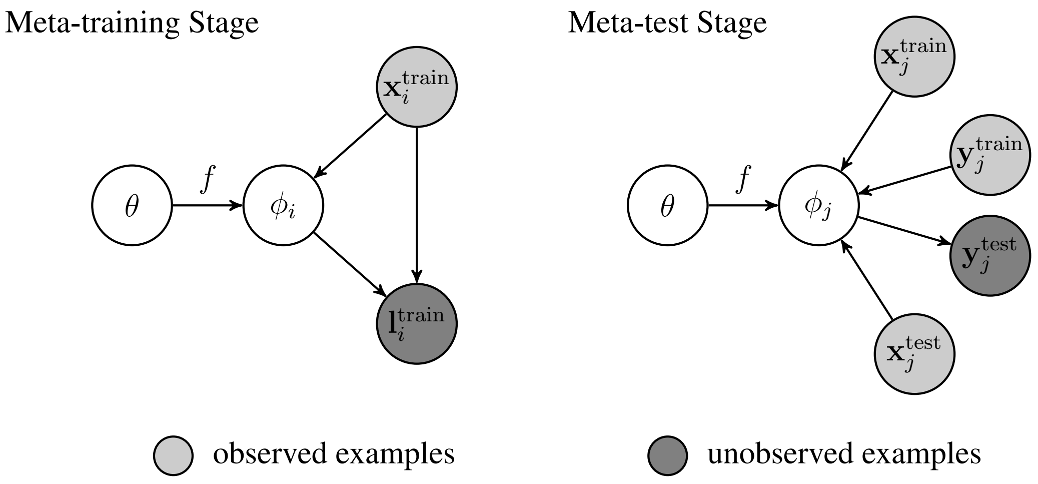

Meta-training Stage
Meta-test Stage
θ
f
ϕ_i
$\mathbf{x}_i^{\text{train}}$
$\mathbf{l}_i^{\text{train}}$
θ
f
ϕ_j
$\mathbf{x}_j^{\text{train}}$
$\mathbf{y}_j^{\text{train}}$
$\mathbf{y}_j^{\text{test}}$
$\mathbf{x}_j^{\text{test}}$
observed examples
unobserved examples

Figure 3.1: Graphical illustration of unsupervised meta-learning. Here task-specific parameter is obtained from meta-parameter by function f. During meta-training, for each sampled task $\mathcal{T}_i$, infer the pseudo-label $\mathbf{l}_i^{\text{train}}$ for each training example, and then optimize the model with respect to θ and ϕ_i. Meta-test stage is the same as supervised meta-training.

From Figure 3.1, during the meta-training stage, we perform exact unsupervised learning on few-shot classification task, while in the meta-test stage, we can either perform supervised learning or semi-supervised learning using the labelled data.

3.2 Model and Learning: meta-level SVEBM

Problem Statement The goal of unsupervised meta-learning is to obtain sufficient general feature embeddings such that they can be adapted to a wide range of human-crafted downstream few-shot classification tasks. During the meta-learning stage, we only assume an unlabelled dataset as $\mathcal{D}_u = \{\mathbf{x}_u\}_{u=1}^{U}$. In the meta-test stage, we tend to use learned meta-parameter to solve a M-way, S-shot classification task, which is to use S support data and labels $\{(\mathbf{x}_s, \mathbf{y}_s)\}_{s=1}^{S}$ to correctly predict the labels of query set with Q unlabelled data as $\{\mathbf{x}_q\}_{q=1}^{Q}$ as aforementioned in Section 1.1.

Unsupervised meta-training During meta-training, we aim to learn sufficient internal representations that can be further transferred to human-crafted few-shot classification tasks. We tend to directly adopt the VAE framework by recruiting another inference network to infer the latent variable. Standing on the empirical Bayesian perspective, the prior is learned from the observed data, which can be modeled by an EBM. The model is detailed in Section 2.2.1. To be brief, suppose we have an EBM parameterized by a Neural Network with parameter α, an inference network with parameter γ and a generator with parameter β. The prior model is

$$p_\alpha(z) = \frac{1}{Z_\alpha} \exp\left(F_\alpha(z)\right) p_0(z) \tag{3.1}$$

where $F_\alpha(z)$ denotes marginal energy as the form of log-sum-exponential.

$$F_\alpha(z) = \log \sum_y \exp\left(\langle y, f_\alpha(z)\rangle\right) \tag{3.2}$$

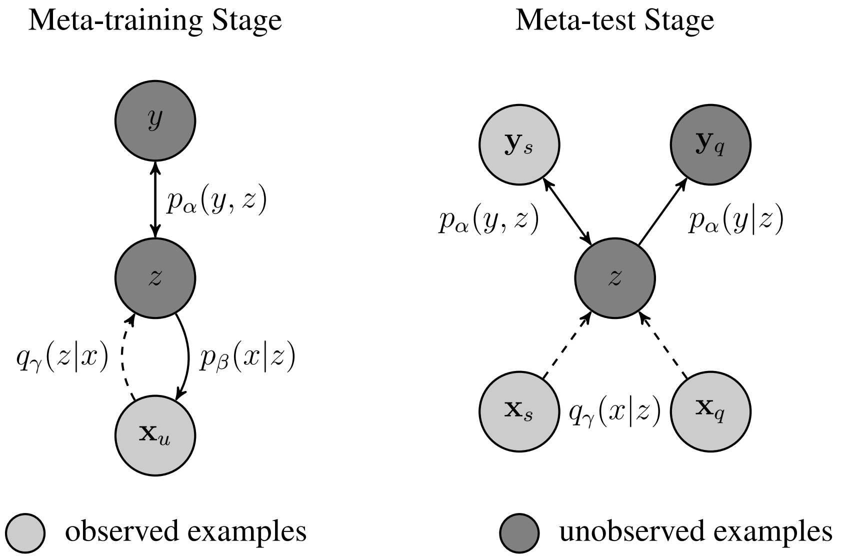
During meta-training, since y is unobserved, we can assume a uniform prior for simplicity so that $F_\alpha(z) = \log \sum \exp\left(f_\alpha(z)\right)$. In this way, unsupervised meta-training is to match the expressive EBM prior to the variational posterior. The posterior of z is sampled from the inference network $q_\gamma(z|x)$ using reparametrization trick. The generator is a top-down model with deconvolutional layers,

$$x = g_\beta(z) + \epsilon \tag{3.3}$$

where $\epsilon \sim N(0, \sigma^2 I_D)$ is random noise with assumed variance σ^2. Then $p_\beta(x|z) = N(g_\beta(z), \sigma^2 I_D)$. The objective function for the meta-training is approximated by ELBO as (2.21).

Supervised meta-test During meta-test, we aim to solve a predefined few-shot classification task with learned meta-parameter. In that sense, we have to carefully define the meta-parameter θ and the task-specific parameter ϕ_i as the terminology used in Chapter 1.

Our task-specific parameter is obviously defined as the task-specific prior modeled by EBM so that $z \sim p_\alpha(z)$ is treated as ϕ_i. It's a little tricky to find out meta-parameter θ. During the meta-training, what has been invariant to specific tasks is the learned internal representations or embeddings for training example x, that is θ as learned $q_\gamma(z|x)$. As what has been discussed in probabilistic MAML in Section 1.2.2, meta-learning is to infer joint distribution of $q(\theta, \phi_i) = q(\theta)q(\phi_i|\theta)$, which is instantiated as $p(x, z) = p_\alpha(z)q_\gamma(x|z)$. In that sense, during meta-test, we can only update EBM as a few-shot classifier when fixing the parameter γ in inference network. The model can be learned by supervised SVEBM.


Meta-training Stage

Meta-test Stage

Figure 3.2: Graphical illustration of the meta-level SVEBM. The dashed lines indicate variational inference. During the meta-training stage (left), y and z are coupled in the latent space which cannot be observed. The model learns the variational posterior by matching the task-specific prior from SVEBM. During the meta-test stage, the posterior is inferred from the inference network. Using the labelled information from $\mathbf{y}_s$, predict the label $\mathbf{y}_q$ using a softmax classifier $p_\alpha(y|z)$.

Multi-task learning for EBM As discussed above, the task-specific prior model is formulated as an EBM. When the model ingests n tasks as a batch, the EBM should generate independent priors for each individual task in a sense that we should instantiate n same structures to generate n priors as $\{z_i\}_{i=1}^n$, such as n 2-layer multi-layer perceptron (MLP). However in the multi-task setting, we do not assume independence between the tasks. We believe there are shared structures among different tasks to certain degree, especially in the unsupervised scenario. In that sense, we can assume those tasks are conditionally independent on some shared structures. We have made a simple attempt as multi-head EBM. The multi-task learning choice is empirical, whether it's positive transfer or negative transfer is heavily based on the concrete tasks. One obvious benefit of using multi-head EBM is to avoid over-fitting for individual task prior generation[Rud17].

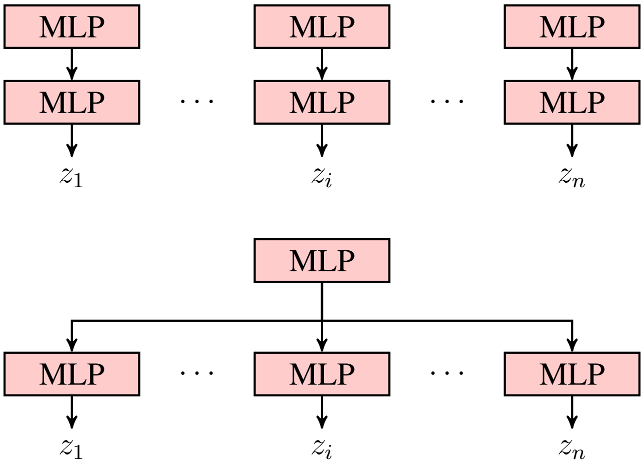


Figure 3.3: Graphical illustration of model architectures of EBM. In our experiment, we use the multi-head version (bottom) instead of the independent version (top).

The full algorithm for meta-level SVEBM is shown in Algorithm 5.

Algorithm 5: Unsupervised learning of meta-level SVEBM

```
Input  : Meta-training iterations $T$, meta-test iterations $T_{\text{te}}$, initial parameters
         $\Theta_0 = (\alpha_0, \beta_0, \gamma_0)$, step size $(\eta_0, \eta_1, \eta_2, \eta_3)$, Langevin sampling step size $s_\alpha$,
         iterations $K_\alpha$, batch size $n$.
Output : $\Theta_T = (\alpha_T, \beta_T, \gamma_T)$
if Meta-training then
    for $t = 0 : T - 1$ do
        1. Mini-batch: Sample a batch of $n$ training tasks $\{\mathcal{T}_i\}_{i=1}^{n} \sim p(\mathcal{T})$, each of which
           has $U$ unlabelled examples $\{x_u\}_{u=1}^{U}$.
        2. Task-specific prior sampling: For each $x_u$, sample $z_u^- \sim \tilde{p}_{\alpha_t}(z)$ using (2.13),
           where $\pi(z) = p_{\alpha_t}(z)$, $s = s_\alpha$ and $K = K_\alpha$.
        3. Variational posterior sampling: For each $x_u$, sample $z_u^+ \sim q_\gamma(z|x_u)$ using
           (2.13) from the inference network with reparametrization trick.
        4. Learning prior model for each head:
           $\alpha_{t+1} = \alpha_t + \eta_0 \frac{1}{U} \sum_u [\nabla_\alpha F_{\alpha_t}(z_u^+) - \nabla_\alpha F_{\alpha_t}(z_u^-)]$
        5. Learning inference and generator models: $\psi = (\beta, \gamma)$
           $\psi_{t+1} = \psi_t + \eta_1 \frac{1}{nU} \sum_{u=1}^{nU} \nabla_\psi \left[\log p_\beta(x_u|z_u^+) - \mathbb{D}_{\text{KL}}\left(q_\gamma(z|x) \| p_0(z)\right) + F_\alpha(z_u^+)\right]$
if Meta-test then
    for $t = 0 : T - 1$ do
        1. Mini-batch: Sample a batch of $n$ training tasks $\{\mathcal{T}_j\}_{j=1}^{n} \sim p(\mathcal{T})$, each of which
           has $S$ support examples $\{x_s, y_s\}_{s=1}^{S}$ and $Q$ query examples $\{x_q\}_{q=1}^{Q}$.
        2. Task-specific prior sampling: For each $x_s, x_q$, sample $z_s^-, z_1^- \sim \tilde{p}_{\alpha_t}(z)$.
        3. Variational posterior sampling: For each $x_s, x_q$, sample $z_s^+, z_q^+ \sim q_\gamma(z|x_u)$
        4. Learning prior model using query set:
           $\alpha_{t+1} = \alpha_t + \eta_2 \frac{1}{Q} \sum_q [\nabla_\alpha F_{\alpha_t}(z_q^+) - \nabla_\alpha F_{\alpha_t}(z_q^-)]$
        5. Learning prior model using support set:
           $\alpha_{t+1} = \alpha_t + \eta_3 \frac{1}{s} \sum_{s=1}^{S} \nabla_\alpha \log p_{\alpha_t}(y_s|z_s^+ = \mu_\gamma(x_s))$
```

3.3 Experiment

In this section, we are trying to validate the effectiveness of meta-level SVEBM on a set of downstream human-crafted tasks by comparison with other baselines.

Baselines We mainly compare our model with three types of unsupervised meta-learning approaches and one supervised meta-learning approaches. **(1) MAML**[FAL17]: model-agnostic meta learning by in Section 1.2 as one supervised meta-learning baseline. **(2) CACTUs**[HLF18]: clustering to automatically construct tasks for unsupervised meta-learning, also known as a unsupervised version of MAML. **(3) UMTRA**[KBS18]: unsupervised meta-learning with tasks constructed by random sampling and augmentation by, also an extended version of unsupervised MAML. **(4) Meta-GMVAE**[LML20]: a meta-level VAE with prior parameterized as a Gaussian mixture prior optimized by Expectation-Maximum (EM) algorithm. **(5) Meta-SVEBM**: our proposed method also build a meta-level VAE while the prior is parameterized by learned SVEBM.

Datasets We implement above models on two benchmark datasets for few-shot classification task. **(1) Omniglot**: This is a dataset with 28×28 gray-scale hand-written characters with 1623 different classes, each of which has 20 examples. Following the experiment setting in [HLF18], we used 1200 classes for unsupervised meta-training, 100 classes for validation and the rest 323 classes for meta-test. We further perform data augmentation by rotating each instance for 90, 180, and 270 degrees so that we have total four times data than the original. **(2) Mini-ImageNet**. The Mini-ImageNet is a crafted subset of ImageNet with total 100 classes, each of which consists of 600 84×84 images. The 64 classes are used for meta-training and the rest 16 classes for validation and 20 classes for meta-test. Instead of directly using the raw images for training, following [LML20], we use high-level features learned from a out-of-shelf contrastive learning framework as SimCLR[CKN20].

Visualization The detailed trained results on Omniglot is shown in Figure 3.4. In this figure, we compare the reconstructed results with the real examples, showing that VAE is learned successfully. We also plot the generated examples where the latent variable z is sampled from the prior model

$p_\alpha(z)$, which shows the multi-modality captured by the EBM and the power of short-run MCMC. As the increase length of Markov chain, the latent variable with lower energy is sampled, which generates more clear characters.

Results In table 3.13.2, we show the few-shot classification results on 1000 randomly sampled tasks comparing with the other baselines mentioned above. Comparing to the state-of-the-art, the Meta-SVEBM outperforms all of the other approaches on MiniImageNet dataset and achieves nearly the best on Omniglot dataset and even comparable to the supervised MAML in both datasets, showing the effectiveness of Meta-SVEBM.

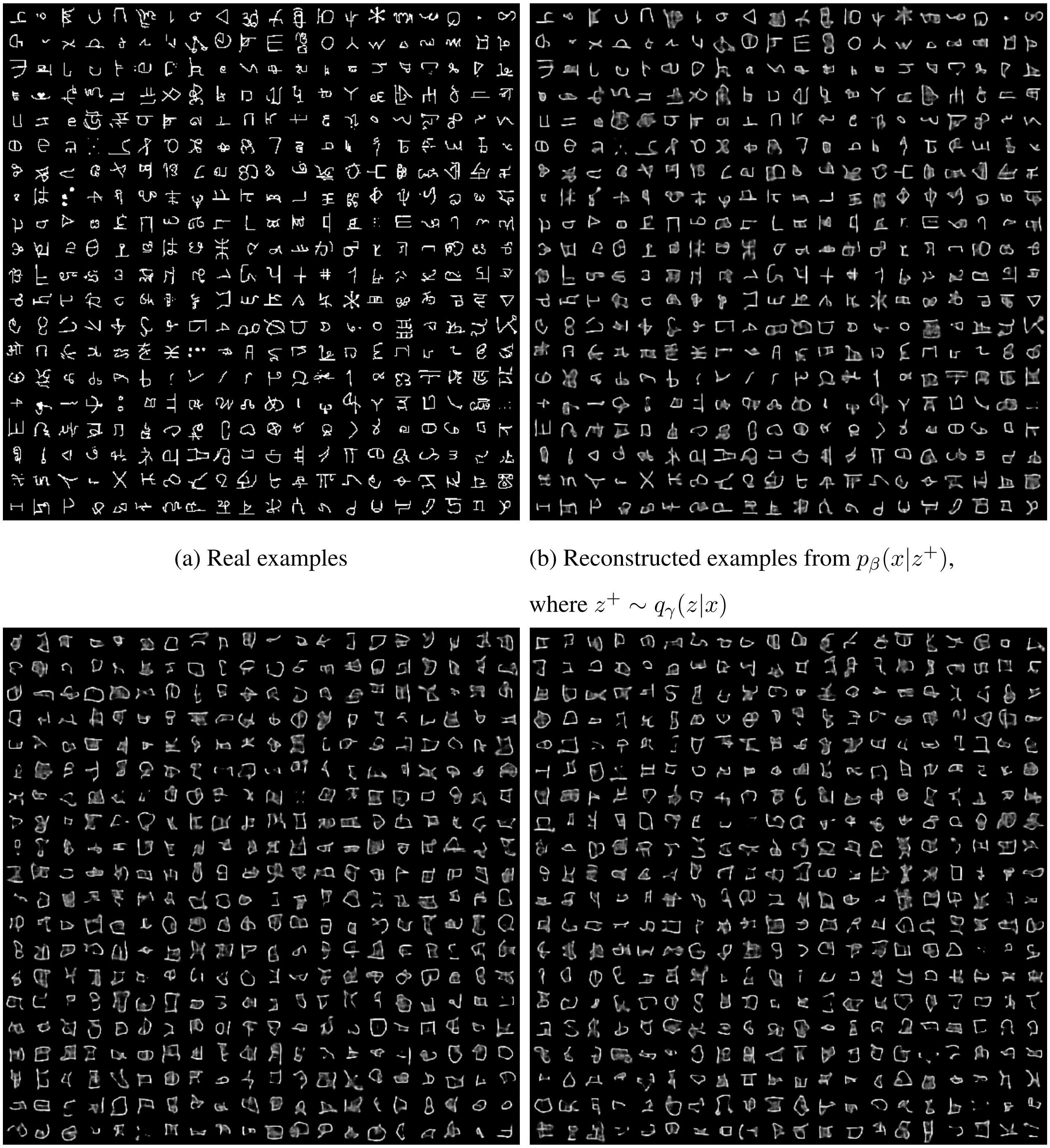
Discussion Meta-SVEBM learns a task-specific prior by matching the variational posterior. One of the amazing phenomena is that with the increasing of the shots, Meta-SVEBM can boost much more performance than the other methods. However, since the algorithm doesn't involve a task generation procedure, it's possible that the meta-training tasks are not sampled from exactly the same task distribution as meta-test tasks. Bridging the gap between the meta-training task space and meta-test task space matters here. Meta-GMVAE designs a special initialization trick for adapting meta-parameters to task-specific parameters for each meta-test task. We should also consider it in the near future.

Table 3.1: Summary of few-shot classification results (way, shot) on the Omniglot dataset. The accuracy is calculated over 1000 randomly sampled meta-test tasks. Bold number denotes the best performance.

Method	Feature Extractor	(5,1)	(5,5)	(20,1)	(20,5)
CACTUs-MAML[HLF18]	BiGAN	58.18	78.66	35.56	58.62
CACTUs-ProtoNets[HLF18]	BiGAN	54.74	71.69	33.40	50.62
CACTUs-MAML[HLF18]	ACAI	68.84	87.78	48.09	73.36
CACTUs-ProtoNets[HLF18]	ACAI	68.12	83.58	47.75	66.27
UMTRA-MAML[KBS18]	N/A	83.80	95.43	74.25	92.12
LASIUM-N-VAE-MAML[KZV20]	N/A	76.11	94.42	–	–
LASIUM-OC-VAE-ProtoNets[KZV20]	N/A	73.22	85.05	–	–
LASIUM-RO-GAN-MAML[KZV20]	N/A	83.26	95.29	–	–
LASIUM-RO-GAN-ProtoNets[KZV20]	N/A	80.15	91.10	–	–
Meta-GMVAE[LML20]	N/A	**94.92**	97.09	**82.21**	90.06
Meta-SVEBM (Ours)	N/A	90.95	**97.38**	77.63	**92.23**
MAML (*supervised*)	N/A	94.46	98.83	84.60	96.29
ProtoNets (*supervised*)	N/A	98.35	99.58	95.31	98.81

Table 3.2: Summary of few-shot classification results (way, shot) on the miniImagenet dataset. The accuracy is calculated over 1000 randomly sampled meta-test tasks. Bold number denotes the best performance.

Method	Feature Extractor	(5,1)	(5,5)	(5,20)	(5,50)
CACTUs-MAML[HLF18]	BiGAN	36.24	51.28	61.33	66.91
CACTUs-ProtoNets[HLF18]	BiGAN	36.62	50.16	59.56	63.27
CACTUs-MAML[HLF18]	DeepCluster	39.90	53.97	63.84	69.64
CACTUs-ProtoNets[HLF18]	DeepCluster	39.18	53.36	61.54	63.55
UMTRA-MAML[KBS18]	N/A	39.93	50.73	61.11	67.15
LASIUM-N-GAN-MAML[KZV20]	N/A	40.19	54.56	65.17	69.13
LASIUM-N-GAN-ProtoNets[KZV20]	N/A	40.05	52.53	59.45	61.43
CACTUs-MAML[LML20]	SimCLR	40.39	52.35	61.09	64.89
UMTRA-MAML[LML20]	SimCLR	40.85	51.47	61.03	67.30
Meta-GMVAE[LML20]	SimCLR	42.82	55.73	63.14	68.26
Meta-SVEBM (Ours)	SimCLR	**43.35**	**56.25**	**67.32**	**73.25**
MAML (*supervised*)	N/A	46.81	62.12	71.03	75.54
ProtoNets (*supervised*)	N/A	46.56	62.29	70.07	72.04



(a) Real examples

(b) Reconstructed examples from $p_\beta(x|z^+)$, where $z^+ \sim q_\gamma(z|x)$

(c) Generated example from $p_\beta(x|z^-)$ where $z^- \sim p_\alpha(z)$

(d) Generated example from $p_\beta(x|z^-)$ after k steps short-run MCMC

Figure 3.4: Visualization of learned Meta-SVEBM.

REFERENCES

[CKN20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A simple framework for contrastive learning of visual representations." In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks." In *International Conference on Machine Learning*, pp. 1126–1135. PMLR, 2017.

[Fin18] Chelsea B Finn. *Learning to learn with gradients*. University of California, Berkeley, 2018.

[FXL18] Chelsea Finn, Kelvin Xu, and Sergey Levine. "Probabilistic model-agnostic meta-learning." *arXiv preprint arXiv:1806.02817*, 2018.

[GPM20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial networks." *Communications of the ACM*, **63**(11):139–144, 2020.

[HLF18] Kyle Hsu, Sergey Levine, and Chelsea Finn. "Unsupervised learning via meta-learning." *arXiv preprint arXiv:1810.02334*, 2018.

[HZR16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[KBS18] Siavash Khodadadeh, Ladislau Bölöni, and Mubarak Shah. "Unsupervised meta-learning for few-shot image classification." *arXiv preprint arXiv:1811.11819*, 2018.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*, **25**:1097–1105, 2012.

[KW13] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes." *arXiv preprint arXiv:1312.6114*, 2013.

[KZV20] Siavash Khodadadeh, Sharare Zehtabian, Saeed Vahidian, Weijia Wang, Bill Lin, and Ladislau Bölöni. "Unsupervised Meta-Learning through Latent-Space Interpolation in Generative Models." *arXiv preprint arXiv:2006.10236*, 2020.

[LCH06] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. "A tutorial on energy-based learning." *Predicting structured data*, **1**(0), 2006.

[LML20] Dong Bok Lee, Dongchan Min, Seanie Lee, and Sung Ju Hwang. “Meta-GMVAE: Mixture of Gaussian VAE for Unsupervised Meta-Learning.” In *International Conference on Learning Representations*, 2020.

[NPH20] Erik Nijkamp, Bo Pang, Tian Han, Linqi Zhou, Song-Chun Zhu, and Ying Nian Wu. “Learning multi-layer latent variable model via variational optimization of short run mcmc for approximate inference.” In *European Conference on Computer Vision*, pp. 361–378. Springer, 2020.

[PHN20] Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. “Learning latent space energy-based prior model.” *arXiv preprint arXiv:2006.08205*, 2020.

[PW21] Bo Pang and Ying Nian Wu. “Latent Space Energy-Based Model of Symbol-Vector Coupling for Text Generation and Classification.” In *International Conference on Machine Learning*, pp. 8359–8370. PMLR, 2021.

[RL16] Sachin Ravi and Hugo Larochelle. “Optimization as a model for few-shot learning.” 2016.

[Rud17] Sebastian Ruder. “An overview of multi-task learning in deep neural networks.” *arXiv preprint arXiv:1706.05098*, 2017.

[SSS17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. “Mastering the game of go without human knowledge.” *nature*, **550**(7676):354–359, 2017.

[Ten99] Joshua Brett Tenenbaum. *A Bayesian framework for concept learning*. PhD thesis, Massachusetts Institute of Technology, 1999.

[UO30] George E Uhlenbeck and Leonard S Ornstein. “On the theory of the Brownian motion.” *Physical review*, **36**(5):823, 1930.

[VSP17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention is all you need.” In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[XZL20] Jianwen Xie, Zilong Zheng, and Ping Li. “Learning energy-based model with variational auto-encoder as amortized sampler.” *arXiv preprint arXiv:2012.14936*, 2020.

[ZWM98] Song Chun Zhu, Yingnian Wu, and David Mumford. “FRAME: Filters, random field and maximum entropy:—towards a unified theory for texture modeling.” In *International Journal on Computer Vision*. Citeseer, 1998.