HDSR
HARVARD DATA SCIENCE REVIEW

Volume 0 Issue 0
DOI: 00.000
ISSN: 2644-2353

Toward a 'Standard Model' of Machine Learning

Zhiting Hu[†,*], Eric P. Xing[‡,◇,♮,**]

[†] Halıcıoğlu Data Science Institute, University of California San Diego, San Diego, USA

[‡] Machine Learning Department, Carnegie Mellon University, Pittsburgh, USA

[♮] Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE

[◇] Petuum Inc., Pittsburgh, USA

Abstract. Machine learning (ML) is about computational methods that enable machines to learn concepts from experience. In handling a wide variety of experience ranging from data instances, knowledge, constraints, to rewards, adversaries, and lifelong interaction in an ever-growing spectrum of tasks, contemporary ML/AI (artificial intelligence) research has resulted in a multitude of learning paradigms and methodologies. Despite the continual progresses on all different fronts, the disparate narrowly focused methods also make standardized, composable, and reusable development of ML approaches difficult, and preclude the opportunity to build AI agents that panoramically learn from all types of experience. This article presents a standardized ML formalism, in particular a 'standard equation' of the learning objective, that offers a unifying understanding of many important ML algorithms in the supervised, unsupervised, knowledge-constrained, reinforcement, adversarial, and online learning paradigms, respectively—those diverse algorithms are encompassed as special cases due to different choices of modeling components. The framework also provides guidance for mechanical design of new ML approaches and serves as a promising vehicle toward *panoramic* machine learning with all experience.

Keywords: standard model, panoramic learning, experience, composable machine learning, unification

Media Summary

Humans learn from a range of experience, what about a computer? The past decades of AI and Machine Learning (ML) research has resulted in a multitude of paradigms and algorithms each specialized to train ML models with a certain type of information and experience in a certain type of problem. While pushing the field forward rapidly, the bewildering and ever-growing variety of paradigms and algorithms also makes it extremely difficult to master existing ML techniques and

[*]zhh019@ucsd.edu, [**]epxing@cs.cmu.edu

to develop universal, repeatable, and reusable computer programs that can simultaneously learn from diverse experience in the real world. It is a constant desire and aspiration to search for a standardized ML formalism that unifies the distinct learning principles, much like the Standard Model in physics, to gain a more holistic understanding of the diverse paradigms and algorithms, lay out a blueprint permitting fuller and more systematic exploration in the design and analysis of new algorithms, and eventually serves as a vehicle toward panoramic machine learning capable of integrating all available information (data, knowledge, constraints, reward, adversary, etc.) in learning and thus applicable to all problems. This work presents an attempt toward this end. In particular, we establish a standard equation of the learning objective, which subsumes many of the known algorithms as special cases and offers guiding principles of designing new more powerful learning algorithms in a mechanical and composable way.

1. Introduction

Human learning has the hallmark of learning concepts from diverse sources of information. Take the example of learning a language. Humans can benefit from various experience—by observing examples through reading and hearing, studying abstract definitions and grammar, making mistakes and getting correction from teachers, interacting with others and observing implicit feedback, and so on. Knowledge of a prior language can also accelerate the acquisition of a new one. How can we build artificial intelligence (AI) agents that are similarly capable of learning from all types of experience? We refer to the capability of flexibly integrating all available experience in learning as *panoramic learning*.

In handling different experience ranging from data instances, knowledge, constraints, to rewards, adversaries, and lifelong interplay in an ever-growing spectrum of tasks, contemporary ML and AI research has resulted in a large multitude of learning paradigms (e.g., supervised, unsupervised, active, reinforcement, adversarial learning), models, optimization techniques, not mentioning countless approximation heuristics and tuning tricks, plus combinations of all the above. While pushing the field forward rapidly, these results also make mastering existing ML techniques very difficult, and fall short of reusable, repeatable, and composable development of ML approaches to diverse problems with distinct available experience.

Those fundamental challenges call for a standardized ML formalism that offers a principled framework for understanding, unifying, and generalizing current major paradigms of learning algorithms, and for mechanical design of new approaches for integrating any useful experience in learning. The power of standardized theory is perhaps best demonstrated in physics, which has a long history of pursuing symmetry and simplicity of its principles: exemplified by the famed Maxwell's equations in the 1800s that reduced various principles of electricity and magnetism into a single electromagnetic theory, followed by General Relativity in the 1910s and the Standard Model in the 1970s, physicists describe the world best by unifying and reducing different theories to a standardized one. Likewise, it is a constant quest in the field of ML to establish a 'Standard Model' (Domingos, 2015; Langley, 1989), that gives a holistic view of the broad learning principles, lays out a blueprint permitting fuller and more systematic exploration in the design and analysis of new algorithms, and eventually serves as a vehicle toward panoramic learning that integrates all available sources of experience.

This paper presents an attempt toward this end. In particular, our principal focus is on the *learning objective* that drives the model training given the experience and thus often lies at the

core for designing new algorithms, understanding learning properties, and validating outcomes. We investigate the underlying connections between a range of seemingly distinct ML paradigms. Each of these paradigms has made particular assumptions on the form of experience available. For example, the present most popular *supervised learning* relies on collections of data instances, often applying a maximum likelihood objective solved with simple gradient descent. Maximum likelihood based *unsupervised learning* instead can invoke different solvers, such as expectation-maximization (EM), variational inference, and wake-sleep in training for varied degree of approximation to the problem. *Active learning* (Settles, 2012) manages data instances which, instead of being given all at once, are adaptively selected. *Reinforcement learning* (Sutton & Barto, 2017) makes use of feedback obtained via interaction with the environment. *Knowledge-constrained learning* like posterior regularization (Ganchev et al., 2010; J. Zhu et al., 2014) incorporates structures, knowledge, and rules expressed as constraints. *Generative adversarial learning* (Goodfellow et al., 2014) leverages a companion model called discriminator to guide training of the model of interest.

In light of these results, we present a standard equation (SE) of the objective function. The SE formulates a rather broad design space of learning algorithms. We show that many of the well-known algorithms of the above paradigms are all instantiations of the general formulation. More concretely, the SE, based on the maximum entropy and variational principles, consists of three principled terms, including the *experience* term that offers a unified language to express arbitrary relevant information to supervise the learning, the *divergence* term that measures the fitness of the target model to be learned, and the *uncertainty* term that regularizes the complexity of the system. The single succinct formula re-derives the objective functions of a large diversity of learning algorithms, reducing them to different choices of the components. The formulation thus shed new light on the fundamental relationships between the diverse algorithms that were each originally designed to deal with a specific type of experience.

The modularity and generality of the framework is particularly appealing not only from the theoretical point of view, but also because it offers guiding principles for designing algorithmic approaches to new problems in a mechanical way. Specifically, the SE by its nature allows combining together all different experience to learn a model of interest. Designing a problem solution boils down to choosing *what* experience to use depending on the problem structure and available resources, without worrying too much about *how* to use the experience in the training. Besides, the standardized ML perspective also highlights that many learning problems in different research areas are essentially the same and just correspond to different specifications of the SE components. This enables us to systematically repurpose successful techniques in one area to solve problems in another.

The remainder of the article is organized as follows. Section 2 gives an overview of relevant learning and inference techniques as a prelude of the standardized framework. Section 3 presents the standard equation as a general formulation of the objective function in learning algorithms. The subsequent two sections discuss different choices of two of the key components in the standard equation, respectively, illustrating that many existing methods are special cases of the formulation: Section 4 is devoted to discussion of the experience function and Section 5 focuses on the divergence function. Section 6 discusses an extended view of the standard equation in dynamic environments. Section 7 focuses on the optimization algorithms for solving the standard equation objective. Section 8 discusses the diverse types of target models. Section 9 discusses the utility of the standardized formalism for mechanical design of panoramic learning approaches. Section 10

reviews related work. Section 11 concludes the article with discussion of future directions—in particular, we discuss the broader aspects of ML not covered in the present work (e.g., more advanced learning such as continual learning in complex evolving environments, theoretical analysis of learnability, generalization and complexity, and automated algorithm composition) and how their unified characterization based on or inspired by the current framework could potentially lead toward a full 'Standard Model' of ML and a turnkey approach to panoramic learning with all types of experience.

2. Preliminaries: The Maximum Entropy View of Learning and Inference

Depending on the nature of the task (e.g., classification or regression), data (e.g., labeled or unlabeled), information scope (e.g., with or without latent variables), and form of domain knowledge (e.g., prior distributions or parameter constraints), and so on, different learning paradigms with often complementary (but not necessarily easy to combine) advantages have been developed for different needs. For example, the paradigms built on the maximum likelihood principles, Bayesian theories, variational calculus, and Monte Carlo simulation have led to much of the foundation underlying a wide spectrum of probabilistic graphical models, exact/approximate inference algorithms, and even probabilistic logic programs suitable for probabilistic inference and parameter estimations in multivariate, structured, and fully or partially observed domains, while the paradigms built on convex optimization, duality theory, regularization, and risk minimization have led to much of the foundation underlying algorithms such as support vector machine (SVM), boosting, sparse learning, structure learning, and so on. Historically, there have been numerous efforts in establishing a unified machine learning framework that can bridge these complementary paradigms so that advantages in model design, solver efficiency, side-information incorporation, and theoretical guarantees can be translated across paradigms. As a prelude of our presentation of the 'standard equation' framework toward this goal, here we begin with a recapitulation of the maximum entropy view of statistical learning. By naturally marrying the probabilistic frameworks with the optimization-theoretic frameworks, the maximum entropy viewpoint had played an important historical role in offering the same lens to understanding several popular methodologies such as maximum likelihood learning, Bayesian inference, and large margin learning.

2.1. **Maximum Likelihood Estimation (MLE).**

We start with the maximum entropy perspective of the maximum likelihood learning.

2.1.1. ***Supervised MLE***.

We consider an arbitrary probabilistic model (e.g., a neural network or probabilistic graphical model for, say, language generation) with parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ to be learned. Let $p_\theta(\boldsymbol{x}) \in \mathcal{P}(\mathcal{X})$ denote the distribution defined by the model, where $\mathcal{X}$ is the data space (e.g., all language text) and $\mathcal{P}(\mathcal{X})$ denotes the set of all probability distributions on $\mathcal{X}$. Given a set of independent and identically distributed (i.i.d.) data examples $\mathcal{D} = \{\boldsymbol{x}^* \in \mathcal{X}\}$, the most common method for estimating the parameters $\boldsymbol{\theta}$ is perhaps maximum likelihood estimation (MLE). MLE learns the model by minimizing the negative log-likelihood:

$$\min_{\boldsymbol{\theta}} -\mathbb{E}_{\boldsymbol{x}^* \sim \mathcal{D}} \left[\log p_\theta(\boldsymbol{x}^*)\right]. \tag{2.1}$$

MLE is known to be intimately related to the *maximum entropy* principle (Jaynes, 1957). In particular, when the model $p_\theta(\boldsymbol{x})$ is in the *exponential family* of the form:

$$p_\theta(\boldsymbol{x}) = \exp\left\{\boldsymbol{\theta} \cdot T(\boldsymbol{x})\right\} / Z(\boldsymbol{\theta}), \tag{2.2}$$

where $T(\boldsymbol{x})$ is the sufficient statistics of data $\boldsymbol{x}$ and $Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}\in\mathcal{X}} \exp\{\boldsymbol{\theta}\cdot T(\boldsymbol{x})\}$ is the normalization factor, it is shown that MLE is the convex dual of maximum entropy estimation.

In a maximum entropy formulation, rather than assuming a specific parametric from of the target model distribution, denoted as $p(\boldsymbol{x})$, we instead impose constraints on the model distribution. Specifically, in the supervised setting, the constraints require the expectation of the features $T(\boldsymbol{x})$ to be equal to the empirical expectation:

$$\mathbb{E}_p\left[T(\boldsymbol{x})\right] = \mathbb{E}_{\boldsymbol{x}^*\sim\mathcal{D}}\left[T(\boldsymbol{x}^*)\right]. \tag{2.3}$$

In general, there exist many distributions $p \in \mathcal{P}(\mathcal{X})$ that satisfy the constraint. The principle of maximum entropy resolves the ambiguity by choosing the distribution such that its Shannon entropy, $\mathrm{H}(p) := -\mathbb{E}_p[\log p(\boldsymbol{x})]$, is maximized. Following this principle, in the supervised setting, we thus have the specific constrained optimization problem:

$$\begin{aligned} &\max_{p(\boldsymbol{x})} \mathrm{H}\left(p(\boldsymbol{x})\right) \\ &s.t.\ \mathbb{E}_p\left[T(\boldsymbol{x})\right] = \mathbb{E}_{\boldsymbol{x}^*\sim\mathcal{D}}\left[T(\boldsymbol{x}^*)\right] \\ &\quad p(\boldsymbol{x}) \in \mathcal{P}(\mathcal{X}). \end{aligned} \tag{2.4}$$

The problem can be solved with the Lagrangian method. Specifically, we write the Lagrangian:

$$\mathcal{L}(p, \boldsymbol{\theta}, \mu) = \mathrm{H}(p(\boldsymbol{x})) - \boldsymbol{\theta}\cdot\left(\mathbb{E}_p\left[T(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{x}^*\sim\mathcal{D}}\left[T(\boldsymbol{x}^*)\right]\right) - \mu\left(\sum\nolimits_{\boldsymbol{x}} p(\boldsymbol{x}) - 1\right), \tag{2.5}$$

where $\boldsymbol{\theta}$ and μ are Lagrangian multipliers. Setting the derivative w.r.t. p and μ to equal zero implies that p must have the same form as in Equation 2.2:

$$p(\boldsymbol{x}) = \exp\left\{\boldsymbol{\theta}\cdot T(\boldsymbol{x})\right\}/Z(\boldsymbol{\theta}), \tag{2.6}$$

where we see the parameters $\boldsymbol{\theta}$ in the exponential family parameterization are the Lagrangian multipliers that enforce the constraints. Plugging the solution back into the Lagrangian, we obtain:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}^*\sim\mathcal{D}}\left[\boldsymbol{\theta}\cdot T(\boldsymbol{x}^*)\right] - \log Z(\boldsymbol{\theta}), \tag{2.7}$$

which is simply the negative of the MLE objective in Equation 2.1.

Thus maximum entropy is dual to maximum likelihood. It provides an alternative view of the problem of fitting a model into data, where the data instances in the training set are treated as constraints, and the learning problem is treated as a constrained optimization problem. This optimization-theoretic view of learning will be revisited repeatedly in the sequel to allow extending machine learning under all experience of which data instances is just a special case.

2.1.2. ***Unsupervised MLE.*** Similar to the MLE framework for supervised learning, unsupervised learning via MLE can also be reformulated as a constraint optimization problem with entropy maximization. Consider learning a multivariate model with latent variables, where each data instance is partitioned into observed variables $\boldsymbol{x} \in \mathcal{X}$ and latent variables $\boldsymbol{y} \in \mathcal{Y}$. For example, in the problem of image clustering, $\boldsymbol{x} \in \mathbb{R}^d$ is the observed image of d pixels and $\boldsymbol{y} \in \{1, \ldots, K\}$ is the unobserved cluster indicator (where K is the number of clusters). The goal is to learn a model $p_\theta(\boldsymbol{x}, \boldsymbol{y})$ that captures the joint distribution of $\boldsymbol{x}$ and $\boldsymbol{y}$. Since $\boldsymbol{y}$ is unobserved, we minimize the

negative log-likelihood with $\boldsymbol{y}$ marginalized out:

$$\min_{\boldsymbol{\theta}} -\mathbb{E}_{\boldsymbol{x}^* \sim \mathcal{D}} \left[\log \sum_{\boldsymbol{y} \in \mathcal{Y}} p_\theta(\boldsymbol{x}^*, \boldsymbol{y}) \right]. \tag{2.8}$$

Direct optimization of the marginal log-likelihood is typically intractable due to the summation over $\boldsymbol{y}$. Earlier work thus developed different solvers with varying levels of approximations.

It can be shown that the intractable negative log-likelihood above can be upper bounded by a more tractable term known as the *variational free energy* (Neal & Hinton, 1998). Let $q(\boldsymbol{y}|\boldsymbol{x})$ represent an arbitrary auxiliary distribution acting as a surrogate of the true posterior $p(\boldsymbol{y}|\boldsymbol{x})$, which is known as a variational distribution. Then, for each instance $\boldsymbol{x}^* \in \mathcal{D}$, we have:

$$\begin{aligned} -\log \sum_{\boldsymbol{y}} p_\theta(\boldsymbol{x}^*, \boldsymbol{y}) &= -\mathbb{E}_{q(\boldsymbol{y}|\boldsymbol{x}^*)} \left[\log \frac{p_\theta(\boldsymbol{x}^*, \boldsymbol{y})}{q(\boldsymbol{y}|\boldsymbol{x}^*)} \right] - \mathrm{KL}\left(q(\boldsymbol{y}|\boldsymbol{x}^*) \| p_\theta(\boldsymbol{y}|\boldsymbol{x}^*)\right) \\ &\leq -\mathbb{E}_{q(\boldsymbol{y}|\boldsymbol{x}^*)} \left[\log \frac{p_\theta(\boldsymbol{x}^*, \boldsymbol{y})}{q(\boldsymbol{y}|\boldsymbol{x}^*)} \right] \\ &= -\mathrm{H}\left(q(\boldsymbol{y}|\boldsymbol{x}^*)\right) - \mathbb{E}_{q(\boldsymbol{y}|\boldsymbol{x}^*)} \left[\log p_\theta(\boldsymbol{x}^*, \boldsymbol{y})\right] := \mathcal{L}(q, \boldsymbol{\theta}), \end{aligned} \tag{2.9}$$

where the inequality holds because KL divergence is always nonnegative. The free energy upper bound contains two terms: the first one is the entropy of the variational distribution, which captures the intrinsic randomness (i.e., amount of information carried by an auxiliary distribution); the second term, now written as $-\mathbb{E}_{q(\boldsymbol{y}|\boldsymbol{x}^*)\tilde{p}_d(\boldsymbol{x}^*)} [\log p_\theta(\boldsymbol{x}^*, \boldsymbol{y})]$, by taking into account the empirical distribution $\tilde{p}_d$ from which the instance $\boldsymbol{x}^*$ is drawn, is the *cross entropy* between the distributions $q(\boldsymbol{y}|\boldsymbol{x}^*)\tilde{p}_d(\boldsymbol{x}^*)$ and $p_\theta(\boldsymbol{x}^*, \boldsymbol{y})$, driving the two to be close and thereby allowing q to approximate p.

The popular expectation maximization (EM) algorithm for unsupervised learning via MLE can be interpreted as minimizing the variational free energy (Neal & Hinton, 1998). In fact, as we discuss subsequently, popular heuristics such as the variational EM and the wake-sleep algorithms, are approximations to the EM algorithm by introducing approximating realizations to either the free energy objective function $\mathcal{L}$ or to the solution space of the variational distribution q.

Expectation Maximization (EM). The most common approach to learning with unlabeled data or partially observed multivariate models is perhaps the EM algorithm (Dempster et al., 1977). With the use of the variational free energy as a surrogate objective to the original marginal likelihood as in Equation 2.9, EM can be also understood as an alternating minimization algorithm, where $\mathcal{L}(q, \boldsymbol{\theta})$ is minimized with regard to q and $\boldsymbol{\theta}$ in two stages, respectively. At each iteration n, the expectation (E) step maximizes $\mathcal{L}(q, \boldsymbol{\theta}^{(n)})$ w.r.t. q. From Equation 2.9, this is achieved by setting q to the current true posterior:

$$\text{E-step:} \quad q^{(n+1)}(\boldsymbol{y}|\boldsymbol{x}^*) = p_{\theta^{(n)}}(\boldsymbol{y}|\boldsymbol{x}^*), \tag{2.10}$$

so that the KL divergence vanishes and the upper bound is tight. In the subsequent maximization (M) step, $\mathcal{L}(q^{(n+1)}, \boldsymbol{\theta})$ is minimized w.r.t. $\boldsymbol{\theta}$:

$$\text{M-step:} \quad \max_{\boldsymbol{\theta}} \mathbb{E}_{q^{(n+1)}(\boldsymbol{y}|\boldsymbol{x}^*)} \left[\log p_\theta(\boldsymbol{x}^*, \boldsymbol{y})\right], \tag{2.11}$$

which is to maximize the expected complete data log-likelihood. The EM algorithm has an appealing property that it monotonically decreases the negative marginal log-likelihood over iterations. To see this, notice that after the E-step the upper bound $\mathcal{L}(q^{(n+1)}, \boldsymbol{\theta}^{(n)})$ is equal to the negative marginal

log-likelihood, and the M-step further decreases the upper bound (and thus the negative marginal log-likelihood).

Variational EM. When the model $p_\theta(\boldsymbol{x}, \boldsymbol{y})$ is complex (e.g., a neural network or a multilayer graphical model), directly working with the true posterior in the E-step becomes intractable. Variational EM overcomes the difficulty with approximations. It considers a restricted family $\mathcal{Q}'$ of the variational distribution $q(\boldsymbol{y})$ such that optimization w.r.t. q within the family is tractable:

$$\text{Variational E-step:} \quad \min_{q \in \mathcal{Q}'} \mathcal{L}(q, \boldsymbol{\theta}^{(t)}). \tag{2.12}$$

A common way to restrict the q family is the *mean-field methods*, which partition the components of $\boldsymbol{y}$ into sub-groups $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M)$ and assume that q factorizes w.r.t. the groups: $q(\boldsymbol{y}) = \prod_{i=1}^{M} q_i(\boldsymbol{y}_i)$. The variational principle summarized in (Wainwright & Jordan, 2008) gives a more principled interpretation of the mean-field and other approximation methods. In particular, in the case where $p_\theta(\boldsymbol{x}, \boldsymbol{y})$ is an exponential family distribution with sufficient statistics $T(\boldsymbol{x}, \boldsymbol{y})$, the exact E-step (Equation 2.10) can be interpreted as seeking the optimal valid *mean parameters* (i.e., expected sufficient statistics) for which the free energy is minimized. For discrete latent variables $\boldsymbol{y}$, the set of all valid mean parameters constitutes a marginal polytope $\mathcal{M}$. In this perspective, the mean-field methods (Equation 2.12) correspond to replacing $\mathcal{M}$ with an *inner* approximation $\mathcal{M}' \subseteq \mathcal{M}$. With the restricted set $\mathcal{M}'$ of mean parameters, the E-step generally no longer tightens the bound of the negative marginal log-likelihood, and the algorithm does not necessarily decrease the negative marginal log-likelihood monotonically. However, the algorithm preserves the property that it minimizes the upper bound of the negative marginal log-likelihood. Besides the mean-field methods, there are other approaches for approximation such as belief propagation. These methods correspond to using an *outer* approximation $\mathcal{M}'' \supseteq \mathcal{M}$ of the marginal polytope, and do not guarantee upper bounds on the negative marginal log-likelihood.

Another approach to restrict the family of q is to assume a parametric distribution $q_\omega(\boldsymbol{y}|\boldsymbol{x})$ and optimize the parameters $\boldsymbol{\omega}$ in the E-step. The approach has been used in black-box variational inference (Ranganath et al., 2014), and variational auto-encoders (VAEs) (Kingma & Welling, 2014) where q is parameterized as a neural network (a.k.a 'inference network,' or 'encoder').

It is worth mentioning that the variational approach has also been used for approximate Gaussian processes (GPs, as a nonparametric methods) (Titsias, 2009; Wilson, Hu, Salakhutdinov, et al., 2016), where $\boldsymbol{y}$ is the inducing points and the variational distribution $q(\boldsymbol{y})$ is parameterized as a Gaussian distribution with a nondiagonal covariance matrix that preserves the structures within the true covariance (and hence is different from the above mean-field approximation, which assumes a diagonal variational covariance matrix). We refer interested readers to (Wilson, Hu, Salakhutdinov, et al., 2016) for more details.

Wake-Sleep. In some cases when the auxiliary q is assumed to have a certain form (e.g., a deep network), the approximate E-step in Equation 2.12 may still be too complex to be tractable, or the gradient estimator (w.r.t. the parameters of q) can suffer from high variance (Mnih & Gregor, 2014; Paisley et al., 2012). To tackle the challenge, more approximations are introduced. The wake-sleep algorithm (G. E. Hinton et al., 1995) is one of such methods. In the E-step w.r.t. q, rather than minimizing $\mathrm{KL}(q(\boldsymbol{y}) \| p_\theta(\boldsymbol{y}|\boldsymbol{x}^*))$ (Equation 2.9) as in EM and variational EM, the wake-sleep algorithm makes an approximation by minimizing the Kullback–Leibler (KL) divergence in

opposite direction:

$$\text{Approximate E-step (Sleep-phase):} \quad \min_{q \in \mathcal{Q}'} \mathrm{KL}\left(p_\theta(\boldsymbol{y}|\boldsymbol{x}^*) \| q(\boldsymbol{y})\right), \tag{2.13}$$

which can be optimized efficiently with gradient descent when q is parameterized. Besides wake-sleep, one can also use other methods for low-variance gradient estimation in Equation 2.12, such as reparameterization gradient (Kingma & Welling, 2014) and score gradient (Glynn, 1990; Mnih & Gregor, 2014; Ranganath et al., 2014).

In sum, the entropy maximization perspective has formulated unsupervised MLE as an optimization-theoretic framework that permits simple alternating minimization solvers. Starting from the upper bound of negative marginal log-likelihood (Equation 2.9) with maximum entropy and minimum cross entropy, the originally intractable MLE problem gets simplified, and a series of optimization algorithms, ranging from (variational) EM to wake-sleep, arise naturally as an approximation to the original solution.

2.2. **Bayesian Inference.** Now we revisit another classical learning framework, Bayesian inference, and examine its intriguing connections with the maximum entropy principle. Interestingly, the the maximum entropy principle can also help to reformulate Bayesian inference as a constraint optimization problem, as for MLE.

Different from MLE, Bayesian approach for statistical inference treats the hypotheses (parameters $\boldsymbol{\theta}$) to be inferred as random variables. Assuming a prior distribution $\pi(\boldsymbol{\theta})$ over the parameters, and considering a probabilistic model that defines a conditional distribution $p(\boldsymbol{x}|\boldsymbol{\theta})$, the inference is based on the Bayes' theorem:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{\pi(\boldsymbol{\theta}) \prod_{\boldsymbol{x}^* \in \mathcal{D}} p(\boldsymbol{x}^*|\boldsymbol{\theta})}{p(\mathcal{D})}, \tag{2.14}$$

where $p(\boldsymbol{\theta}|\mathcal{D})$ is the posterior distribution after observing the data $\mathcal{D}$ (which we assume are i.i.d.); and $p(\mathcal{D}) = \int_{\boldsymbol{\theta}} \pi(\boldsymbol{\theta}) \prod_{\boldsymbol{x}^*} p(\boldsymbol{x}^*|\boldsymbol{\theta}) d\boldsymbol{\theta}$ is the marginal likelihood.

Interestingly, the early work by Zellner (1988) showed the relations between Bayesian inference and maximum entropy, by reformulating the statistical inference problem from the perspective of information processing, and rediscovering the Bayes' theorem as the optimal information processing rule. More specifically, statistical inference can be seen as a procedure of information processing, where the system receives input information in the form of prior knowledge and data, and emits output information in the form of parameter estimates and others. An efficient inference procedure should generate an output distribution such that the system retains all input information and not inject any extraneous information. The learning objective is thus to minimize the difference between the input and output information w.r.t. the output distribution:

$$\begin{aligned} &\min_{q(\boldsymbol{\theta})} \; -\mathrm{H}(q(\boldsymbol{\theta})) + \log p(\mathcal{D}) - \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log \pi(\boldsymbol{\theta}) + \sum\nolimits_{\boldsymbol{x}^* \in \mathcal{D}} \log p(\boldsymbol{x}^*|\boldsymbol{\theta})\right] \\ &s.t.\; q(\boldsymbol{\theta}) \in \mathcal{P}(\boldsymbol{\Theta}), \end{aligned} \tag{2.15}$$

where the first two terms measure the output information in the output distribution $q(\boldsymbol{\theta})$ and marginal $p(\mathcal{D})$, and the third term measures the input information in the prior $\pi(\boldsymbol{\theta})$ and data likelihood $p(\boldsymbol{x}^*|\boldsymbol{\theta})$. Here $\mathcal{P}(\boldsymbol{\Theta})$ is the space of all probability distributions over $\boldsymbol{\theta}$.

The optimal solution of $q(\boldsymbol{\theta})$ is precisely the the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ due to the Bayes' theorem (Equation 2.14). The proof is straightforward by noticing that the objective can be rewritten as $\min_q \mathrm{KL}(q(\boldsymbol{\theta})\|p(\boldsymbol{\theta}|\mathcal{D}))$.

Similar to the case of duality between MLE and maximum entropy (Equation 2.4), the same entropy maximization principle can cast Bayesian inference as a constrained optimization problem. As Jaynes (1988) commented, this fresh interpretation of Bayes' theorem "could make the use of Bayesian methods more attractive and widespread, and stimulate new developments in the general theory of inference." (Jaynes, 1988, p.280) The next subsection reviews how entropy maximization as a "useful tool in generating probability distributions" (Jaynes, 1988, p.280) has related to and resulted in more general learning and inference frameworks, such as posterior regularization.

2.3. **Posterior Regularization.** The optimization-based formulation of Bayesian inference in Equation 2.15 offers important additional flexibility in learning by allowing rich constraints on machine learning models to be imposed to regularize the outcome. For example, in Equation 2.15 we have seen the standard normality constraint of a probability distribution being imposed on the posterior q. It is natural to consider other types of constraints that encode richer problem structures and domain knowledge, which can regularize the model to learn desired behaviors.

The idea has led to posterior regularization (PR, Ganchev et al., 2010) or regularized Bayes (RegBayes, J. Zhu et al., 2014), which augments the Bayesian inference objective with additional constraints:

$$\begin{aligned} \min_{q,\boldsymbol{\xi}} \; & -\mathrm{H}\left(q(\boldsymbol{\theta})\right) - \mathbb{E}_{q(\boldsymbol{\theta})}\left[\sum\nolimits_{\boldsymbol{x}^*\in\mathcal{D}} \log p(\boldsymbol{x}^*|\boldsymbol{\theta})\pi(\boldsymbol{\theta})\right] + U(\boldsymbol{\xi}) \\ s.t. \; & q(\boldsymbol{\theta}) \in \mathcal{Q}(\boldsymbol{\xi}) \\ & \boldsymbol{\xi} \geq 0, \end{aligned} \tag{2.16}$$

where we have rearranged the terms and dropped any constant factors in Equation 2.15, and added constraints with $\boldsymbol{\xi}$ being a vector of slack variables, $U(\boldsymbol{\xi})$ a penalty function (e.g., ℓ_1 norm of $\boldsymbol{\xi}$), and $\mathcal{Q}(\boldsymbol{\xi})$ a subset of valid distributions over $\boldsymbol{\theta}$ that satisfy the constraints determined by $\boldsymbol{\xi}$. The optimization problem is generally easy to solve when the penalty/constraints are convex and defined w.r.t. a linear operator (e.g., expectation) of the posterior q. For example, let $T(\boldsymbol{x}^*;\boldsymbol{\theta})$ be a feature vector of data instance $\boldsymbol{x}^* \in \mathcal{D}$, the constraint posterior set Q can be defined as:

$$Q(\boldsymbol{\xi}) := \{q(\boldsymbol{\theta}) \; : \; \mathbb{E}_q\left[T(\boldsymbol{x}^*;\boldsymbol{\theta})\right] \leq \boldsymbol{\xi}, \; \forall \boldsymbol{x}^* \in \mathcal{D}\}, \tag{2.17}$$

which bounds the feature expectations with $\boldsymbol{\xi}$.

Max-margin constraint is another expectation constraint that has shown to be widely effective in classification and regression (Vapnik, 1998). The maximum entropy discrimination (MED) by Jaakkola et al. (2000) regularizes linear regression models with the max-margin constraints, which is latter generalized to more complex models $p(\boldsymbol{x}|\boldsymbol{\theta})$, such as Markov networks (Taskar et al., 2004) and latent variable models (J. Zhu et al., 2014). Formally, let $\boldsymbol{y}^* \in \mathbb{R}$ be the observed label associated with $\boldsymbol{x}^*$. The margin-based constraint says that a classification/regression function $h(\boldsymbol{x};\boldsymbol{\theta})$ should make at most ϵ deviation from the true label $\boldsymbol{y}^*$. Specifically, consider the common choice of the function h as a linear function: $h(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{\theta}^\top T(\boldsymbol{x})$, where $T(\boldsymbol{x})$ is, with a slight abuse of notation,

the feature of instance $\boldsymbol{x}$. The constraint is written as:

$$\begin{cases} \boldsymbol{y}^* - \mathbb{E}_q\left[\boldsymbol{\theta}^\top T(\boldsymbol{x}^*)\right] \leq \epsilon + \xi \\ -\boldsymbol{y}^* + \mathbb{E}_q\left[\boldsymbol{\theta}^\top T(\boldsymbol{x}^*)\right] \leq \epsilon + \xi', \end{cases} \tag{2.18}$$

for all instances $(\boldsymbol{x}^*, \boldsymbol{y}^*) \in \mathcal{D}$.

Alternating optimization for posterior regularization. Having seen EM-style alternating minimization algorithms being applied as a general solver for a number of optimization-theoretic frameworks described above, it is not surprising that the posterior regularization framework can also be solved with an alternating minimization procedure. For example, consider the simple case of linear constraint in Equation 2.17, penalty function $U(\boldsymbol{\xi}) = \|\boldsymbol{\xi}\|_1$, and q factorizing across $\boldsymbol{\theta} = \{\boldsymbol{\theta}_c\}$. At each iteration n, the solution of $q(\boldsymbol{\theta}_c)$ is given as (Ganchev et al., 2010):

$$q^{(n+1)}(\boldsymbol{\theta}_c) = \exp\left\{\mathbb{E}_{q^{(n)}(\boldsymbol{\theta}_{\setminus c})} \sum_{\boldsymbol{x}^*} \log p(\boldsymbol{x}^*|\boldsymbol{\theta})\pi(\boldsymbol{\theta}) + T(\boldsymbol{x}^*;\boldsymbol{\theta})\right\}/Z, \tag{2.19}$$

where $\boldsymbol{\theta}_{\setminus c}$ denotes all components of $\boldsymbol{\theta}$ except $\boldsymbol{\theta}_c$, and Z is the normalization factor. Intuitively, a configuration of $\boldsymbol{\theta}_c$ with a higher expected constraint value $\mathbb{E}_{\setminus c} T(\boldsymbol{x}^*;\boldsymbol{\theta})$ will receive a higher probability under $q^{(n+1)}(\boldsymbol{\theta}_c)$. The optimization procedure iterates over all components c of $\boldsymbol{\theta}$.

2.4. **Summary.** In this section, we have seen that the maximum entropy formalism provides an alternative insight into the classical learning frameworks of MLE, Bayesian inference, and posterior regularization. It provides a general expression of these three paradigms as a constrained optimization problem, with a paradigm-specific loss on the model parameters $\boldsymbol{\theta}$ and an auxiliary distribution q, over a properly designed constraint space $\mathcal{Q}$ where q must reside:

$$\begin{gathered} \min_{q,\boldsymbol{\theta}} \mathcal{L}(q, \boldsymbol{\theta}) \\ s.t.\ q \in \mathcal{Q}. \end{gathered} \tag{2.20}$$

In particular, the use of the auxiliary distribution q converts the originally highly complex problem of directly optimizing $\boldsymbol{\theta}$ against data, to an alternating optimization problem over q and $\boldsymbol{\theta}$, which is algorithmically easier to solve since q often acts as an easy-to-optimize proxy to the target model. The auxiliary q can also be more flexibly updated to absorb influence from data or constraints, offering a teacher-student–style iterative mechanism to incrementally update $\boldsymbol{\theta}$ as we will see in the sequel.

By reformulating learning as a constrained optimization problem, the maximum entropy point of view also offers a great source of flexibility for applying many powerful tools for efficient approximation and enhanced learning, such as variational approximation (e.g., by relaxing $\mathcal{Q}$ to be easy-to-inference family of q such as the mean field family, Jordan et al. (1999) and Xing et al. (2002)), convex duality (e.g., facilitating dual sparsity of support vectors via the complementary slackness in the KKT conditions), and kernel methods as used in (Taskar et al., 2004; J. Zhu & Xing, 2009).

It is intriguing that, in the dual point of view on the problem of (supervised) MLE, data instances are encoded as constraints (Equation 2.4), much like the structured constraints in posterior regularization. In the following sections, we present the standardized formalism of machine learning algorithms and show that indeed a myriad types of experience besides data instances and constraints can all be encoded in the same generic form and be used in learning.

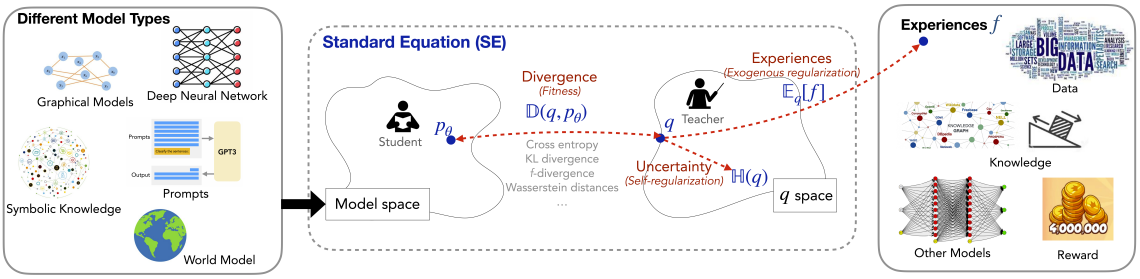


Figure 1. An illustration of the standard equation (SE) as a general formulation of the objective function, used to learn an arbitrary target model p_θ with any forms of experience. All different forms of experience are formulated uniformly as an experience function f. The SE contains three terms, including the experience function for incorporating the exogenous information, the divergence function for improving the fitness of the target model by matching with the auxiliary 'teacher' model q, and the uncertainty function for controlling the complexity of the learning system.

3. A Standard Model for Objective Function

Generalizing from Equation 2.16, we present the following general formulation for learning a target model via a constrained loss minimization program. We would refer to the formulation as the 'Standard Equation' because it presents a general space of learning objectives that encompasses many specific formalisms used in different machine learning paradigms.

Without loss of generality, let $\boldsymbol{t} \in \mathcal{T}$ be the variable of interest, for example, the input-output pair $\boldsymbol{t} = (\boldsymbol{x}, \boldsymbol{y})$ in a prediction task, or the target variable $\boldsymbol{t} = \boldsymbol{x}$ in generative modeling. Let $p_\theta(\boldsymbol{t})$ be the target model with parameters $\boldsymbol{\theta}$ to be learned. Generally, the SE is agnostic to the specific forms of the target model, meaning that the target model can take an arbitrary form as desired by the problem at hand (e.g., classification, regression, generation, control) and can be of arbitrary types ranging from deep neural networks of arbitrary architectures, prompts for pretrained models, symbolic systems (e.g., knowledge graph), probabilistic graphical models of arbitrary dependence structures, and so on. We discuss more details of the different choices of the target model in Section 8.

Let $q(\boldsymbol{t})$ be an auxiliary distribution. The SE is written as:

$$\begin{aligned} &\min_{q,\boldsymbol{\theta},\boldsymbol{\xi}} \; -\alpha \mathbb{H}\left(q\right) + \beta \mathbb{D}\left(q, p_\theta\right) + U(\boldsymbol{\xi}) \\ &s.t. \; -\mathbb{E}_q\left[f_k^{(\theta)}\right] \leq \xi_k, \; k = 1, \ldots, K. \end{aligned} \tag{3.1}$$

The SE contains three major terms that constitute a learning formalism: the *uncertainty function* $\mathbb{H}\left(\cdot\right)$ that controls the compactness of the output model (e.g., as measured by the amount of allowed randomness while trying to fit experience); the *divergence function* $\mathbb{D}\left(\cdot, \cdot\right)$ that measures the distance between the target model to be trained and the auxiliary model that facilitates a teacher–student mechanism as shown below; and the *experience function*, which is introduced by a penalty term $U(\boldsymbol{\xi})$ that draws in the set of 'experience functions' $f_k^{(\theta)}$ that represent external experience of various kinds for training the target model. The hyperparameters $\alpha, \beta \geq 0$ enable trade-offs between these components.

Experience function. Perhaps the most powerful in terms of impacting the learning outcome and utility is the experience functions $f_k^{(\theta)}$. An experience function $f^{(\theta)}(\boldsymbol{t}) \in \mathbb{R}$ measures the goodness of a configuration $\boldsymbol{t}$ in light of any given experience. The superscript (θ) highlights that the experience in some settings (e.g., reward experience as in Section 4.3) could depend on or be coupled with the target model parameters $\boldsymbol{\theta}$. In the following, we omit the superscript when there is no ambiguity. As discussed in Section 4, all diverse forms of experience that can be utilized for model training, such as data examples, constraints, logical rules, rewards, and adversarial discriminators, can be encoded as an experience function. The experience function hence provides a unified language to express all exogenous information about the target model, which we consider as an essential ingredient for panoramic learning to flexibly incorporate diverse experience in learning. Based on the uniform treatment of experience, a standardized optimization program as above can be formulated to identify the desired model. Specifically, the experience functions contribute to the optimization objective via the penalty term $U(\boldsymbol{\xi})$ over slack variables $\boldsymbol{\xi} \in \mathbb{R}^K$ applied to the expectation $\mathbb{E}_q[f_k]$. The effect of maximizing the expectation is such that the auxiliary model q is encouraged to produce samples of high quality in light of the experience (i.e., samples receiving high scores as evaluated by the experience function).

Divergence function. The divergence function $\mathbb{D}(q, p_\theta)$ measures the 'quality' of the target model p_θ in terms of its distance (divergence) with the auxiliary model q. Intuitively, we want to minimize the distance from p_θ to q, which is optimized to fit the experience as above. Section 5 gives a concrete example of how the divergence term would directly impact the model training: with a certain specification of the different components (e.g., the experience function, α/β), the SE in Equation 3.1 would reduce to $\min_{\boldsymbol{\theta}} \mathbb{D}(p_d, p_\theta)$. That is, the learning objective is to minimize the divergence between the target model distribution p_θ and the data distribution p_d, and the divergence function $\mathbb{D}(\cdot, \cdot)$ determines the specific optimization problem. The divergence function can have a variety of choices, ranging from the family of f-divergence (e.g., KL divergence), or Bregman divergence, to optimal transport distance (e.g., Wasserstein distance), and so on. We discuss the divergence term in Section 5 in more detail.

Uncertainty function. The uncertainty function $\mathbb{H}(q)$ describes the uncertainty of the auxiliary distribution q and thus controls the complexity of the learning system. It conforms with the maximum entropy principle discussed in Section 2 that one should pick the most uncertain solution among those that fit all experience. Like other components in SE, the uncertainty measure $\mathbb{H}(\cdot)$ can take different forms, such as the popular Shannon entropy, as well as other generalized ones such as Tsallis entropy. In this article, we assume Shannon entropy by default.

For the discussion in the following sections, it is often convenient to consider a special case of the SE in Equation 3.1. Specifically, we assume a common choice of the penalty $U(\boldsymbol{\xi}) = \sum_k \xi_k$, and, with a slight abuse of notations, $f = \sum_k f_k$. In this case, the SE in Equation 3.1 can equivalently be written in an unconstrained form:

$$\min_{q, \boldsymbol{\theta}} \; -\alpha \mathbb{H}(q) + \beta \mathbb{D}(q, p_\theta) - \mathbb{E}_q[f], \tag{3.2}$$

which can be easily seen by optimizing Equation 3.1 over $\boldsymbol{\xi}$. In the special unconstrained form, the interplay between the exogenous experience, divergence, and the endogenous uncertainty become more explicit.

Optimization: Teacher-student mechanism. The introduction of the auxiliary distribution q relaxes the learning problem of p_θ, originally only over $\boldsymbol{\theta}$, to be now alternating between q and $\boldsymbol{\theta}$. Here q acts as a conduit between the exogenous experience and the target model: it on the one hand subsumes the experience (by maximizing the expected f value), and on the other hand passes it incrementally to the target model (by minimizing the divergence $\mathbb{D}$). The following fixed point iteration between q and $\boldsymbol{\theta}$ illustrates this optimization strategy under the SE. Let us plug into Equation 3.2 the popular cross entropy (CE) as the divergence function, that is, $\mathbb{D}(q, p_\theta) = -\mathbb{E}_q[\log p_\theta]$, and Shannon entropy as the uncertainty measure, that is, $\mathbb{H}(q) = -\mathbb{E}_q[\log q]$. We further assume the experience f is independent of the model parameters $\boldsymbol{\theta}$ (the assumption is indeed not necessary for the teacher step). We have, at iteration n:

$$\begin{aligned} &\text{Teacher:} \quad q^{(n+1)}(\boldsymbol{t}) = \exp\left\{\frac{\beta \log p_{\theta^{(n)}}(\boldsymbol{t}) + f(\boldsymbol{t})}{\alpha}\right\} \;/\; Z \\ &\text{Student:} \quad \boldsymbol{\theta}^{(n+1)} = \operatorname*{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{q^{(n+1)}(\boldsymbol{t})}\big[\log p_\theta(\boldsymbol{t})\big], \end{aligned} \tag{3.3}$$

where Z is the normalization factor. The first step embodies a 'teacher's update' where the teacher q ingests experience f and builds on current states of the student $p_{\theta^{(n)}}$; the second step is reminiscent of a 'student's update' where the student p_θ updates its states by maximizing its alignment (here measured by CE) with the teacher.

Besides, the auxiliary q is an easy-to-manipulate intermediate form in the training that permits rich approximate inference tools for tractable optimization. We have the flexibility of choosing its surrogate functions, ranging from the principled variational approximations for the target distribution in a properly relaxed space (e.g., mean fields) where gaps and bounds can be characterized, to the arbitrary neural network-based 'inference networks' that are highly expressive and easy to compute. As can be easily shown (e.g., see Section 4.1.3), popular training heuristics, such as EM, variational EM, wake-sleep, forward and backward propagation, and so on, are all direct instantiations or variants of the above teacher-student mechanism with different choices of the form of q.

More generally, a broad set of sophisticated algorithms, such as the policy gradient for reinforcement learning and the generative adversarial learning, can also be easily derived by plugging in specific designs of the experience function f and divergence $\mathbb{D}$. Table 1 summarizes various specifications of the SE components that recover a range of existing well-known algorithms from different paradigms. As shown in more detail in the subsequent sections, the standard equation (Equation 3.1 and 3.2) offers a unified and universal paradigm for model training under many scenarios based on many types of experience, potentiating a turnkey implementation and a more generalizable theoretical characterization.

4. Experience Function

The experience function $f(\boldsymbol{t})$ in the standard equation can be instantiated to encode vastly distinct types of experience. Different choices of $f(\boldsymbol{t})$ result in learning algorithms applied to different problems. With particular choices, the standard equation rediscovers a wide array of well-known algorithms. The resulting common treatment of the previously disparate algorithms is appealing as it offers new holistic insights into the commonalities and differences of those algorithms. Table 1 shows examples of extant algorithms that are recovered by the standard equation.

Experience type	Experience function f	Divergence $\mathbb{D}$	α	β	Algorithm
Data instances	$f_{\text{data}}(\boldsymbol{x};\mathcal{D})$	CE	1	1	Unsupervised MLE
	$f_{\text{data}}(\boldsymbol{x},\boldsymbol{y};\mathcal{D})$	CE	1	ϵ	Supervised MLE
	$f_{\text{data-self}}(\boldsymbol{x},\boldsymbol{y};\mathcal{D})$	CE	1	ϵ	Self-supervised MLE
	$f_{\text{data-w}}(\boldsymbol{t};\mathcal{D})$	CE	1	ϵ	Data Re-weighting
	$f_{\text{data-aug}}(\boldsymbol{t};\mathcal{D})$	CE	1	ϵ	Data Augmentation
	$f_{\text{active}}(\boldsymbol{x},\boldsymbol{y};\mathcal{D})$	CE	1	ϵ	Active Learning (Ertekin et al., 2007)
Knowledge	$f_{rule}(\boldsymbol{x},\boldsymbol{y})$	CE	1	1	Posterior Regularization (Ganchev et al., 2010)
	$f_{rule}(\boldsymbol{x},\boldsymbol{y})$	CE	$\mathbb{R}$	1	Unified EM (Samdani et al., 2012)
Reward	$\log Q^{\theta}(\boldsymbol{x},\boldsymbol{y})$	CE	1	1	Policy Gradient
	$\log Q^{\theta}(\boldsymbol{x},\boldsymbol{y}) + Q^{in,\theta}(\boldsymbol{x},\boldsymbol{y})$	CE	1	1	+ Intrinsic Reward
	$Q^{\theta}(\boldsymbol{x},\boldsymbol{y})$	CE	$\rho > 0$	$\rho > 0$	RL as Inference
Model	$f^{\text{mimicking}}_{\text{model}}(\boldsymbol{x},\boldsymbol{y};\mathcal{D})$	CE	1	ϵ	Knowledge Distillation (G. Hinton et al., 2015)
Variational	binary classifier	JSD	0	1	Vanilla GAN (Goodfellow et al., 2014)
	discriminator	f-divergence	0	1	f-GAN (Nowozin et al., 2016)
	1-Lipschitz discriminator	W_1 distance	0	1	WGAN (Arjovsky et al., 2017)
	1-Lipschitz discriminator	KL	0	1	PPO-GAN (Y. Wu et al., 2020)
Online	$f_{\tau}(\boldsymbol{t})$	CE	$\rho > 0$	$\rho > 0$	Multiplicative Weights (Freund & Schapire, 1997)

Table 1. Example configurations of the components in the standard equation (Eqs.3.1, 3.2), which recover different existing algorithms. Here, 'CE' means Cross Entropy; 'JSD' is the Jensen-Shannon divergence; 'W_1 dist.' is the first-order Wasserstein distance; and 'KL' is the KL divergence. Refer to Sections 4, 5, and 6 for more details.

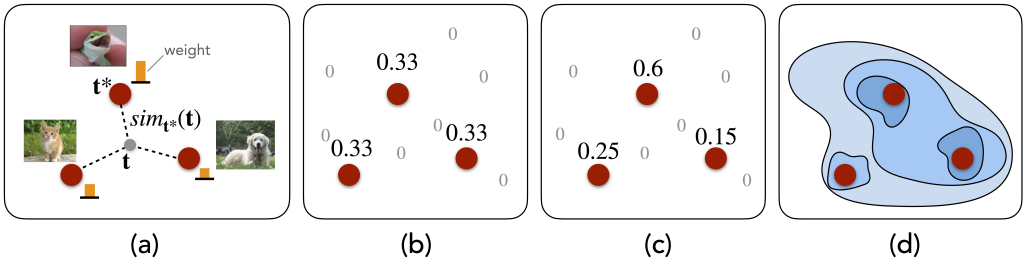


Figure 2. Illustration of the data instance experience: **(a)** The general view of the data instance experience function. An observed data set $\mathcal{D} = \{\boldsymbol{t}^*\}$ (assuming three data instances in the figure) is given. For a configuration $\boldsymbol{t} \in \mathcal{T}$, its experience function value $f(\boldsymbol{t})$ depends on its similarity with the observed data instances $\boldsymbol{t}^*$. Each observed data instance may also be associated with a weight. **(b)** With the indicator function $\mathbb{I}_{\boldsymbol{t}^*}(\boldsymbol{t})$ as the similarity metric and the default uniform weight of each observed $\boldsymbol{t}^*$ (e.g., Equation 4.2), the distribution of the experience function value matches the empirical data distribution (i.e., uniform over the observed instances, and 0 on all other configurations in $\mathcal{T}$). **(c)** By associating different observed instances $\boldsymbol{t}^*$ with different weights (Equation 4.6, data reweighting), the experience function corresponds to the reweighted empirical distribution. **(d)** By setting the similarity metric to a soft version (Equation 4.8, data augmentation), the experience function corresponds to a distribution over $\mathcal{T}$ that is more smooth, with nonzero probability on configurations other than the observed data instances.

4.1. **Data Instance Experience.** We first consider the most common type of experience, namely, data instances, which are assumed to be independent and identically distributed (i.i.d.). Such data instance experience can appear in a wide range of contexts, including supervised, self-supervised, unsupervised, actively supervised, and other scenarios with data augmentation and manipulation. Figure 2 illustrates the experience functions based on the data instances.

4.1.1. ***Supervised data instances.*** Without loss of generality and for consistency of notations with the rest of the section, we consider data instances to consist of a pair of input-output variables, namely $\boldsymbol{t} = (\boldsymbol{x}, \boldsymbol{y})$. For example, in image classification, $\boldsymbol{x}$ represents the input image and $\boldsymbol{y}$ is the object label. In the supervised setting, we observe the full data drawn i.i.d. from the data distribution $(\boldsymbol{x}^*, \boldsymbol{y}^*) \sim p_d(\boldsymbol{x}, \boldsymbol{y})$. For an arbitrary configuration $(\boldsymbol{x}_0, \boldsymbol{y}_0)$, its probability $p_d(\boldsymbol{x}_0, \boldsymbol{y}_0)$ under the data distribution can be seen as measuring the *expected similarity* between $(\boldsymbol{x}_0, \boldsymbol{y}_0)$ and true data samples $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, and be written as $p_d(\boldsymbol{x}_0, \boldsymbol{y}_0) = \mathbb{E}_{p_d(\boldsymbol{x}^*, \boldsymbol{y}^*)}\left[\mathbb{I}_{(\boldsymbol{x}^*, \boldsymbol{y}^*)}(\boldsymbol{x}_0, \boldsymbol{y}_0)\right]$. Here the similarity measure is $\mathbb{I}_{(\boldsymbol{x}^*, \boldsymbol{y}^*)}(\boldsymbol{x}, \boldsymbol{y})$, an indicator function that takes the value 1 if $(\boldsymbol{x}, \boldsymbol{y})$ equals $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ and 0 otherwise (we will see other similarity measures shortly). In practice, we are given an empirical distribution $\tilde{p}_d(\boldsymbol{x}, \boldsymbol{y})$ by observing a collection of instances $\mathcal{D}$ on which the expected similarity is evaluated:

$$\mathbb{E}_{(\boldsymbol{x}^*, \boldsymbol{y}^*) \sim \mathcal{D}}\left[\mathbb{I}_{(\boldsymbol{x}^*, \boldsymbol{y}^*)}(\boldsymbol{x}, \boldsymbol{y})\right] = \frac{m(\boldsymbol{x}, \boldsymbol{y})}{N}, \tag{4.1}$$

where N is the size of the data set $\mathcal{D}$, and $m(\boldsymbol{x}, \boldsymbol{y})$ is the number of occurrences of the configuration $(\boldsymbol{x}, \boldsymbol{y})$ in $\mathcal{D}$.

The experience function f accommodates the data instance experience straightforwardly as below:

$$f := f_{\text{data}}(\boldsymbol{x}, \boldsymbol{y}; \mathcal{D}) = \log \mathbb{E}_{(\boldsymbol{x}^*, \boldsymbol{y}^*) \sim \mathcal{D}}\left[\mathbb{I}_{(\boldsymbol{x}^*, \boldsymbol{y}^*)}(\boldsymbol{x}, \boldsymbol{y})\right]. \tag{4.2}$$

Figure 2 (a)-(b) shows an illustration. In particular, the logarithm of the expected similarity is used as the experience function score, that is, the more 'similar' a configuration $(\boldsymbol{x}, \boldsymbol{y})$ is to the observed data instances, the higher its quality. The logarithm serves to make the subsequent derivations more convenient as can be seen below.

With this from of f, we show that the SE derives the conventional supervised MLE algorithm.

Supervised MLE. In the SE Equation 3.2 (with cross entropy and Shannon entropy), we set $\alpha = 1$, and β to a very small positive value ϵ. As a result, the auxiliary distribution $q(\boldsymbol{x}, \boldsymbol{y})$ is determined directly by the full data instances (not the model p_θ). That is, the solution of q in the teacher-step (Equation 3.3) is:

$$q(\boldsymbol{x}, \boldsymbol{y}) = \exp\left\{\frac{\beta \log p_\theta(\boldsymbol{x}, \boldsymbol{y}) + f_{\text{data}}(\boldsymbol{x}, \boldsymbol{y}; \mathcal{D})}{\alpha}\right\} / Z \approx \exp\left\{f_{\text{data}}(\boldsymbol{x}, \boldsymbol{y}; \mathcal{D})\right\} / Z = \tilde{p}_d(\boldsymbol{x}, \boldsymbol{y}), \quad (4.3)$$

which reduces to the empirical distribution. The subsequent student-step that maximizes the log-likelihood of samples from q then leads to the supervised MLE updates w.r.t. $\boldsymbol{\theta}$.

4.1.2. ***Self-supervised data instances.*** Given an observed data instance $\boldsymbol{t}^* \in \mathcal{D}$ in general, one could potentially derive various supervision signals based on the structures of the data and the target model. In particular, one could apply a "split" function that artificially partitions $\boldsymbol{t}^*$ into two parts $(\boldsymbol{x}^*, \boldsymbol{y}^*) = \mathit{split}(\boldsymbol{t}^*)$ in different, sometimes stochastic ways. Then the two parts are treated as the input and output for the properly designed target model $p_\theta(\boldsymbol{x}, \boldsymbol{y})$ for supervised MLE as above, by plugging in the slightly altered experience function:

$$f := f_{\text{data-self}}(\boldsymbol{x}, \boldsymbol{y}; \mathcal{D}) = \log \mathbb{E}_{\boldsymbol{t}^* \sim \mathcal{D},\ (\boldsymbol{x}^*, \boldsymbol{y}^*) = \mathit{split}(\boldsymbol{t}^*)} \left[\mathbb{I}_{(\boldsymbol{x}^*, \boldsymbol{y}^*)}(\boldsymbol{x}, \boldsymbol{y})\right]. \quad (4.4)$$

A key difference from the above standard supervised learning setting is that now the target variable $\boldsymbol{y}$ is not costly obtained labels or annotations, but rather part of the massively available data instances. The paradigm of treating part of observed instance as the prediction target is called 'self-supervised' learning (e.g., Lecun & Misra, 2021) and has achieved great success in language and vision modeling. For example, in language modeling (Brown et al., 2020; Devlin et al., 2019), the instance $\boldsymbol{t}$ is a piece of text, and the 'split' function usually selects from $\boldsymbol{t}$ one or few words to be the target $\boldsymbol{y}$ and the remaining words to be $\boldsymbol{x}$.

4.1.3. ***Unsupervised data instances.*** In the unsupervised setting, for each instance $\boldsymbol{t} = (\boldsymbol{x}, \boldsymbol{y})$, such as (image, cluster index), we only observe the $\boldsymbol{x}$ part. That is, we are given a data set $\mathcal{D} = \{\boldsymbol{x}^*\}$ without the associated $\boldsymbol{y}^*$. The data set defines the empirical distribution $\tilde{p}_d(\boldsymbol{x})$. The experience can be encoded in the same form as the supervised data (Equation 4.2) but now with only the information of $\boldsymbol{x}^*$:

$$f := f_{\text{data}}(\boldsymbol{x}; \mathcal{D}) = \log \mathbb{E}_{\boldsymbol{x}^* \sim \mathcal{D}} \left[\mathbb{I}_{\boldsymbol{x}^*}(\boldsymbol{x})\right]. \quad (4.5)$$

Applying the SE to this setting with proper specifications derives the unsupervised MLE algorithm.

Unsupervised MLE. The form of Equation 3.2 is reminiscent of the variational free energy objective in the standard EM for unsupervised MLE (Equation 2.9). We can indeed get exact correspondence by setting $\alpha = \beta = 1$, and setting the auxiliary distribution $q(\boldsymbol{x}, \boldsymbol{y}) = \tilde{p}_d(\boldsymbol{x}) q(\boldsymbol{y}|\boldsymbol{x})$. The reason for $\beta = 1$, which differs from the specification $\beta = \epsilon$ in the supervised setting, is that the auxiliary distribution q cannot be determined fully by the unsupervised 'incomplete' data

experience alone. Instead, it additionally relies on p_θ through the divergence term. Here q is assumed a specialized decomposition $q(\boldsymbol{x}, \boldsymbol{y}) = \tilde{p}_d(\boldsymbol{x})q(\boldsymbol{y}|\boldsymbol{x})$ where $\tilde{p}_d(\boldsymbol{x})$ is fixed and thus not influenced by p_θ. In contrast, if no structure of q is assumed, we could potentially obtain an extended, *instance-weighted* version of EM where each instance $\boldsymbol{x}^*$ is weighted by the marginal likelihood $p_\theta(\boldsymbol{x}^*)$, in line with the previous weighted EM methods for robust clustering (e.g., Gebru et al., 2016; Yu et al., 2011).

4.1.4. ***Manipulated data instances.*** Data manipulation, such as reweighting data instances or augmenting an existing data set with new instances, is often a crucial step for efficient learning, such as in a low data regime or in presence of low-quality data sets (e.g., imbalanced labels). We show that the rich data manipulation schemes can be treated as experience and be naturally encoded in the experience function (Hu, Tan, et al., 2019). This is done by extending the data-instance experience function (Equation 4.2), in particular by enriching the similarity metric in different ways. The discussion here generally applies to data instance $\boldsymbol{t}$ of any structures, for example, $\boldsymbol{t} = (\boldsymbol{x}, \boldsymbol{y})$ or $\boldsymbol{t} = \boldsymbol{x}$.

Data reweighting. Rather than assuming the same importance of all data instances, we can associate each instance $\boldsymbol{t}^*$ with an importance weight $w(\boldsymbol{t}^*) \in \mathbb{R}$, so that the learning pays more attention to those high-quality instances, while low-quality ones (e.g., with noisy labels) are downplayed. This can be done by scaling the above 0/1 indicator function (e.g., Equation 4.2) with the weight (Figure 2[c]):

$$f := f_{\text{data-w}}(\boldsymbol{t}; \mathcal{D}) = \log \mathbb{E}_{\boldsymbol{t}^* \sim \mathcal{D}} \left[w(\boldsymbol{t}^*) \cdot \mathbb{I}_{\boldsymbol{t}^*}(\boldsymbol{t}) \right]. \tag{4.6}$$

Plugging $f_{\text{data-w}}$ into the SE (Equation 3.2) with the same other specification of supervised MLE ($\alpha = 1, \beta = \epsilon$), we get the update rule of model parameters $\boldsymbol{\theta}$ in the student-step (Equation 3.3):

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{t}^* \sim \mathcal{D}} \left[w(\boldsymbol{t}^*) \cdot \log p_\theta(\boldsymbol{t}^*) \right], \tag{4.7}$$

which is the familiar weighted supervised MLE. The weights w can be specified a priori based on heuristics, for example, using inverse class frequency. In many cases it is desirable to automatically induce and adapt the weights during the course of model training. In Section 9.2, we discuss how the SE framework can easily enable automated data reweighting by reusing existing algorithms that were designed to solve other seemingly unrelated problems.

Data augmentation. Data augmentation expands existing data by adding synthetically modified copies of existing data instances (e.g., by rotating an existing image at random angles), and is widely used for increasing data size or encouraging invariance in learned representations (e.g., object label is invariant to image rotation). The indicator function $\mathbb{I}$ as the similarity metric in Equation 4.2 restrictively requires exact match between the true $\boldsymbol{t}^*$ and the configuration $\boldsymbol{t}$. Data augmentation arises as a 'relaxation' to the similarity metric. Let $a_{\boldsymbol{t}^*}(\boldsymbol{t}) \geq 0$ be a distribution that assigns non-zero probability to not only the exact $\boldsymbol{t}^*$ but also other configurations $\boldsymbol{t}$ related to $\boldsymbol{t}^*$ in certain ways (e.g., all rotated images $\boldsymbol{t}$ of the observed image $\boldsymbol{t}^*$). Replacing the indicator function metric in Equation 4.2 with the new $a_{\boldsymbol{t}^*}(\boldsymbol{t}) \geq 0$ yields the experience function for data augmentation (Figure 2[d]):

$$f := f_{\text{data-aug}}(\boldsymbol{t}; \mathcal{D}) = \log \mathbb{E}_{\boldsymbol{t}^* \sim \mathcal{D}} \left[a_{\boldsymbol{t}^*}(\boldsymbol{t}) \right]. \tag{4.8}$$

The resulting student-step updates of $\boldsymbol{\theta}$, keeping $(\alpha = 1, \beta = \epsilon)$ of supervised MLE, is thus:

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{t}^* \sim \mathcal{D},\ \boldsymbol{t} \sim a_{\boldsymbol{t}^*}(\boldsymbol{t})} \left[\log p_\theta(\boldsymbol{t})\right]. \tag{4.9}$$

The metric $a_{\boldsymbol{t}^*}(\boldsymbol{t})$ can be defined in various ways, leading to different augmentation strategies. For example, setting $a_{\boldsymbol{t}^*}(\boldsymbol{t}) \propto \exp\{R(\boldsymbol{t}, \boldsymbol{t}^*)\}$, where $R(\boldsymbol{t}, \boldsymbol{t}^*)$ is a task-specific evaluation metric such as BLEU for machine translation, results in the reward-augmented maximum likelihood (RAML) algorithm (Norouzi et al., 2016). Besides the manually designed strategies, we can also specify $a_{\boldsymbol{t}^*}(\boldsymbol{t})$ as a parameterized transformation process and learn any free parameters thereof automatically (Section 6). Notice the same form of the augmentation experience $f_{\text{data-aug}}$ and the reweighting experience $f_{\text{data-w}}$, where the similarity metrics both include learnable components (i.e., $a_{\boldsymbol{t}^*}(\boldsymbol{t})$ and $w(\boldsymbol{t}^*)$, respectively). Thus the same approach to automated data reweighting can also be applied for automated data augmentation, as discussed more in Section 9.2.

4.1.5. ***Actively supervised data instances.*** Instead of access to data instances $\boldsymbol{x}^*$ with readily available labels $\boldsymbol{y}^*$, in the active supervision setting, we are presented with a large pool of unlabeled instances $\mathcal{D} = \{\boldsymbol{x}^*\}$ as well as a certain budget for querying an oracle (e.g., human annotators) for labeling a limited set of instances. To minimize the need for labeled instances, we need to strategically select queries from the pool according to an *informativeness* measure $u(\boldsymbol{x}) \in \mathbb{R}$. For example, $u(\boldsymbol{x})$ can be the predictive uncertainty on the instance $\boldsymbol{x}$, quantified by the Shannon entropy of the predictive distribution or the vote entropy based on a committee of predictors (Dagan & Engelson, 1995).

Mapping the standard equation to this setting, we show the informativeness measure $u(\boldsymbol{x})$ is subsumed as part of the experience. Intuitively, $u(\boldsymbol{x})$ encodes our heuristic belief about sample 'informativeness'. This heuristic is a form of information we inject into the learning system. Denote the oracle as o from which we can draw a label $\boldsymbol{y}^* \sim o(\boldsymbol{x}^*)$. The active supervision experience function is then defined as:

$$f := f_{\text{active}}(\boldsymbol{x}, \boldsymbol{y}; \mathcal{D}) = \log \mathbb{E}_{\boldsymbol{x}^* \sim \mathcal{D}, \boldsymbol{y}^* \sim o(\boldsymbol{x}^*)} \left[\mathbb{I}_{(\boldsymbol{x}^*, \boldsymbol{y}^*)}(\boldsymbol{x}, \boldsymbol{y})\right] + \lambda \cdot u(\boldsymbol{x}), \tag{4.10}$$

where the first term is essentially the same as the supervised data experience function (Equation 4.2) with the only difference that now the label $\boldsymbol{y}^*$ is from the oracle rather than pre-given in $\mathcal{D}$; $\lambda > 0$ is a trade-off parameter. The formulation of the active supervision is interesting as it is simply a combination of the common supervision experience and the informativeness measure in an *additive* manner.

We plug f_{active} into the SE and obtain the algorithm to carry out learning. The result turns out to recover classical active learning algorithms.

Active learning. Specifically, in Equation 3.2, setting $f = f_{\text{active}}$, and $(\alpha = 1, \beta = \epsilon)$ as in supervised MLE, the resulting student-step in Equation 3.3 for updating $\boldsymbol{\theta}$ is written as

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}^* \sim \tilde{p}_d(\boldsymbol{x}) \cdot \exp\{\lambda u(\boldsymbol{x})\},\ \boldsymbol{y}^* \sim o(\boldsymbol{x}^*)} \left[\log p_\theta(\boldsymbol{x}^*, \boldsymbol{y}^*)\right]. \tag{4.11}$$

If the pool $\mathcal{D}$ is large, the update can be carried out by the following procedure: we first pick a random subset $\mathcal{D}_{\text{sub}}$ from $\mathcal{D}$, and select a sample from $\mathcal{D}_{\text{sub}}$ according to the informativeness distribution proportional to $\exp\{\lambda u(\boldsymbol{x})\}$ over $\mathcal{D}_{\text{sub}}$. The sample is then labeled by the oracle, which is finally used to update the target model. By setting λ to a very large value (i.e., a near-zero

'temperature' $1/\lambda$), we tend to select the *most* informative sample from $\mathcal{D}_{\text{sub}}$. The procedure rediscovers the algorithm proposed in (Ertekin et al., 2007) and more generally the pooling-based active learning algorithms (Settles, 2012).

4.2. **Knowledge-Based Experience.** Many aspects of problem structures and human knowledge are difficult if not impossible to be expressed through individual data instances. Examples include the knowledge of expected feature values, maximum margin structures (Section 2.3), logical rules, and so on. The knowledge generally imposes constraints that we want the target model to satisfy. The experience function in the standard equation is a natural vehicle for incorporating such knowledge constraints in learning. Given a configuration $\boldsymbol{t}$, the experience function $f(\boldsymbol{t})$ measures the degree to which the configuration satisfies the constraints.

As an example, we consider first-order logic (FOL) rules, which provide an expressive declarative language to encode complex symbolic knowledge (Hu et al., 2016). More concretely, let $f_{\text{rule}}(\boldsymbol{t})$ be an FOL rule w.r.t. the variables $\boldsymbol{t}$. For flexibility, we use soft logic (Bach et al., 2017) to formulate the rule. Soft logic allows continuous truth values from the interval $[0, 1]$ instead of $\{0, 1\}$, and the Boolean logical operators are redefined as:

$$\begin{aligned} A\&B &= \max\{A + B - 1, 0\}, \quad A \vee B = \min\{A + B, 1\} \\ A_1 \wedge \cdots \wedge A_N &= \sum\nolimits_i A_i/N, \quad \neg A = 1 - A. \end{aligned} \tag{4.12}$$

Here & and $\wedge$ are two different approximations to logical conjunction: & is useful as a selection operator (e.g., $A\&B = B$ when $A = 1$, and $A\&B = 0$ when $A = 0$), while $\wedge$ is an averaging operator. To give a concrete example, consider the problem of sentiment classification, where given a sentence $\boldsymbol{x}$, we want to predict its sentiment $\boldsymbol{y} \in \{\text{negative } 0, \text{positive } 1\}$. A challenge for a sentiment classifier is to understand the contrastive sense within a sentence and capture the dominant sentiment precisely. For example, if a sentence is of structure 'A-but-B' with the connective 'but', the sentiment of the half sentence after 'but' dominates. Let $\boldsymbol{x}_B$ be the half sentence after 'but' and $\tilde{\boldsymbol{y}}_B \in [0, 1]$ the (soft) sentiment prediction over $\boldsymbol{x}_B$ by the current model, a possible way to express the knowledge as a logical rule $f_{\text{rule}}(\boldsymbol{x}, \boldsymbol{y})$ is:

$$f := f_{\text{rule}}(\boldsymbol{x}, \boldsymbol{y}) = \text{has-'A-but-B'-structure}(\boldsymbol{x}) \Rightarrow (\mathbb{I}(\boldsymbol{y} = 1) \Rightarrow \tilde{\boldsymbol{y}}_B \;\&\; \tilde{\boldsymbol{y}}_B \Rightarrow \mathbb{I}(\boldsymbol{y} = 1)), \tag{4.13}$$

where $\mathbb{I}(\cdot)$ is an indicator function that takes 1 when its argument is true, and 0 otherwise. Given an instantiation (a.k.a. grounding) of $(\boldsymbol{x}, \boldsymbol{y}, \tilde{\boldsymbol{y}}_B)$, the truth value of $f_{\text{rule}}(\boldsymbol{x}, \boldsymbol{y})$ can be evaluated by definitions in Equation 4.12. Intuitively, the $f_{\text{rule}}(\boldsymbol{x}, \boldsymbol{y})$ truth value gets closer to 1 when $\boldsymbol{y}$ and $\tilde{\boldsymbol{y}}_B$ are more consistent.

We then make use of the knowledge-based experience such as $f_{\text{rule}}(\boldsymbol{t})$ to drive learning. The standard equation rediscovers classical algorithms for learning with symbolic knowledge.

Posterior regularization and extensions. By setting $\alpha = \beta = 1$ and f to a constraint function such as f_{rule}, the SE with cross entropy naturally leads to a generalized posterior regularization framework (Hu et al., 2016):

$$\min_{\boldsymbol{\theta}, q} \; -\mathrm{H}\left(q(\boldsymbol{t})\right) - \mathbb{E}_{q(\boldsymbol{t})}\left[\log p_\theta(\boldsymbol{t})\right] - \mathbb{E}_{q(\boldsymbol{t})}\left[f_{\text{rule}}(\boldsymbol{t})\right], \tag{4.14}$$

which extends the conventional Bayesian inference formulation (Section 2.3) by permitting regularization on arbitrary random variables of arbitrary models (e.g., deep neural networks) with complex rule constraints.

The trade-off hyperparameters can also take other values. For example, by allowing arbitrary $\alpha \in \mathbb{R}$, the objective corresponds to the *unified expectation maximization* (UEM) algorithm (Samdani et al., 2012) that extends the posterior regularization for added flexibility.

4.3. **Reward Experience.** We now consider a very different learning setting commonly seen in robotic control and other sequential decision making problems. In this setting, experience is gained by the agent interacting with external environment and collecting feedback in the form of rewards. Formally, we consider a Markov decision process (MDP) as illustrated in Figure 3, where $\boldsymbol{t} = (\boldsymbol{x}, \boldsymbol{y})$ is the state-action pair. For example, in playing a video game, the state $\boldsymbol{x}$ is the game screen by the environment (the game engine) and $\boldsymbol{y}$ can be any game actions. At time t, the environment is in state $\boldsymbol{x}_t$. The agent draws an action $\boldsymbol{y}_t$ according to the policy $p_\theta(\boldsymbol{y}|\boldsymbol{x})$. The state subsequently transitions to $\boldsymbol{x}_{t+1}$ following certain transition dynamics of the environment, and yields a reward $r_t = r(\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathbb{R}$. The general goal of the agent is to learn the policy $p_\theta(\boldsymbol{y}|\boldsymbol{x})$ to maximize the reward in the long run. There could be different specifications of the goal. In this section we focus on the one where we want to maximize the expected discounted reward starting from a state drawn from an arbitrary state distribution $p_0(\boldsymbol{x})$, with a discount factor $\gamma \in [0, 1]$ applied to future rewards.

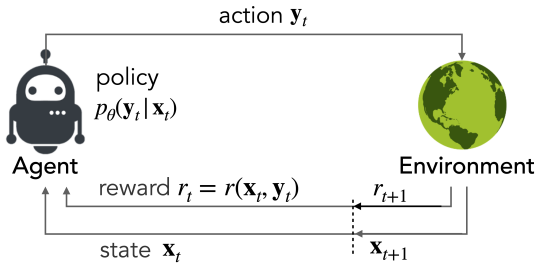


Figure 3. The MDP setting.

A base concept that plays a central role in characterizing the learning in this setting is the *action value function*, also known as the Q function, which is the expected discounted future reward of taking action $\boldsymbol{y}$ in state $\boldsymbol{x}$ and continuing with the policy p_θ:

$$Q^\theta(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid \boldsymbol{x}_0 = \boldsymbol{x}, \boldsymbol{y}_0 = \boldsymbol{y}\right], \tag{4.15}$$

where the expectation is taken by following the state dynamics induced by the policy (thus the dependence of Q^θ on policy parameters $\boldsymbol{\theta}$). We next discuss how $Q^\theta(\boldsymbol{x}, \boldsymbol{y})$ can be used to specify the experience function in different ways, which in turn derives various known algorithms in reinforcement learning (RL) (Sutton & Barto, 2017). Note that here we are primarily interested in learning the conditional model (policy) $p_\theta(\boldsymbol{y}|\boldsymbol{x})$. Yet we can still define the joint distribution as $p_\theta(\boldsymbol{x}, \boldsymbol{y}) = p_\theta(\boldsymbol{y}|\boldsymbol{x})p_0(\boldsymbol{x})$.

4.3.1. ***Expected future reward.*** The first simple way to use the reward signals as the experience is by defining the experience function as the logarithm of the expected future reward:

$$f := f^\theta_{\text{reward},1}(\boldsymbol{x}, \boldsymbol{y}) = \log Q^\theta(\boldsymbol{x}, \boldsymbol{y}), \tag{4.16}$$

which leads to the classical policy gradient algorithm (Sutton et al., 2000).

Policy gradient. With $\alpha = \beta = 1$, we arrive at policy gradient. To see this, consider the teacher-student optimization procedure in Equation 3.3, where the teacher-step yields the q solution:

$$q^{(n)}(\boldsymbol{x}, \boldsymbol{y}) = p_{\theta^{(n)}}(\boldsymbol{x}, \boldsymbol{y}) Q^{\theta^{(n)}}(\boldsymbol{x}, \boldsymbol{y}) \;/\; Z, \tag{4.17}$$

and the student-step updates $\boldsymbol{\theta}$ with the gradient at $\boldsymbol{\theta} = \boldsymbol{\theta}^{(n)}$:

$$\begin{aligned} &\mathbb{E}_{q^{(n)}(\boldsymbol{x},\boldsymbol{y})} \left[\nabla_\theta \log p_\theta(\boldsymbol{x}, \boldsymbol{y})\right] + \mathbb{E}_{q^{(n)}(\boldsymbol{x},\boldsymbol{y})} \left[\nabla_\theta f^\theta_{\text{reward},1}(\boldsymbol{x}, \boldsymbol{y})\right] \;\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(n)}} \\ &= 1/Z \cdot \sum_{\boldsymbol{x}} p_0(\boldsymbol{x}) \nabla_\theta \sum_{\boldsymbol{y}} p_\theta(\boldsymbol{y}|\boldsymbol{x}) Q^\theta(\boldsymbol{x}, \boldsymbol{y}) \;\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(n)}} \\ &= 1/Z \cdot \sum_{\boldsymbol{x}} \mu^\theta(\boldsymbol{x}) \sum_{\boldsymbol{y}} Q^\theta(\boldsymbol{x}, \boldsymbol{y}) \nabla_\theta p_\theta(\boldsymbol{y}|\boldsymbol{x}) \;\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(n)}}. \end{aligned} \tag{4.18}$$

Here the first equation is due to the log-derivative trick $g \nabla \log g = \nabla g$; and the second equation is due to the policy gradient theorem (Sutton et al., 2000), where $\mu^\theta(\boldsymbol{x}) = \sum_{t=0}^\infty \gamma^t p(\boldsymbol{x}_t = \boldsymbol{x})$ is the unnormalized discounted state visitation measure. The final form is exactly the policy gradient up to a multiplication factor $1/Z$.

We can also consider a slightly different use of the reward, by directly setting the experience function to the Q function:

$$f := f^\theta_{\text{reward},2}(\boldsymbol{x}, \boldsymbol{y}) = Q^\theta(\boldsymbol{x}, \boldsymbol{y}). \tag{4.19}$$

This turns out to connect to the known RL-as-inference approach that has a long history of research (e.g., Abdolmaleki et al., 2018; Dayan & Hinton, 1997; Deisenroth et al., 2013; Levine, 2018; Rawlik et al., 2012).

RL as inference. We set $\alpha = \beta := \rho > 0$. The configuration corresponds to the approach that casts RL as a probabilistic inference problem. To see this, we introduce an additional binary random variable o, with $p(o = 1|\boldsymbol{x}, \boldsymbol{y}) \propto \exp\{Q(\boldsymbol{x}, \boldsymbol{y})/\rho\}$. Here $o = 1$ is interpreted as the event that maximum reward is obtained, $p(o = 1|\boldsymbol{x}, \boldsymbol{y})$ is seen as the 'conditional likelihood', and ρ is the temperature. The goal of learning is to maximize the marginal likelihood of optimality: $\log p(o = 1)$, which, however, is intractable to solve. Much like how the standard equation applied to unsupervised MLE provides a surrogate variational objective for the marginal data likelihood (Section 4.1.3), here the standard equation also derives a variational bound for $\log p(o = 1)$ (up to a constant factor) with the above specification of (f, α, β):

$$\begin{aligned} -\log p(o = 1) &= -\log \mathbb{E}_{p_\theta(\boldsymbol{x},\boldsymbol{y})} \left[p(o = 1|\boldsymbol{x}, \boldsymbol{y})\right] \\ &\le -\rho \mathrm{H}\left(q\right) - \rho \mathbb{E}_{q(\boldsymbol{x},\boldsymbol{y})} \left[\log p_\theta(\boldsymbol{x}, \boldsymbol{y})\right] - \mathbb{E}_{q(\boldsymbol{x},\boldsymbol{y})} \left[Q^\theta(\boldsymbol{x}, \boldsymbol{y})\right]. \end{aligned} \tag{4.20}$$

Following the teacher-student procedure in Equation 3.3, the teacher-step produces the q solution:

$$q^{(n)}(\boldsymbol{x}, \boldsymbol{y}) = p_{\theta^{(n)}}(\boldsymbol{x}, \boldsymbol{y}) \exp\left\{Q^{\theta^{(n)}}(\boldsymbol{x}, \boldsymbol{y})/\rho\right\} \;/\; Z. \tag{4.21}$$

The subsequent student-step involves approximation by fixing $\boldsymbol{\theta} = \boldsymbol{\theta}^{(n)}$ in $Q^\theta(\boldsymbol{x}, \boldsymbol{y})$ in the above variational objective, and minimizes only $\mathbb{E}_{q^{(n)}(\boldsymbol{x},\boldsymbol{y})} \left[\log p_\theta(\boldsymbol{x}, \boldsymbol{y})\right]$ w.r.t. $\boldsymbol{\theta}$.

4.3.2. ***Intrinsic reward.*** Rewards provided by the extrinsic environment can be sparse in many real-world sequential decision problems. Learning in such problems is thus difficult due to the lack of supervision signals. A method to alleviate the difficulty is to supplement the extrinsic reward

with dense *intrinsic* reward that is generated by the agent itself (i.e., the agent is intrinsically motivated). The intrinsic reward can be induced in various ways, such as the 'curiosity'-based reward that encourages the agent to explore novel or 'surprising' states (Houthooft et al., 2016; Pathak et al., 2017; Schmidhuber, 2010), or the 'optimal reward', which is designed with the goal of encouraging maximum extrinsic reward at the end (Singh et al., 2010; Z. Zheng et al., 2018). Formally, let $r_t^{in} = r^{in}(\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathbb{R}$ be the intrinsic reward at time t with state $\boldsymbol{x}_t$ and action $\boldsymbol{y}_t$. For example, in (Pathak et al., 2017), r_t^{in} is the prediction error (i.e., the 'surprise') of the next state $\boldsymbol{x}_{t+1}$. Let $Q^{in,\theta}(\boldsymbol{x}, \boldsymbol{y})$ denote the action-value function for the intrinsic reward, defined in a similar way as the extrinsic $Q^{\theta}(\boldsymbol{x}, \boldsymbol{y})$:

$$Q^{in,\theta}(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t^{in} \mid \boldsymbol{x}_0 = \boldsymbol{x}, \boldsymbol{y}_0 = \boldsymbol{y}\right]. \tag{4.22}$$

It is straightforward to derive the intrinsically motivated variant of the policy gradient algorithm (and other RL algorithms discussed below), by replacing the standard extrinsic-only $Q^{\theta}(\boldsymbol{x}, \boldsymbol{y})$ in the experience function Equation 4.16 with the combined $Q^{\theta}(\boldsymbol{x}, \boldsymbol{y}) + Q^{in,\theta}(\boldsymbol{x}, \boldsymbol{y})$. Let $f^{\theta}_{\text{reward,ex+in}}(\boldsymbol{x}, \boldsymbol{y})$ denote the resulting experience function that incorporates both the extrinsic and the additive intrinsic rewards.

We can notice some sort of symmetry between $f^{\theta}_{\text{reward,ex+in}}(\boldsymbol{x}, \boldsymbol{y})$ and the actively supervised data experience f_{active} in Equation 4.10, which augments the standard supervised data experience with the additive informativeness measure $u(\boldsymbol{x})$. The resemblance could naturally inspire mutual exchange between the research areas of intrinsic reward and active learning, for example, using the active learning informativeness measure as the intrinsic reward r^{in}, as was studied in earlier work (L. Li et al., 2011; Pathak et al., 2019; Schmidhuber, 2010).

4.4. **Model-Based Experience.** A model may also learn from other models of the same or related tasks. For example, one can learn a small-size target model by mimicking the outputs of a larger pretrained model, or an ensemble of multiple models, that is more powerful but often too expensive to deploy. Thus, the large model serves as the experience, or the source of information about the task at hand. By seeing that the large source model is effectively providing 'pseudo-labels' on the observed inputs $\mathcal{D} = \{\boldsymbol{x}^*\}$, we can readily write down the corresponding experience function, as a variant of the standard supervised data experience function in Equation 4.2:

$$f := f^{\text{mimicking}}_{\text{model}}(\boldsymbol{x}, \boldsymbol{y}; \mathcal{D}) = \log \mathbb{E}_{\boldsymbol{x}^* \sim \mathcal{D}, \tilde{\boldsymbol{y}} \sim p_{\text{model}'}(\boldsymbol{y}|\boldsymbol{x}^*)} \left[\mathbb{I}_{(\boldsymbol{x}^*, \tilde{\boldsymbol{y}})}(\boldsymbol{x}, \boldsymbol{y})\right], \tag{4.23}$$

where $\tilde{\boldsymbol{y}} \sim p_{\text{model}'}(\boldsymbol{y}|\boldsymbol{x}^*)$ denotes that we draw label samples from the output distribution of the large source model.

Another way of model-based experience is that the pretrained model directly measures the score of a given configuration $(\boldsymbol{x}, \boldsymbol{y})$ with its (log-)likelihood:

$$f := f^{\text{scoring}}_{\text{model}}(\boldsymbol{x}, \boldsymbol{y}) = \log p_{\text{model}'}(\boldsymbol{y}|\boldsymbol{x}). \tag{4.24}$$

As a concrete example, consider learning a text generation model that aims to generate text $\boldsymbol{y}$ conditioning on a given sentiment label $\boldsymbol{x}$ (either positive or negative). A natural form of experience that has encoded the concept of 'sentiment' is a pretrained sentiment classifier, which can be used to measure the likelihood (plausibility) that the given sentence $\boldsymbol{y}$ has the given sentiment $\boldsymbol{x}$ (Hu

et al., 2017). The likelihood serves as the experience function score used to drive target model training through the SE.

Knowledge distillation. Plugging the model-based experience function $f_{\text{model}}^{\text{mimicking}}$ (Equation 4.23) into SE rediscovers the well-known knowledge distillation algorithm (G. Hinton et al., 2015). Specifically, by following the same SE specification of the supervised MLE (Section 4.1.1) and setting $f = f_{\text{model}}^{\text{mimicking}}$, we obtain the knowledge distillation objective:

$$\mathbb{E}_{\boldsymbol{x}^* \sim \mathcal{D}, \tilde{\boldsymbol{y}} \sim p_{\text{model}'}(\boldsymbol{y}|\boldsymbol{x}^*)} \left[\log p_\theta(\tilde{\boldsymbol{y}}|\boldsymbol{x}^*) \right], \tag{4.25}$$

which trains the target model p_θ by encouraging it to mimic the source model outputs (and thus the source model is also called 'teacher' model—in a similar sense to but not to be confused with the teacher-student mechanism for optimization described in Sections 3 and 7).

5. Divergence Function

We now turn to the divergence term $\mathbb{D}(q, p_\theta)$ that measures the distance between the auxiliary distribution q and the model distribution p_θ in the SE. The discussion in the prior section has assumed specific case of $\mathbb{D}$ being the cross entropy. Yet there is a rather rich set of choices for the divergence function, such as f-divergence (e.g., KL divergence, Jensen-Shannon divergence), optimal transport distance (e.g., Wasserstein distance), and so on.

To see a concrete example of how the divergence function may influence the learning, consider the experience to be data instances with the data distribution $p_d(\boldsymbol{t})$, and, following the configurations of supervised MLE (Section 4.1.1), set $f = f_{\text{data}}$, $\alpha = 1$, $\beta = \epsilon$, and the uncertainty measure $\mathbb{H}$ to be the Shannon entropy. As a result, the solution of q in Equation 3.2 reduces to the data distribution $q(\boldsymbol{t}) = p_d(\boldsymbol{t})$. The learning of the model thus reduces to minimizing the divergence between the model and data distributions:

$$\min_{\boldsymbol{\theta}} \mathbb{D}(p_d, p_\theta), \tag{5.1}$$

which is a common objective shared by many ML algorithms depending on how the divergence function is specialized. Thus, in this setting, the divergence function directly determines the learning objective. In the following sections, we will see other richer influences of $\mathbb{D}$ in combination with other SE components.

We next discuss some of the common choices for the divergence function, which opens up the door to recover and generalize more learning algorithms besides those discussed earlier, such as the generative adversarial learning (e.g., GANs, Goodfellow et al., 2014) that is widely used to simulate complex distributions (e.g., natural image distributions).

5.1. Cross Entropy and Kullback–Leibler (KL) Divergence.

The diverse algorithms discussed in Section 4 have all been based on the cross entropy as the divergence function in SE, namely,

$$\mathbb{D}(q, p_\theta) = -\mathbb{E}_q \left[\log p_\theta \right]. \tag{5.2}$$

A nice advantage of using the cross entropy is the close-form solution of q in the teacher-student procedure, as shown in Equation 3.3, which makes the optimization and analysis easier.

In the case where the uncertainty function is the Shannon entropy $\mathbb{H}(q) = -\mathbb{E}_q[\log q]$ (as is commonly assumed in this article), one could alternatively see the above algorithms as using the KL divergence for $\mathbb{D}$, by noticing that $\mathrm{KL}(q, p_\theta) = \mathbb{E}_q[\log q] - \mathbb{E}_q[\log p_\theta]$. That is, given specific balancing weights (α_0, β_0), the divergence and uncertainty terms in SE can be rearranged as:

$$\alpha_0 \mathbb{E}_q[\log q] - \beta_0 \mathbb{E}_q\left[\log p_\theta\right] := (\alpha_0 - \beta_0)\mathbb{E}_q\left[\log p_\theta\right] + \beta_0 \mathrm{KL}\left(q, p_\theta\right), \tag{5.3}$$

where the KL divergence term corresponds to $\mathbb{D}$ in SE if we see $\alpha = \alpha_0 - \beta_0$ and $\beta = \beta_0$.

5.2. **Jensen-Shannon (JS) Divergence.** JS divergence provides another common choice for the divergence function:

$$\mathbb{D}(q, p_\theta) = \mathrm{JS}\left(q \| p_\theta\right) = \frac{1}{2}\mathrm{KL}\left(q \| h\right) + \frac{1}{2}\mathrm{KL}\left(p_\theta \| h\right), \tag{5.4}$$

where $h := \frac{1}{2}(q + p_\theta)$ is the mean distribution. In particular, by considering the specific instantiation in Equation 5.1 of the SE and setting $\mathbb{D}$ to the JS divergence, we can derive the algorithm for learning the generative adversarial networks (GANs, Goodfellow et al., 2014) as shown below. From this perspective, the key concept in generative adversarial learning, namely the *discriminator*, arises as an approximation to the optimization procedure. We discuss in Section 6 an alternative view of the learning paradigm where the discriminator plays the role of 'dynamic' experience in SE.

Generative adversarial learning: The functional descent view. To optimize the objective in Equation 5.1 with the JS divergence, *probability functional descent* (PFD, Chu et al., 2019) offers an elegant way that recovers the optimization procedure of GANs originally developed in Goodfellow et al. (2014). Here we give the PFD result directly, and provide a more detailed review of the PFD optimization in Section 7.

Specifically, let $J(p) := \mathbb{D}(p_d, p)$ in Equation 5.1, which is a functional on the distribution $p \in \mathcal{P}(\mathcal{T})$. The PFD approach (Chu et al., 2019) shows that minimizing $J(p)$ w.r.t p can equivalently be done by solving the following saddle-point problem:

$$\inf_{p} \sup_{\varphi} \mathbb{E}_p\left[\varphi(\boldsymbol{t})\right] - J^*(\varphi), \tag{5.5}$$

where $\varphi : \mathcal{T} \to \mathbb{R}$ is a continuous function, and $J^*(\varphi) = \sup_h \mathbb{E}_h\left[\varphi(\boldsymbol{t})\right] - J(h)$ is the convex conjugate of J. In the case of $J(p)$ being the JS divergence with p_d, if we approximate the above optimization by parameterizing $\varphi(\boldsymbol{t})$ as $\varphi_\phi = \frac{1}{2}\log(1 - C_\phi) - \frac{1}{2}\log 2$ where $C_\phi : \mathcal{T} \to [0, 1]$ is a binary classifier (a.k.a., discriminator) and p as p_θ (i.e., the target model), Equation 5.5 recovers the original GAN algorithm (Goodfellow et al., 2014):

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \frac{1}{2}\mathbb{E}_{p_d}\left[\log C_\phi(\boldsymbol{t})\right] - \frac{1}{2}\mathbb{E}_{p_\theta}\left[\log(1 - C_\phi(\boldsymbol{t}))\right]. \tag{5.6}$$

5.3. **Wasserstein Distance.** Another distance measure that is receiving increasingly interest is the Wasserstein distance, a member of the optimal transport distance family (Peyré, Cuturi, et al., 2019; Santambrogio, 2015). Compared to many of the divergence metrics (e.g., KL divergence), Wasserstein distance has the desirable properties as a distance metric, such as symmetry and the triangle inequality. Based on the Kantorovich duality (Santambrogio, 2015, Section 1.2), the first-order Wasserstein distance between the two distributions q and p can be written as:

$$W_1(q, p) = \sup_{\|\varphi\|_L \leq 1} \mathbb{E}_q\left[\varphi(\boldsymbol{t})\right] - \mathbb{E}_p\left[\varphi(\boldsymbol{t})\right], \tag{5.7}$$

where $\|\varphi\|_L \leq 1$ is the constraint of $\varphi : \mathcal{T} \to \mathbb{R}$ being a 1-Lipschitz function.

Wasserstein GAN. Setting the divergence function in Equation 5.1 to the Wasserstein distance W_1, we thus recover the Wasserstein GAN algorithm (Arjovsky et al., 2017):

$$\min_{\boldsymbol{\theta}} W_1(p_d, p_\theta) = \min_{\boldsymbol{\theta}} \sup_{\|\varphi\|_L \leq 1} \mathbb{E}_{p_d}\left[\varphi(\boldsymbol{t})\right] - \mathbb{E}_{p_\theta}\left[\varphi(\boldsymbol{t})\right], \tag{5.8}$$

which is shown to be more robust than the original GAN algorithm based on the JS divergence (Section 5.2).

6. Dynamic SE

The standard equation Equation 3.1 so far has played the role of the ultimate learning objective that fully defines the learning problem in an analytical form. As seen in Sections 4 and 5, many of the known algorithms are special cases of the SE objective. On the other hand, in a dynamic or online setting, the learning objective itself may be evolving over time. For example, the data instances may follow changing distributions or come from evolving tasks (e.g., a sequence of tasks that are increasingly complex); the experience in a strategic game context can involve complex interactions with the target model through co-training or adversarial dynamics; and the success criteria of the model w.r.t. the experience can be adapting. In this section, we discuss an *extended* view of the SE in dealing with the learning in such dynamic contexts. Instead of serving as the static overall objective function, now the SE is a core part of an outer loop in the learning procedure.

More specifically, each of the SE components (e.g., experience function, divergence function, balancing weights) can change over time. For example, consider a dynamic experience function f_τ, which is indexed by τ indicating its evolution over time, iterations, or tasks. This differs from the experience discussed earlier in Section 4 that is defined a priori (e.g., a static set of data instances) and encoded as a fixed experience function f. With the dynamic experience, a learning procedure, using the special case of SE in Equation 3.2, can be written as:

$$\begin{aligned} &\text{for } \tau = 1, 2, \ldots : \\ &\quad \text{Acquire experience } f_\tau, \\ &\quad \text{Solve SE: } \min_{q, \boldsymbol{\theta}} -\alpha \mathbb{H}(q) + \beta \mathbb{D}(q, p_\theta) - \mathbb{E}_q\left[f_\tau\right]. \end{aligned} \tag{6.1}$$

Here the SE governs the optimization of the target model p_θ given the updated experience at each τ. We discuss in more detail the SE with dynamic experience (Section 6.1) and other dynamic components (Section 6.2), which further recovers several well-known algorithms in existing learning paradigms.

6.1. Dynamic Experience.

The experience can change over time due to different reasons. For example, the experience can involve optimization, often together with the target model, resulting in a bi-level optimization scheme (Section 6.1.1); or alternatively, the experience may just come from the environment sequentially with an unknown dynamic (Section 6.1.2).

6.1.1. *Variational Experience With Optimization.*

As a concrete example, consider the experiences that do not have an analytic form, but instead are defined in a variational way (i.e., as a

solution to an optimization problem):

$$f_\tau := \operatorname*{argmax}_f \mathcal{J}(f, p_{\theta^{(\tau-1)}}), \tag{6.2}$$

with an optimization objective $\mathcal{J}$, where $p_{\theta^{(\tau-1)}}$ is the target model learned with the experience from the last iteration.

A particular example is the *adversarial* experience emergingly used in many generation and representation learning problems. Specifically, recall the data instance experience $f_{\text{data}}(\boldsymbol{t}; \mathcal{D})$ that measures the closeness between a configuration $\boldsymbol{t}$ with the true data $\mathcal{D}$ based on data instance matching (Equation 4.2). Such manually defined measures could be subjective, suboptimal, or demanding expertise or heavy engineering to be properly specified. An alternative way that sidesteps the drawbacks is to automatically induce a closeness measure $f_\phi(\boldsymbol{t})$, where $\boldsymbol{\phi}$ denotes any free parameters associated with the experience and is to be learned. For example, one can measure the closeness of a configuration $\boldsymbol{t}$ to the data set $\mathcal{D}$ based on a *discriminator* (or *critic*) that evaluates how easily $\boldsymbol{t}$ can be differentiated from the instances in $\mathcal{D}$. A concrete application is in image generation, where a binary discriminator takes as input an image sample and tells whether the input image is a real instance from the observed image corpus $\mathcal{D}$ or a fake one produced by the model. The similar idea of discriminator-based closeness measure was also explored in the likelihood-free inference literature (Gutmann et al., 2018).

The discriminator/critic as the experience can be learned or adapted together with the target model training in an iterative way, as in Equation 6.2. We show below that the discriminator/critic-based experience, in combination of certain choices of the divergence function $\mathbb{D}$, re-derives the generative adversarial learning (Goodfellow et al., 2014) from a different perspective than Section 5.2.

Generative adversarial learning: The variational experience view. The functional descent view of generative adversarial learning presented in Section 5.2 is based on the treatment that the experience is the given static data instances, and the various GAN algorithms are due to the different choices of the divergence function. The extended view of SE in this section also allows an alternative viewpoint of the learning paradigm, that gives more flexibility in not only choosing the divergence function but also the experience function, leading to a richer set of GAN variants.

In this viewpoint, we consider experience that is defined variationally as mentioned above. That is, the experience function f, as a measure of the goodness of a sample $\boldsymbol{t}$, is not specified a priori but rather defined through an optimization problem. As a concrete example, we define f as a binary classifier f_ϕ with sigmoid activation and parameters $\boldsymbol{\phi}$, where the value $f_\phi(\boldsymbol{t})$ measures the *log* probability of the sample $\boldsymbol{t}$ being a real instance (as opposed to a model generation). Thus the higher $f_\phi(\boldsymbol{t})$ value, the higher quality of sample $\boldsymbol{t}$. The parameters $\boldsymbol{\phi}$ of the experience function need to be learned. We can do so by augmenting the standard equation (Equation 3.2) with added optimization of $\boldsymbol{\phi}$ in various ways. The following equation gives one of the approaches:

$$\min_{q,\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \; -\alpha \mathbb{H}(q) + \beta \mathbb{D}(q, p_\theta) - \mathbb{E}_q[f_\phi] + \mathbb{E}_{p_d}[f_\phi], \tag{6.3}$$

where, besides the optimization of q and $\boldsymbol{\theta}$, we additionally maximize over $\boldsymbol{\phi}$ with the extra term $\mathbb{E}_{p_d}[f_\phi]$ to form the classification problem $\max_{\boldsymbol{\phi}} -\mathbb{E}_q[f_\phi] + \mathbb{E}_{p_d}[f_\phi]$. Further assuming a particular configuration of the tradeoff hyperparameters $\alpha = 0$ and $\beta = 1$, the resulting objective

$$\min_{q,\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \; \mathbb{D}(q, p_\theta) - \mathbb{E}_q[f_\phi] + \mathbb{E}_{p_d}[f_\phi], \tag{6.4}$$

turns out to relate closely to generative adversarial learning.

In particular, with proofs adapted from Farnia and Tse (2018), Equation 6.4 recovers the vanilla GAN algorithm when $\mathbb{D}$ is the Jensen-Shannon divergence and assuming the space of f_ϕ, denoted as $\mathcal{F}$, is convex. More specifically, if we denote the probability $C_\phi(\boldsymbol{t}) = \exp f_\phi(\boldsymbol{t})$, then the equation reduces to the familiar GAN objective in Equation 5.6. The results can be extended to the more general case of f-GAN (Nowozin et al., 2016): if we set $\mathbb{D}$ to an f-divergence and do not restrict the form (e.g., classifier) of the experience function f_ϕ, then with mild conditions, the equation recovers the f-GAN algorithm. Now consider $\mathbb{D}$ as the first-order Wasserstein distance and suppose the f_ϕ-space $\mathcal{F}$ is a convex subset of 1-Lipschitz functions. It can be shown that Equation 6.4 reduces to the Wasserstein GAN algorithm as shown in Equation 5.8 where φ now corresponds to f_ϕ. Note that for the above configurations, if f_ϕ is parameterized as a neural network with a fixed architecture (e.g., ConvNet), its space $\mathcal{F}$ is not necessarily convex (i.e., a linear combination of two neural networks in $\mathcal{F}$ is not necessarily in $\mathcal{F}$). In such cases we formulate the optimization of the experience function over $\mathrm{conv}(\mathcal{F})$, the convex hull of $\mathcal{F}$ containing any convex combination of neural network functions in $\mathcal{F}$ (Farnia & Tse, 2018), and see the various GAN algorithms as approximations by considering only the subset $\mathcal{F} \subseteq \mathrm{conv}(\mathcal{F})$.

Besides the above examples of divergence $\mathbb{D}$ that each leads to a different GAN algorithm, we can consider even more options, such as the hybrid f-divergence and Wasserstein distance studied in (Farnia & Tse, 2018). Of particular interest is to set $\mathbb{D}$ to the KL divergence $\mathbb{D}(q, p_\theta) = \mathrm{KL}(q\|p_\theta)$, motivated by the simplicity in the sense that the auxiliary distribution q has a closed-form solution:[1] at each iteration n,

$$q^{(n+1)}(\boldsymbol{t}) = p_{\theta^{(n)}}(\boldsymbol{t}) \exp\left\{f_{\phi^{(n)}}(\boldsymbol{t})\right\} \;/\; Z, \tag{6.5}$$

where Z is the normalization factor. As shown in Y. Wu et al. (2020), the particular form of solution results in a new variant of GANs that enables more stable optimization of the experience function (discriminator) f_ϕ. Concretely, the discriminator f_ϕ can now be optimized with importance reweighting:

$$\begin{aligned} &\max_{\boldsymbol{\phi}} - \mathbb{E}_{\boldsymbol{t}\sim q^{(n+1)}}\left[f_\phi(\boldsymbol{t})\right] + \mathbb{E}_{\boldsymbol{t}\sim p_d}\left[f_\phi(\boldsymbol{t})\right] \\ &\quad = -\frac{1}{Z}\mathbb{E}_{\boldsymbol{t}\sim p_\theta^{(n)}}\Big[\exp\{f_{\phi^{(n)}}(\boldsymbol{t})\}\cdot f_\phi(\boldsymbol{t})\Big] + \mathbb{E}_{\boldsymbol{t}\sim p_d}\left[f_\phi(\boldsymbol{t})\right], \end{aligned} \tag{6.6}$$

where importance sampling is used to estimate the expectation under $q^{(n+1)}$, using the generator $p_\theta^{(n)}$ as the proposal distribution. Compared to the vanilla and Wasserstein GANs above, the fake samples from the generator are now weighted by the exponentiated discriminator score $\exp\{f_{\phi^{(n)}}(\boldsymbol{t})\}$ when used to update the discriminator. Intuitively, the mechanism assigns higher weights to samples that can fool the discriminator better, while low-quality samples are downplayed to avoid degrading the discriminator performance during training.

Besides the generative adversarial learning, in Section 4 we briefly mentioned that many of the conventional experience can also benefit from the idea of introducing adaptive or learnable components, for example, data instances with automatically induced data weights or learned augmentation

[1] We can alternatively derive the objective from Equation 6.3, by setting $\alpha = \beta = 1, \mathbb{D}$ to the cross entropy, and $\mathbb{H}$ to the Shannon entropy. Note that $\mathrm{KL}(q\|p_\theta) = -\mathrm{H}(q) - \mathbb{E}_q[\log p_\theta]$. Thus the solution of q can be derived as in Equation 3.3.

policies (Section 4.1.4). We discuss more in Section 9.2 about how the unified SE as the basis can effortlessly derive efficient approaches for joint model and experience optimization.

6.1.2. ***Online Experience From the Environment***. Another form of dynamic experience comes from the changing environments, such as the data stream in online learning (Shalev-Shwartz et al., 2012) whose distribution can change over time, or the experience in lifelong learning (Thrun, 1998) that differs across a series of tasks.

In particular, we consider an online setting: at each time $\tau \in \{1, ..., T\}$, a predictor is given an input and is required to make a prediction (e.g., if the stock market will go up or down tomorrow where). We have access to the recommended prediction by each of the K experts $\boldsymbol{t} = \{1, ..., K\}$, and make our prediction accordingly. As a result, the environment reveals a reward based on the discrepancy between the prediction and the true answer. The sequence of data instances follows a dynamic that is unknown and can even be adversarially adaptive to the predictor's behavior (e.g., in the problem of spam email filtering, or other strategic game environments) (Shalev-Shwartz et al., 2012). In such cases, we can only hope the predictor to achieve some relative performance guarantee, in particular w.r.t. the best single expert in hindsight. This is formally captured by *regret*, which is the difference between the cumulative reward of the predictor and that of the best single expert. We further consider the specific setting where we submit a 'soft' prediction $p_\tau(\boldsymbol{t})$, that is, the distribution (a vector of normalized weights) over the K experts, and receive the rewards $r_\tau(\boldsymbol{t}) \in \mathbb{R}$ for each expert $\boldsymbol{t}$. The goal of the learning problem is then to update the distribution $p_\tau(\boldsymbol{t})$ at every step τ for minimal regret $\sum_{\tau=1}^{T} r_\tau(\boldsymbol{t}^*) - \sum_{\tau=1}^{T} \mathbb{E}_{p_\tau(\boldsymbol{t})}[r_\tau(\boldsymbol{t})]$, where $\boldsymbol{t}^*$ is the best single expert.

The rewards from the environment naturally serves as the dynamic experience:

$$f_\tau(\boldsymbol{t}) := r_\tau(\boldsymbol{t}). \tag{6.7}$$

In the following, we show that the SE rediscovers the classical multiplicative weights (MW), or Hedge algorithm (Freund & Schapire, 1997) to the above online problem. Similar ideas of multiplicative weights have been widely used in diverse fields such as optimization, game theory, and economics (Arora et al., 2012).

Multiplicative weights. To update the distribution (i.e., normalized weight vector) over the K experts at each time τ, we treat the distribution as the target model p_θ to be learned in the SE. In other words, p_θ is directly parameterized as the normalized weight vector $p_\theta := \boldsymbol{\theta} = \{\theta_{\boldsymbol{t}}\}_{\boldsymbol{t}=1}^{K}$, where $\theta_{\boldsymbol{t}} \geq 0$ is the probability of expert $\boldsymbol{t}$, with $\sum_{\boldsymbol{t}=1}^{K} \theta_{\boldsymbol{t}} = 1$. Starting from the SE in Equation 3.2, and assuming $\mathbb{D}$ to be the cross entropy, $\mathbb{H}$ the Shannon entropy, and $\alpha = \beta > 0$, we obtain the update rule of the target model at each time τ following the teacher-student mechanism in Equation 3.3. Specifically, the teacher step has:

$$\text{Teacher:} \quad q^{(\tau+1)}(\boldsymbol{t}) = p_{\theta^{(\tau)}}(\boldsymbol{t}) \exp\left\{\alpha^{-1} f_\tau(\boldsymbol{t})\right\} \ / \ Z. \tag{6.8}$$

The subsequent student step is to minimize the cross entropy between the target model p_θ and $q^{(\tau+1)}$ (see Equation 3.3). Given the definition of p_θ as the vector of probabilities, the student step is equivalent to directly setting p_θ to the vector of $q^{(\tau+1)}(\boldsymbol{t})$ values. Therefore:

$$\text{Student:} \quad p_{\theta^{(\tau+1)}}(\boldsymbol{t}) = p_{\theta^{(\tau)}}(\boldsymbol{t}) \exp\left\{\alpha^{-1} f_\tau(\boldsymbol{t})\right\} \ / \ Z, \tag{6.9}$$

which is precisely the multiplicative weight update rule for the expert distribution/weights. That is, at each time, the weights are updated by multiplying them with a factor that depends on the reward.

6.2. **Dynamic SE as Interpolation of Algorithms.** Besides the dynamic experience, other components in the SE, such as the divergence function $\mathbb{D}$ and the balancing weights (α, β), can also be indexed by the time τ with desired evolution. In particular, the previous sections have shown that the different specifications of the SE components correspond to different specific algorithms, many of which are well-known and have different properties. The dynamic SE with the evolving specifications, therefore, can be seen as *interpolating* between the algorithms during the course of training. The interpolation allows the learning to enjoy different desired properties in different training stages, resulting in improved efficacy.

As a concrete example, Tan et al. (2018) learn a text generation model by starting with the simple supervised MLE algorithm, with the experience function $f_{\tau=0} = f_{\text{data}}$ (Equation 4.2) and balancing weights $(\alpha_{\tau=0} = 1, \beta_{\tau=0} = \epsilon)$ as described in Section 4.1.1. After warming up with the supervised MLE for n iterations, the approach then changes the experience function to the data augmentation-based one $f_{\tau=n} = f_{\text{data-aug}}$ defined in Equation 4.8, which introduces noise and task-specific evaluation information into the training. As revealed in Section 4.1.4, this stage in effect corresponds to the reward-augmented maximum likelihood (RAML) algorithm (Norouzi et al., 2016). The approach proceeds by further annealing the balancing weights, specifically by gradually increasing β_τ from ϵ to 1 as τ increases. The increase of β effectively gets the learning closer to a reinforcement-style learning (notice that the policy gradient algorithm has $\alpha = \beta = 1$ as described in Section 4.3.1). Intuitively, the target model $p_{\theta^{(t)}}$, as part of the $q^{(\tau+1)}$ solution in the teacher step weighted by the increasing weight β (see Equation 3.3), serves to produce more data samples. Those samples are weighted by the experience function and used for updating the target model further, simulating the policy gradients. Therefore, the whole learning procedure spans multiple paradigms of learning (from supervised MLE, RAML, to reinforcement learning) that increasingly introduces more noise and exploration for improved robustness. All the involved paradigms of algorithms are perfectly encompassed in the single SE formulation, allowing users to simply tweak and evolve the relevant components for the interpolation.

7. Optimization Algorithms

Thus far, we have discussed the standard equation as the unified objective function. Learning the target model p_θ amounts to optimizing the objective w.r.t the model parameters $\boldsymbol{\theta}$. That is, the standardized objective presents an optimization problem, for which an optimization solver is applied to obtain the target model solution p_{θ^*}. This section is devoted to discussion of various optimization algorithms.

For a simple objective such as that of the vanilla supervised MLE (Equation 2.1) with a tractable model, stochastic gradient descent can be used to optimize the model parameters $\boldsymbol{\theta}$ straightforwardly. With a more complex model or a more complex objective like the general SE, more sophisticated optimization algorithms are needed, such as the teacher-student procedure exemplified in Equation 3.3. Like the standardized formulation of the objective function, we would like to quest for a *standardized optimization algorithm* that is generally applicable to optimizing the objective under vastly different specifications. Yet it seems still unclear whether such a universal solver exists

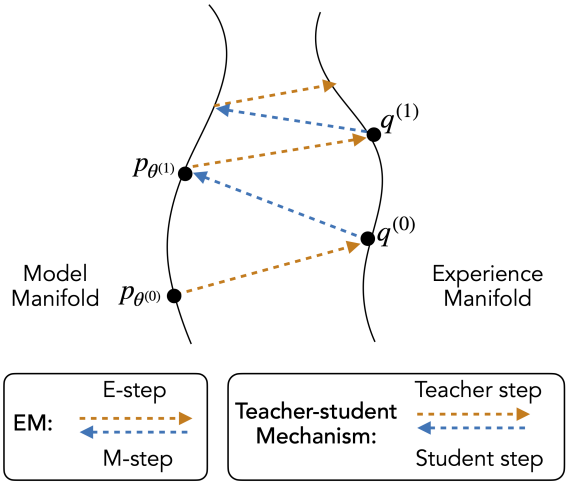


Figure 4. EM and the teacher-student mechanism entail alternating projection to achieve local minima.

and what it looks like. On the other hand, some techniques may hold the promise to generalize to broad settings.

7.1. **Alternating Projection.** Alternating projection (Bauschke & Borwein, 1996; Csiszár, 1975) provides a general class of optimization algorithms from a geometry point of view, subsuming as special cases many of the optimization algorithms discussed above, ranging from the EM (Section 2.1.2) to the teacher-student algorithm (Section 3). At a high level, the algorithms consider the optimization problem as to find a common point of a collection of sets, and achieve it by alternatingly projecting between those sets. For example, the EM algorithm entails alternating projection, as shown in Figure 4, where the E-step (Equation 2.10) projects the current model distribution $p_{\theta^{(n)}}(\boldsymbol{x}, \boldsymbol{y})$ onto the set of distributions whose marginal over $\boldsymbol{x}$ equals the empirical distribution, i.e., $q^{(n+1)} = \operatorname{argmin}_q \mathrm{KL}\left(q(\boldsymbol{y}|\boldsymbol{x})\tilde{p}_d(\boldsymbol{x}) \| p_{\theta^{(n)}}(\boldsymbol{x}, \boldsymbol{y})\right)$, then the subsequent M-step (Equation 2.11) projects $q^{(n+1)}$ onto the set of possible model distributions through $\min_\theta \mathrm{KL}\left(q^{(n+1)}(\boldsymbol{y}|\boldsymbol{x})\tilde{p}_d(\boldsymbol{x}) \| p_\theta(\boldsymbol{x}, \boldsymbol{y})\right)$.

In general, the SE in Equation 3.1 or (3.2), with both the model parameters $\boldsymbol{\theta}$ and the auxiliary distribution q to be learned, can naturally be optimized with an alternating-projection style procedure, which we have referred to as the teacher-student mechanism.

7.2. **The Teacher-Student Mechanism.** We have seen a special case of the teacher-student mechanism in Equation 3.3 for solving the specifically instantiated SE (e.g., with cross entropy as the divergence function $\mathbb{D}$). The optimization procedure is also an example of alternating projection (Figure 4). Specifically, the teacher $q^{(n+1)}$ is the projection of the student $p_{\theta^{(n)}}$ onto the set defined by the experience, and the student $p_{\theta^{(n+1)}}$ is the projection of the teacher $q^{(n+1)}$ onto the set of model distributions.

7.2.1. ***The Teacher Step***. The teacher step in Equation 3.3 has a closed-form solution for the teacher $q^{(n+1)}$ due to the choice of cross entropy as the divergence function $\mathbb{D}$ in SE:

$$\text{Teacher:} \quad q^{(n+1)}(\boldsymbol{t}) = \exp\left\{\frac{\beta \log p_{\theta^{(n)}}(\boldsymbol{t}) + f(\boldsymbol{t})}{\alpha}\right\} \Big/ Z. \tag{7.1}$$

In the more general case where other complex divergence functions are used (as those in Section 5), a closed-form teacher is usually not available. The probabilistic functional descent (PFD) mentioned in Section 5.2, with approximations to the influence function using convex duality, offers a possible way of solving for q for a broader class of divergences, such as Jensen-Shannon divergence and Wasserstein distance. It thus presents a promising venue for future research to develop more generic solvers for the broad learning problems characterized by SE.

On the other hand, as can be seen in the student step discussed shortly, sometimes we do not necessarily need a closed-form teacher $q^{(n+1)}$ in the learning, but only need to be able to draw samples from $q^{(n+1)}$.

Probability functional descent. Generally, the SE (Equation 3.1 or 3.2) defines a loss over the auxiliary distribution q, denoted as $J(q)$, which is a functional on the auxiliary distribution space $\mathcal{Q}(\mathcal{T})$. The Gâteaux derivative of J at q, if exists, is defined as (Fernholz, 2012):

$$J'_q(h-q) = \lim_{\epsilon \to 0^+} \frac{J(q+\epsilon(h-q)) - J(q)}{\epsilon} \tag{7.2}$$

for any given $h \in \mathcal{Q}(\mathcal{T})$. Intuitively, $J'_q(h-q)$ describes the change of the $J(q)$ value with respect to an infinitesimal change in q in the direction of $(h-q)$. The Gâteaux derivative $J'_q(h-q)$ can alternatively be computed with the *influence function* of J at q, denoted as $\psi_q : \mathcal{T} \to \mathbb{R}$, through:

$$J'_q(h-q) = \int_{\boldsymbol{t}} \psi_q(\boldsymbol{t})(h-q)d\boldsymbol{t} = \mathbb{E}_h\left[\psi_q(\boldsymbol{t})\right] - \mathbb{E}_q\left[\psi_q(\boldsymbol{t})\right]. \tag{7.3}$$

The above notions allow us to define gradient descent applied to the functional J. Concretely, we can define a linear approximation to $J(q)$ around a given q_0:

$$\begin{aligned} J(q) &\approx J(q_0) + J'_{q_0}(q-q_0) \\ &= J(q_0) + \mathbb{E}_q\left[\psi_{q_0}(\boldsymbol{t})\right] - \mathbb{E}_{q_0}\left[\psi_{q_0}(\boldsymbol{t})\right] \\ &= \mathbb{E}_q\left[\psi_{q_0}(\boldsymbol{t})\right] + const. \end{aligned} \tag{7.4}$$

Thus, $J(q)$ can approximately be minimized with an iterative descent procedure: at each iteration n, we perform a descent step that decreases $\mathbb{E}_q\left[\psi_{q^{(n)}}(\boldsymbol{t})\right]$ w.r.t. q, yielding $q^{(n+1)}$ for the next iteration.

Once the functional gradient is defined as above, the remaining problem of the optimization is then about how to obtain the influence function ψ_q given the functional $J(q)$. In some cases the influence function as defined in Equation 7.3 is not directly tractable and approximations are needed. Chu et al. (2019) developed a variational approximation method applied when J is convex (which is the case in Equation 3.2 when $\mathbb{D}$ is convex w.r.t q). Concretely, with the convex conjugate of J defined as $J^*(\varphi) = \sup_h \mathbb{E}_h\left[\varphi(\boldsymbol{t})\right] - J(h)$, it can be shown under mild conditions that the influence function for J at q is:

$$\psi_q = \mathrm{argmax}_{\varphi \in \mathcal{C}(\mathcal{T})}\, \mathbb{E}_q\left[\varphi(\boldsymbol{t})\right] - J^*(\varphi), \tag{7.5}$$

where $\mathcal{C}(\mathcal{T})$ is the space of continuous functions $\mathcal{T} \to \mathbb{R}$. We thus can approximate the influence function by parameterizing it as a neural network and training the network to maximize the objective $\mathbb{E}_q\left[\varphi(\boldsymbol{t})\right] - J^*(\varphi)$. Plugging the approximation of influence function into the above functional descent

procedure leads to the full PFD optimization:

$$\inf_{q} \sup_{\varphi} \mathbb{E}_q \left[\varphi(\boldsymbol{t})\right] - J^*(\varphi), \tag{7.6}$$

which is a saddle-point problem.

7.2.2. ***The Student Step***. The student step optimizes the SE objective w.r.t. the target model parameters $\boldsymbol{\theta}$, given $q^{(n+1)}$ from the teacher step. The optimization is to minimize the divergence between the student p_θ and the teacher $q^{(n+1)}$:

$$\text{Student:} \quad \boldsymbol{\theta}^{(n+1)} = \operatorname*{argmin}_{\boldsymbol{\theta}} \mathbb{D}\left(q^{(n+1)}, p_\theta\right). \tag{7.7}$$

Section 5 discussed the different choices of the divergence function $\mathbb{D}$. In particular, in the same setting of Equation 3.3 where $\mathbb{D}$ is the common cross entropy (or KL divergence as discussed in Section 5.1), the student step is written as:

$$\boldsymbol{\theta}^{(n+1)} = \operatorname*{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{q^{(n+1)}(\boldsymbol{t})}\big[\log p_\theta(\boldsymbol{t})\big], \tag{7.8}$$

where $q^{(n+1)}$ is in the form of Equation 7.1 above. The optimization then amounts to first drawing samples from the teacher $\tilde{\boldsymbol{t}} \sim q^{(n+1)}$, and then updating $\boldsymbol{\theta}$ by maximizing the log-likelihood of those samples under p_θ. We could apply various sampling methods to draw samples from $q^{(n+1)}$, such as Markov chain Monte Carlo (MCMC) methods, like Gibbs sampling when $\boldsymbol{t}$ is discrete and Hamiltonian or Langevin MC for continuous $\boldsymbol{t}$ (Duane et al., 1987; Neal, 1992; Welling & Teh, 2011), and sampling based on stochastic differential equations (SDEs, Liu et al., 2022; Song et al., 2020).

In the special case of $\alpha = \beta$, the teacher model reduces to $q^{(n+1)}(\boldsymbol{t}) \propto p_{\theta^{(n)}}(\boldsymbol{t}) \exp\{f(\boldsymbol{t})/\alpha\}$. This special form allows us to use a simple *importance sampling* method (Hu et al., 2018), with $p_{\theta^{(n)}}$ (i.e., the current model distribution) as the proposal distribution:

$$\boldsymbol{\theta}^{(n+1)} = \operatorname*{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{\tilde{\boldsymbol{t}} \sim p_{\theta^{(n)}}(\boldsymbol{t})}\big[\exp\{f(\tilde{\boldsymbol{t}})/\alpha\} \cdot \log p_\theta(\tilde{\boldsymbol{t}})\big] \;/\; Z. \tag{7.9}$$

Here the experience function $f(\boldsymbol{t})$ serves as a sample reweighting mechanism (Y. Wu et al., 2020) that highlights the 'high-quality' samples (i.e., those receiving high goodness scores in the light of the experience) for updating the parameters $\boldsymbol{\theta}$.

For other choices of the divergence function $\mathbb{D}$ other than the cross entropy or KL divergence, sampling from $q^{(n+1)}$ may similarly be sufficient in order to estimate and optimize the divergence Equation 7.7. The probability functional descent (PFD) can also be used to optimize with a certain class of divergences, following similar derivations discussed in Sections 5 and 7.2.1.

8. The Target Model

Besides the objective function (Sections 3-6) and the optimization (Section 7) above, we now briefly discuss the third and last core ingredient of an ML approach, namely, the target model p_θ. A key feature of the SE as an objective function is that the formulation is agnostic to the specific choices of the target model. That is, one can use the SE to train effectively anything with a learnable component (denoted as $\boldsymbol{\theta}$ in general), ranging from black-box neural networks, to probabilistic graphical models, symbolic models, and the combinations thereof. We will mention a few examples

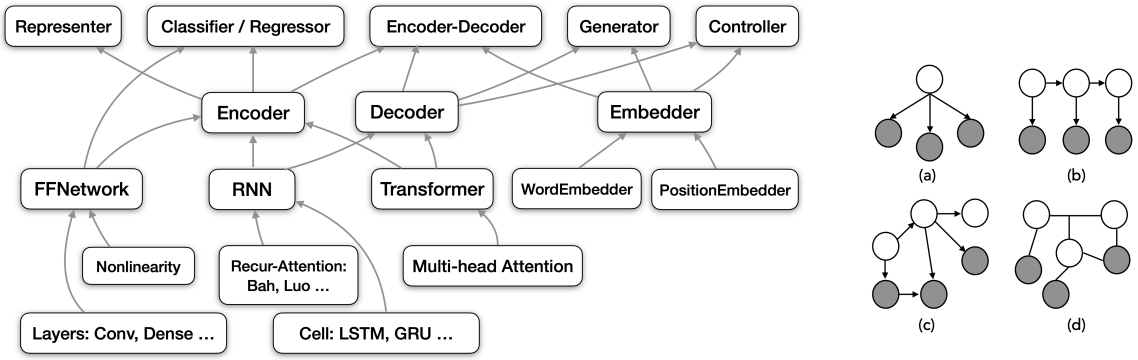


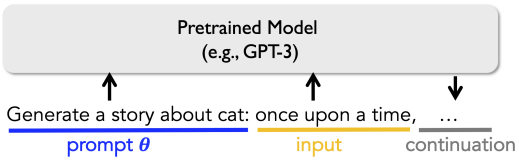
Figure 5. Left: An (incomplete) set of neural model components at different granularities. Components at a lower level are composed to form higher-level, more complex ones. **Right:** Directed/undirected probabilistic graphical models of various dependence structures.

of the target model that are of radically different categories, illustrating the utility and generality of the SE.

Deep neural networks of any architectures. The target model can be a deep neural network of an arbitrary architecture, composed of different neural layers and units. Figure 5, right panel, illustrates a set of common neural components and how they are composed to form more complex ones at different levels of granularity. Neural models of different architectures are used for different tasks of interest. For example, an image classifier $p_\theta(\boldsymbol{y}|\boldsymbol{x})$, with input image $\boldsymbol{x}$ and output object label $\boldsymbol{y}$, typically consists of an encoder and a classification layer (a.k.a., classification head), where the encoder can be a ConvNet (LeCun et al., 1989) or transformer (Vaswani et al., 2017) and the classification layer is often a simple multilayer perceptron (MLP) consisting of several neural dense layers. As another example, a neural language model $p_\theta(\boldsymbol{t})$ over text sequence $\boldsymbol{t}$ is a generator with a decoder component (which is in turn implemented via a transformer or other neural architectures).

The $\boldsymbol{\theta}$ to be learned includes the neural network weights. It can be randomly initialized and then trained from scratch. On the other hand, with the prevalence of *pretrained models*, such as BERT (Devlin et al., 2019) for text representation and GPT-3 (Brown et al., 2020) for language modeling, we can often initialize $\boldsymbol{\theta}$ with the appropriate pretrained model weights and finetune it to the downstream tasks of interest. Sometimes it is sufficient to finetune only a subset of the model parameters (e.g., the classification head of the image classifier) while keeping all other parts fixed. In many downstream tasks, it is often difficult to obtain a large number of supervised data instances to train/finetune $\boldsymbol{\theta}$ with simple supervised MLE. In such cases, SE offers a more flexible framework that allows plugging in all other forms of experience (as those in Section 4) related to the downstream tasks for more effective learning.

Prompts for pretrained models. Updating large pretrained models can be prohibitively expensive due to the massive amount of model parameters. Prompting is a new emerging way of steering the pretrained models for performing downstream tasks without changing the model parameters (Brown et al., 2020). A prompt is a short sequence of text tokens or embedding vectors which, by concatenating with the input and being fed together into the model, stimulates the model to perform the task of interest and produce the desired output for the input (Figure 6, left). It is thus



Head entity	*Relation*	*Tail entity*
students	*worth celebrating*	graduate
newborn	*can but not good at*	sit
social worker	*can help*	foster child
honey	*ingredient for*	honey cake
cabbage	*ingredient for*	cabbage salad
China	*separated by the ocean*	Japan
Africa	*separated by the ocean*	Europe

Figure 6. Left: prompt as the learnable component $\boldsymbol{\theta}$ to steer pretrained models (such as GPT-3) to perform the downstream task of interest (e.g., generating a story of a certain topic). **Right:** The novel learned knowledge tuples by treating the symbolic knowledge graph as the target model (Hao et al., 2022).

desirable to learn the prompt that enables optimal performance of the pretrained models on the task of interest. For continuous prompt, which is a sequence of differentiable embedding vectors (Lester et al., 2021; X. L. Li & Liang, 2021), we could naturally treat the prompt as the learnable $\boldsymbol{\theta}$, and plug the resulting target model p_θ, now the pretrained model plus the learnable prompt, into the SE for training. On the other hand, for discrete prompt, which is a sequence of text tokens, one can instead parameterize a prompt-generation network as $\boldsymbol{\theta}$ which, after training, generates optimal discrete prompts for the downstream task. For example, Deng et al. (2022) used both supervised data instances and task reward to learn the target prompt-generation models.

Symbolic knowledge graphs. The target model can even be a symbolic system such as a knowledge graph (KG) or a rule set. Here $\boldsymbol{\theta}$ denotes the KG structure to be learned, and $p_\theta(\boldsymbol{t})$ can be seen as a distribution assigning a (non-)uniform nonzero probability to any knowledge tuple $\boldsymbol{t}$ in the KG and zero to all other tuples. Hao et al. (2022) presents a concrete instance of the learning system that incrementally learns (extracts) a commonsense relational KG (Figure 6, right), using the pretrained language models such as BERT (Devlin et al., 2019) as the experience.

Probabilistic graphical models and composite models. The target model $p_\theta(\boldsymbol{t})$ can also be probabilistic graphical models (Jordan, 2003; Koller & Friedman, 2009), a rich family of models characterizing the conditional dependence structure between random variables with a directed/undirected graph (Figure 5, right). Graphical models may also be composed with the neural modules to form more complex composite models, typically with the neural modules extracting features from the raw inputs and the graphical modules capturing the high-level structures (Johnson et al., 2016; Wilson, Hu, Salakhutdinov, et al., 2016; S. Zheng et al., 2015). As discussed in the earlier sections, the common learning and inference approaches for probabilistic graphic models, such as the (variational) EM algorithm, are special instances of SE and its teacher-student mechanism. The SE framework offers a generalized formulation for learning graphical and composite models.

9. Panoramic Learning with All Experience

The preceding sections have presented a standardized formalism of machine learning, on the basis of the standard equation of objective function, that provides a succinct, structured formulation of a broad design space of learning algorithms, and subsumes a wide range of known algorithms in a unified manner. The simplicity, modularity, and generality of the framework is particularly

appealing not only from the theoretical perspective but also because it offers guiding principles for mechanical design of algorithmic approaches to challenging problems, in the presence of diverse experience, and hence serves as an important step toward the goal of panoramic learning. In this section, we discuss the use of the standard equation to drive systematic design of new learning methods, which in turn yield various algorithmic approaches to problems in different application domains.

9.1. **Combining Rich Experience.** As one of the original motivations for the standardization, the framework allows us to combine together all different experience to learn models of interest. Learning with multiple types of experience is a necessary capability of AI agents to be deployed in a real dynamic environment and to improve from heterogeneous signals in different forms, at varying granularities, from diverse sources, and with different intents (e.g., adversaries). Such versatile learning capability is best exemplified by human learning—for instance, we learn a new language by reading and hearing examples, studying grammars and rules, practicing and interacting with others, receiving feedback, associating with prior languages we already mastered, and so forth. To build a similar panoramic-learning AI agent, having a standardized learning formalism is perhaps an indispensable step.

The standard equation we have presented is naturally designed to fit the needs. In particular, the experience function provides a straightforward vehicle for experience combination. For example, the simplest approach is perhaps to make a weighted composition of multiple individual experience functions:

$$f(\boldsymbol{t}) = \sum\nolimits_i \lambda_i f_i(\boldsymbol{t}), \tag{9.1}$$

where each f_i characterizes a specific source of experience, and $\lambda_i > 0$ is the respective weight. One can readily plug in arbitrary available experience, such as data, logical rules, constraints, rewards, and auxiliary models, as components of the experience function.

With the SE, designing an approach to a problem thus boils down to choosing and formulating *what* experience to use depending on the problem structure and available resources, without worrying too much about *how* to use the experience. This provides a potentially new modularized and repeatable way of producing ML approaches to different problems, as compared to the previous practice of designing bespoke algorithm for each individual problem. Sections 4 and 6 have discussed possible formulations of the diverse types of experience as an experience function to be plugged into Equation 9.1. It is still an open question how even more types of experience, such as massive knowledge graphs (Bach et al., 2019; Tan et al., 2020), can efficiently be formulated as an experience function that assesses the 'goodness' of an input $\boldsymbol{t}$. On the other hand, the discussion in the next subsection offers new opportunities that relieve users from having to manually specify every detail of the experience function. Instead, users only need to specify parts of the experience function of which they are certain, and leave the remaining parts plus the weights λ_i (Equation 9.1) to be automatically learned together with the target model.

Case study: Text attribute transfer. As a case study of learning from rich experience, consider the problem of text attribute transfer where we want to rewrite a given piece of text to possess a desired attribute (Hu et al., 2017; Shen et al., 2017). Taking the sentiment attribute, for example, given a sentence $\boldsymbol{x}$ (e.g., a customer's review *"the manager is a horrible person"*) and a target

Method	Negative to Positive
Original	it was super dry and had a weird taste to the entire slice .
(Shen et al., 2017)	it was super friendly and had a nice touch to the same .
f_{sc}	good good good
$f_{\text{sc}} + f_{\text{data}}$	it was super well-made and had a weird taste to the entire slice .
$f_{\text{sc}} + f_{\text{data}} + f_{\text{LM}}$	it was super fresh and had a delicious taste to the entire slice .
Method	**Positive to Negative**
Original	besides that , the wine selection they have is pretty awesome as well .
(Shen et al., 2017)	after that , the quality prices that does n't pretty much well as .
f_{sc}	besides horrible horrible as
$f_{\text{sc}} + f_{\text{data}}$	besides that , the wine selection they have is pretty borderline as atrocious .
$f_{\text{sc}} + f_{\text{data}} + f_{\text{LM}}$	besides that , the wine selection they have is pretty horrible as well .

Method	Attribute accuracy (↑)	Preservation (↑)	Fluency (↓)
(Shen et al., 2017)	79.5%	12.4	51.4
f_{sc}	99.6%	1.2	259.2
$f_{\text{sc}} + f_{\text{data}}$	87.7%	65.6	177.7
$f_{\text{sc}} + f_{\text{data}} + f_{\text{LM}}$	91.2%	57.8	53.95

Table 2. Results of text attribute transfer on the Yelp data (Hu et al., 2017; Yang et al., 2018). **Top:** Generated samples of different methods given the *Original* sentence, for negative-to-positive and positive-to-negative transfer, respectively. Positive and negative words are highlighted in red and blue, respectively. **Bottom:** Performance of different methods. For the evaluation metrics: "attribute accuracy" is evaluated with a pretrained sentiment classifier on the model generations; "preservation" is the BLEU score ($\in [0, 100]$) that evaluates the lexical overlapping between the generated sentences and the original input sentences; "fluency" is the perplexity score of generated text evaluated by a pretrained language model (LM) (the lower, the better). The model trained with all three types of experience achieves the best overall performance in terms of all the three aspects.

sentiment a (e.g., positive), the goal of the problem is to generate a new sentence $\boldsymbol{y}$ that (1) possesses the target sentiment, (2) preserves all other characteristics of the original sentence, and (3) is fluent (e.g., the transferred sentence "*the manager is a perfect person*"). To learn an attribute transfer model $p_\theta(\boldsymbol{y}|\boldsymbol{x}, a)$, a key challenge of the problem is the lack of direct supervision data (i.e., pairs of sentences that are exact the same except for sentiment), making it necessary to use other forms of experience. Here we briefly describe an approach originally presented in Hu et al. (2017) and Yang et al. (2018), highlighting how the approach can be built mechanically, by formulating relevant experience directly based on the problem definition and then plugging them into the SE.

We can identify three types of experience, corresponding to the above three desiderata, respectively. First, the model needs to learn the concept of 'sentiment' to be able to modify the attribute of text. As mentioned in Section 4.4, a natural form of experience that has encoded the concept is a pretrained sentiment classifier (SC). The first experience function can then be defined as $f_{\text{sc}}(\boldsymbol{x}, a, \boldsymbol{y}) = \text{SC}(a, \boldsymbol{y})$, which evaluates the log likelihood of the transferred sentence $\boldsymbol{y}$ possessing the sentiment a. The higher value the $\boldsymbol{y}$ achieves, the higher quality it is considered in light of the experience. The second desideratum requires the model to reconstruct as much of the input text $\boldsymbol{x}$

as possible. We combine the second experience $f_{\text{data}}(\boldsymbol{x}, a, \boldsymbol{y}|\mathcal{D})$ (Equation 4.2) defined by a set of simple reconstruction data instances $\mathcal{D} = \{(\boldsymbol{x}^*, a^* = a_{\boldsymbol{x}^*}, \boldsymbol{y}^* = \boldsymbol{x}^*)\}$, where the target sentiment a^* is set to the sentiment of the original sentence $\boldsymbol{x}^*$, and by the problem definition the ground-truth output $\boldsymbol{y}^*$ is exact the same as the input $\boldsymbol{x}^*$. Such data thus carry the information of preserving the input content. Finally, for the requirement of generating fluent text, we can again naturally use an auxiliary model as the experience, namely, a pretrained language model (LM) $f_{\text{LM}}(\boldsymbol{x}, a, \boldsymbol{y}) = \text{LM}(\boldsymbol{y})$ that estimates the log likelihood of a sentence $\boldsymbol{y}$ under the natural language distribution. After identifying the experience $(f_{\text{sc}}, f_{\text{data}}, f_{\text{LM}})$, we then combine them together with Equation 9.1 and plug into the SE to train the target model $p_\theta(\boldsymbol{y}|\boldsymbol{x}, a)$. More experimental details can be found in Hu et al. (2017) and Yang et al. (2018). Table 2 shows the empirical results on the common Yelp corpus of customer reviews. We can see that by plugging in the relevant experience, the resulting model successfully learns the respective aspects of the task. For example, with only f_{sc}, the model is able to transfer the sentiment attribute but fails on content preservation and fluency. Adding the second experience f_{data} encourages preservation. Further with f_{LM}, the model substantially improves the fluency, achieving the best overall performance. The case study demonstrates the necessity of integrating the diverse experience for solving the problem.

9.2. **Repurposing Learning Algorithms for New Problems.** The standardized formalism sheds new light on fundamental relationships between a number of learning problems in different research areas, showing that they are essentially the same under the SE perspective. This opens up a wide range of opportunities for generalizing existing algorithms, which were originally designed for specialized problems, to a much broader set of new problems. It is also made easy to exchange between the diverse research areas in aspects of modeling, theoretical understanding, approximation, and optimization. For example, an earlier successful approach to challenges in one area can now be readily applied to address challenges in another. Similarly, a future progress made in one problem could immediately unlock progresses in many others.

Case study: Learning with imperfect experience. To illustrate, let us consider a set of concrete problems concerning with distinct types of experience, which were often studied by researchers in different areas: **(1)** The first problem is to integrate structured knowledge constraints in model training, where some components of the constraints, as well as the constraint weights, cannot be specified a priori and are to be induced automatically; **(2)** The second problem concerns supervised learning, where one has access to only a small set of data instances with imbalanced labels, and we want to automate the data manipulation (e.g., augmentation and reweighting) to maximize the training performance; **(3)** The last problem is to stabilize the notoriously difficult training of generative adversarial networks (GANs) for a wide range of image and text generation tasks.

The three problems, though seemingly unrelated at first sight, can all be reduced to the same underlying problem in the unified SE view, namely, learning with imperfect experience f (e.g., underspecified knowledge constraints, small imbalanced data, and unstable discriminator). We want to automatically adapt/improve the imperfect experience in order to better supervise the target model training. This readily falls into the dynamic SE setting described in Section 6, where now the experience function is $f_\phi(\boldsymbol{t})$ associated with learnable parameters $\boldsymbol{\phi}$. For example, in problem (1), $f_\phi(\boldsymbol{t}) = \sum_i \lambda_\phi^i f_{\text{rule},\phi}^i(\boldsymbol{t})$, where any learnable components in each knowledge constraint $f_{\text{rule},\phi}^i$ (Section 4.2) and the weights λ_ϕ^i constitute the $\boldsymbol{\phi}$ to be learned (Hu et al., 2018). In problem (2), $f_\phi(\boldsymbol{t})$ is instantiated as $f_{\text{data-w},\phi}(\boldsymbol{t}; \mathcal{D})$ (Equation 4.6) with learnable data weights $w(\boldsymbol{t}^*) \in \boldsymbol{\phi}$, or as

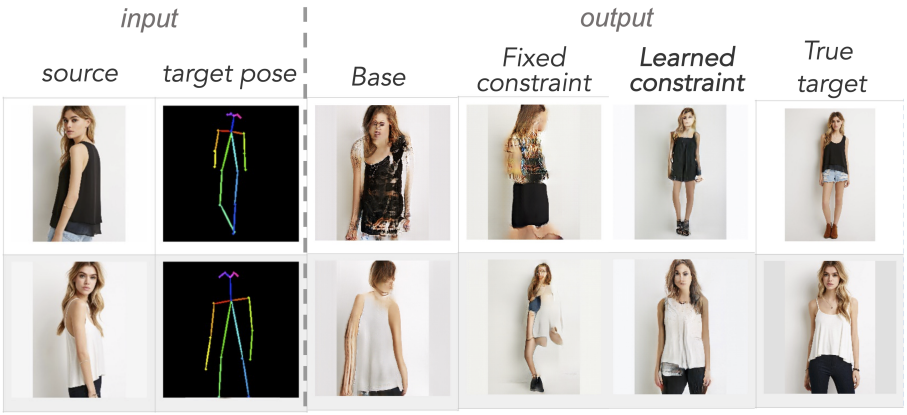


Figure 7. Example outputs of pose-conditioning human image generation. Given an image of a person as well as a target pose represented as a skeleton, the goal is to generate a new image of the same person under the target pose. The *base* model that learns a neural model with only limited available supervised data fails to make meaningful generation. Hu et al. (2018) added a structured constraint on the human body structure, based on a pretrained human part parser. Learning with the *fixed constraint* (i.e., with the pretrained parser) does not improve the generation quality. In contrast, learning the constraint together with the target model within the dynamic standard equation framework leads to substantially improved output (*learned constraint*), close to the *true target*.

$f_{\text{data-aug},\phi}(\boldsymbol{t}; \mathcal{D})$ (Equation 4.8) with the metric for augmentation $a_{\boldsymbol{t}^*}(\boldsymbol{t}) \in \boldsymbol{\phi}$ to be learned (Hu, Tan, et al., 2019). In problem (3), we have discussed the training of $f_\phi(\boldsymbol{t})$ as the GAN discriminator in Section 6.1, but we want to improve the training stability (Y. Wu et al., 2020). Thus, one approach for efficient updates of the general experience function f_ϕ would address all three problems together.

To seek for solutions, we again take advantage of the unified SE perspective that enables us to reuse existing successful techniques instead of having to invent new ones. In particular, the connection of the experience function f with the reward in Section 4.3 naturally inspires us to repurpose known techniques from the fertile reinforcement learning (RL) literature, especially those of learning reward functions such as inverse RL (Ziebart et al., 2008) or learning implicit reward (Z. Zheng et al., 2018), for learning the experience function f_ϕ in our problems. For instance, following (Ziebart et al., 2008), one can acquire and update the experience function at each iteration in Equation 6.2 through $\min_{\boldsymbol{\phi}} -\mathbb{E}_{\boldsymbol{t}^* \sim \tilde{p}_{\text{data}}} [\log q(\boldsymbol{t}^*)]$, where q taking the form in Equation 3.3 now depends on $\boldsymbol{\phi}$. The resulting procedure induces an importance reweighting scheme that is shown to stabilize the discriminator training in GANs (Y. Wu et al., 2020), as well as learn meaningful constraints (Hu et al., 2018). Figure 7 provides a demonstration that learning the constraints together with the target model within the dynamic SE framework leads to substantial improvement.

10. Related Work

It has been a constant aspiration to search for basic principles that unify the different paradigms in machine learning (Bishop, 2013; Domingos, 2015; Gori, 2017; Hu, Wilson, et al., 2019; Langley, 1989). Extensive efforts have been made to build unifying views of methods on particular fronts. For example, Roweis and Ghahramani (1999) unified various unsupervised learning algorithms with a linear Gaussian model; Wainwright and Jordan (n.d.) presented the variational method for inference

in general exponential-family graphical models; Domingos (2015) and Richardson and Domingos (2006) presented Markov logic networks that combine Bayesian Markov network with first-order logic for uncertain inference; Knoblauch et al. (2019) developed a generalized form of variational Bayesian inference by allowing losses and divergences beyond the standard likelihood and KL divergence, which subsumes existing variants for Bayesian posterior approximation; Altun and Smola (2006) showed the duality between regularized divergence (e.g., Bregman and f-divergence) minimization and statistical inference (e.g., MAP estimation); Arora et al. (2012) presented a common formulation of different multiplicative weights update methods; Mohamed and Lakshminarayanan (2016) connected generative adversarial learning with a rich set of other statistical learning principles; Y. N. Wu et al. (2019) discussed connections between generative, discriminative, and energy-based models. Ma et al. (2022) presented parsimony and self-consistency as the guiding principles for learning from data. Those unified treatments shed new light on the sets of originally specialized methods and foster new progress in the respective fields. LeCun (2022) presented a modeling architecture to construct autonomous intelligent agents that combines concepts such as world model and hierarchical joint embedding. Our standardized formalism of the learning objective is complementary and offers a general framework for training the relevant model architectures. The framework also covers the key learning ingredients mentioned in LeCun (2022), including the self-supervised learning (Section 4.1.2) and intrinsic motivation (Section 4.3.2).

Integrating diverse sources of information in training has been explored in previous work, which is often dedicated to specific tasks. Roth (2017) presented different ways of deriving supervision signals in different scenarios. Y. Zhu et al. (2020) discussed the integration of physical and other knowledge in solving vision problems. The distant or weak supervision approaches (Mintz et al., 2009; Ratner et al., 2017) automatically create (noisy) instance labels from heuristics, which are then used in the supervised training procedure. The panoramic learning we discussed here makes use of broader forms of experience not necessarily amenable to be converted into supervised labels, such as reward, discriminator-like models, and many structured constraints. The experience function $f(\boldsymbol{y})$ offers such flexibility for expressing all those experiences.

11. Future Directions

We have presented a standardized machine learning formalism, materialized as the standard equation of the objective function, that formulates a vast algorithmic space governed by a few components regarding the experience, model fitness measured with divergence, and uncertainty. The formalism gives a holistic view of the diverse landscape of learning paradigms, allows a mechanical way of designing ML approaches to new problems, and provides a vehicle toward panoramic learning that integrates all available experience in building an AI agent. The work shapes a range of exciting open questions and opportunities for future study. We discuss a few of these directions below.

Continual learning in complex dynamic environments. We have discussed the SE for learning with all diverse forms of experience, in both static environments (e.g., fixed data or reward distributions) and dynamic environments (e.g., optimized or online experience). An exciting next step is to deploy the SE framework to build an AI agent that continually learns in the real-world complex and fast-evolving context, in which the AI agent must learn to identify the relevant experience out of massive external information, to acquire increasingly complex new concepts or skills.

Establishing and applying the standardized formalism to the broader learning settings is expected to unleash even more power by enabling principled design of learning systems that continuously improve by interacting with and collecting diverse signals from the outer world.

Theoretical analysis of panoramic learning. The paradigm of panoramic learning poses new questions about theoretical understanding. A question of particular importance in practice is about how we can guarantee better performance after integrating more experience. The analysis is challenging because the different types of experience can each encode different information, sometimes noisy and even conflicting with each other (e.g., not all data instances would comply with a logic rule), and thus plugging in an additional source of experience does not necessarily lead to positive effects. But before that, a more basic question to ask is, perhaps, how can we characterize learning with some special or novel forms of experience, such as logic rules and auxiliary models? What mathematical tools may we use for characterization, and what would the convergence guarantees, complexity, robustness, and other theoretical and statistical properties be? Inspired by how we generalized the specialized algorithms to new problems, a promising way of the theoretical analysis would be to again leverage the standard equation and repurpose the existing analyses originally dealing with supervised learning, online learning, and reinforcement learning, to now analyze the learning process with all other experiences.

From standardization to automation. As in other mature engineering disciplines such as mechanical engineering, standardization is followed by automation. The standardized ML formalism opens up the possibility of automating the process of creating and improving ML algorithms and solutions. The current 'AutoML' practice has largely focused on automatic search of neural network architectures and hyperparameters, thanks to the well-defined architectural and hyperparameter search spaces. The standard equation that defines a structured algorithmic space would similarly suggest opportunities for automatic search or optimization of learning algorithms, which is expected to be more efficient than direct search on the programming code space (Real et al., 2020). We briefly discussed in Section 9 how new algorithms can mechanically be created by composing experience and/or other algorithmic components together. It would be significant to have an automated engine that further streamlines the process. For example, once a new advance is made to reinforcement learning, the engine would automatically amplify the progress and deliver enhanced functionalities of learning data manipulation and adapting knowledge constraints. Similarly, domain experts can simply input a variety of experiences available into their own problems, and expect an algorithm to be automatically composed to learn the target model they want. The sophisticated algorithm manipulation and creation would greatly simplify machine learning workflow in practice and boost the accessibility of ML to much broader users.

From Maxwell's equations to General Relativity, and to quantum mechanics and Standard Model, "Physics is the study of symmetry," remarked physicist Phil Anderson (Anderson, 1972, p.394). The end goal of physics research seems to be clear—a 'theory of everything' that fully explains and links together all physical aspects. The 'end goal' of ML/AI is surely much more elusive. Yet the unifying way of thinking would be incredibly valuable, to continuously unleash the extensive power of current vibrant research, to produce more principled understanding, and to build more versatile AI solutions.

References

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., & Riedmiller, M. (2018). Maximum a posteriori policy optimisation. *International Conference on Learning Representations*.

Altun, Y., & Smola, A. (2006). Unifying divergence minimization and statistical inference via convex duality. *International Conference on Computational Learning Theory*, 139–153.

Anderson, P. W. (1972). More is different: Broken symmetry and the nature of the hierarchical structure of science. *Science*, *177*(4047), 393–396.

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. *International Conference on Machine Learning*, 214–223.

Arora, S., Hazan, E., & Kale, S. (2012). The multiplicative weights update method: A meta-algorithm and applications. *Theory of Computing*, *8*(1), 121–164.

Bach, S. H., Broecheler, M., Huang, B., & Getoor, L. (2017). Hinge-loss Markov random fields and probabilistic soft logic. *The Journal of Machine Learning Research*, *18*(1), 3846–3912.

Bach, S. H., Rodriguez, D., Liu, Y., Luo, C., Shao, H., Xia, C., Sen, S., Ratner, A., Hancock, B., Alborzi, H., et al. (2019). Snorkel drybell: A case study in deploying weak supervision at industrial scale. *Proceedings of the 2019 International Conference on Management of Data*, 362–375.

Bauschke, H. H., & Borwein, J. M. (1996). On projection algorithms for solving convex feasibility problems. *SIAM review*, *38*(3), 367–426.

Bishop, C. M. (2013). Model-based machine learning. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *371*(1984), 20120222.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877–1901.

Chu, C., Blanchet, J., & Glynn, P. (2019). Probability functional descent: A unifying perspective on GANs, variational inference, and reinforcement learning. *International Conference on Machine Learning*, 1213–1222.

Csiszár, I. (1975). I-divergence geometry of probability distributions and minimization problems. *The Annals of Probability*, 146–158.

Dagan, I., & Engelson, S. P. (1995). Committee-based sampling for training probabilistic classifiers. *Proceedings of the Twelfth International Conference on International Conference on Machine Learning*, 150–157.

Dayan, P., & Hinton, G. E. (1997). Using expectation-maximization for reinforcement learning. *Neural Computation*, *9*(2), 271–278.

Deisenroth, M. P., Neumann, G., Peters, J., et al. (2013). A survey on policy search for robotics. *Foundations and Trends® in Robotics*, *2*(1–2), 1–142.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, *39*(1), 1–22.

Deng, M., Wang, J., Hsieh, C.-P., Wang, Y., Guo, H., Shu, T., Song, M., Xing, E. P., & Hu, Z. (2022). RLPrompt: Optimizing discrete text prompts with reinforcement learning. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186.

Domingos, P. (2015). *The master algorithm: How the quest for the ultimate learning machine will remake our world*. Basic Books.

Duane, S., Kennedy, A. D., Pendleton, B. J., & Roweth, D. (1987). Hybrid Monte Carlo. *Physics letters B*, *195*(2), 216–222.

Ertekin, S., Huang, J., Bottou, L., & Giles, L. (2007). Learning on the border: Active learning in imbalanced data classification. *Proceedings of the sixteenth ACM conference on Conference on Information and Knowledge Management*, 127–136.

Farnia, F., & Tse, D. (2018). A convex duality framework for GANs. *Advances in Neural Information Processing Systems*, *31*, 5248–5258.

Fernholz, L. T. (2012). *Von Mises calculus for statistical functionals* (Vol. 19). Springer Science & Business Media.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, *55*(1), 119–139.

Ganchev, K., Gillenwater, J., Taskar, B., et al. (2010). Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, *11*(Jul), 2001–2049.

Gebru, I. D., Alameda-Pineda, X., Forbes, F., & Horaud, R. (2016). EM algorithms for weighted-data clustering with application to audio-visual scene analysis. *IEEE transactions on pattern analysis and machine intelligence*, *38*(12), 2402–2415.

Glynn, P. W. (1990). Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, *33*(10), 75–84.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, 2672–2680.

Gori, M. (2017). *Machine learning: A constraint-based approach*. Morgan Kaufmann.

Gutmann, M. U., Dutta, R., Kaski, S., & Corander, J. (2018). Likelihood-free inference via classification. *Statistics and Computing*, *28*(2), 411–425.

Hao, S., Tan, B., Tang, K., Zhang, H., Xing, E. P., & Hu, Z. (2022). BertNet: Harvesting knowledge graphs from pretrained language models. *arXiv*. https://arxiv.org/abs/2206.14268

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv*. https://arxiv.org/abs/1503.02531

Hinton, G. E., Dayan, P., Frey, B. J., & Neal, R. M. (1995). The" wake-sleep" algorithm for unsupervised neural networks. *Science*, *268*(5214), 1158.

Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., & Abbeel, P. (2016). VIME: Variational information maximizing exploration. *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 1117–1125.

Hu, Z., Ma, X., Liu, Z., Hovy, E., & Xing, E. (2016). Harnessing deep neural networks with logic rules. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2410–2420.

Hu, Z., Tan, B., Salakhutdinov, R., Mitchell, T., & Xing, E. P. (2019). Learning data manipulation for augmentation and weighting. *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 15764–15775.

Hu, Z., Wilson, A. G., Finn, C., Lee, L., Neiswanger, W., Qin, L., Berg-Kirkpatrick, T., Salakhutdinov, R., & Xing, E. P. (2019). The NeurIPS workshop on learning with rich experience: Integration of learning paradigms. https://sites.google.com/view/neurips2019lire

Hu, Z., Yang, Z., Liang, X., Salakhutdinov, R., & Xing, E. P. (2017). Toward controlled generation of text. *International Conference on Machine Learning*, 1587–1596.

Hu, Z., Yang, Z., Salakhutdinov, R., Liang, X., Qin, L., Dong, H., & Xing, E. P. (2018). Deep generative models with learnable knowledge constraints. *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 10522–10533.

Jaakkola, T., Meila, M., & Jebara, T. (2000). Maximum entropy discrimination. *Advances in Neural Information Processing Systems*, 470–476.

Jaynes, E. T. (1957). Information theory and statistical mechanics. *Physical Review*, *106*(Article 4), 620.

Jaynes, E. T. (1988). [optimal information processing and Bayes's theorem]: Comment. *The American Statistician*, *42*(4), 280–281. Retrieved October 27, 2022, from http://www.jstor.org/stable/2685144

Johnson, M. J., Duvenaud, D., Wiltschko, A. B., Datta, S. R., & Adams, R. P. (2016). Composing graphical models with neural networks for structured representations and fast inference. *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2954–2962.

Jordan, M. I. (2003). An introduction to probabilistic graphical models. *University of California, Berkeley*.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., & Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine learning*, *37*(2), 183–233.

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. *International Conference on Learning Representations*.

Knoblauch, J., Jewson, J., & Damoulas, T. (2019). Generalized variational inference: Three arguments for deriving new posteriors. *arXiv*. https://arxiv.org/abs/1904.02063

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. MIT press.

Langley, P. (1989). Toward a unified science of machine learning. *Machine Learning*, *3*(4), 253–259.

Lecun, Y., & Misra, I. (2021). Self-supervised learning: The dark matter of intelligence. *Blog*. https://ai.facebook.com/blog/self-supervised-learning-the-dark-matter-of-intelligence.

LeCun, Y. (2022). A path towards autonomous machine intelligence. *Preprint*. https://openreview.net/pdf?id=BZ5a1r-kVsf

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), 541–551.

Lester, B., Al-Rfou, R., & Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 3045–3059.

Levine, S. (2018). Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv*. https://arxiv.org/abs/1805.00909

Li, L., Littman, M. L., Walsh, T. J., & Strehl, A. L. (2011). Knows what it knows: A framework for self-aware learning. *Machine Learning*, *3*(82), 399–443.

Li, X. L., & Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 4582–4597.

Liu, G., Feng, Z., Gao, Y., Yang, Z., Liang, X., Bao, J., He, X., Cui, S., Li, Z., & Hu, Z. (2022). Composable text controls in latent space with ODEs. *arXiv*. https://arxiv.org/abs/2208.00638

Ma, Y., Tsao, D., & Shum, H.-Y. (2022). On the principles of parsimony and self-consistency for the emergence of intelligence. *Frontiers of Information Technology & Electronic Engineering*, *23*(9), 1298–1323.

Mintz, M., Bills, S., Snow, R., & Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 1003–1011.

Mnih, A., & Gregor, K. (2014). Neural variational inference and learning in belief networks. *International Conference on Machine Learning*, 1791–1799.

Mohamed, S., & Lakshminarayanan, B. (2016). Learning in implicit generative models. *arXiv*. https://arxiv.org/abs/1610.03483

Neal, R. M. (1992). *Bayesian training of backpropagation networks by the hybrid monte carlo method* (tech. rep.). Citeseer.

Neal, R. M., & Hinton, G. E. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models* (pp. 355–368). Springer.

Norouzi, M., Bengio, S., Jaitly, N., Schuster, M., Wu, Y., Schuurmans, D., et al. (2016). Reward augmented maximum likelihood for neural structured prediction. *Advances In Neural Information Processing Systems*, 1723–1731.

Nowozin, S., Cseke, B., & Tomioka, R. (2016). f-GAN: Training generative neural samplers using variational divergence minimization. *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 271–279.

Paisley, J., Blei, D. M., & Jordan, M. I. (2012). Variational bayesian inference with stochastic search. *Proceedings of the 29th International Conference on Machine Learning*.

Pathak, D., Agrawal, P., Efros, A. A., & Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2778–2787.

Pathak, D., Gandhi, D., & Gupta, A. (2019). Self-supervised exploration via disagreement. *International Conference on Machine Learning*, 5062–5071.

Peyré, G., Cuturi, M., et al. (2019). Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, *11*(5-6), 355–607.

Ranganath, R., Gerrish, S., & Blei, D. (2014). Black box variational inference. *Artificial Intelligence and Statistics*, 814–822.

Ratner, A., Bach, S. H., Ehrenberg, H., Fries, J., Wu, S., & Ré, C. (2017). Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, *11*(3), 269.

Rawlik, K., Toussaint, M., & Vijayakumar, S. (2012). On stochastic optimal control and reinforcement learning by approximate inference. *Proceedings of Robotics: Science and Systems VIII*.

Real, E., Liang, C., So, D., & Le, Q. (2020). AutoML-zero: Evolving machine learning algorithms from scratch. *International Conference on Machine Learning*, 8007–8019.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine learning*, *62*(1), 107–136.

Roth, D. (2017). Incidental supervision: Moving beyond supervised learning. *Thirty-First AAAI Conference on Artificial Intelligence*.

Roweis, S., & Ghahramani, Z. (1999). A unifying review of linear gaussian models. *Neural computation*, *11*(2), 305–345.

Samdani, R., Chang, M.-W., & Roth, D. (2012). Unified expectation maximization. *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 688–698.

Santambrogio, F. (2015). Optimal transport for applied mathematicians. *Birkäuser, NY*, *55*(58-63), 94.

Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*.

Settles, B. (2012). Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *6*(1), 1–114.

Shalev-Shwartz, S., et al. (2012). Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, *4*(2), 107–194.

Shen, T., Lei, T., Barzilay, R., & Jaakkola, T. (2017). Style transfer from non-parallel text by cross-alignment. *NeurIPS*.

Singh, S., Lewis, R. L., Barto, A. G., & Sorg, J. (2010). Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., & Poole, B. (2020). Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations*.

Sutton, R. S., & Barto, A. G. (2017). *Reinforcement learning: An introduction (2nd ed.)* Cambridge, MA: MIT Press.

Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 1057–1063.

Tan, B., Hu, Z., ZichaoYang, R., & Xing, E. (2018). Connecting the dots between mle and rl for sequence generation. *arXiv*. https://arxiv.org/abs/1811.09740

Tan, B., Qin, L., Xing, E., & Hu, Z. (2020). Summarizing text on any aspects: A knowledge-informed weakly-supervised approach. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6301–6309.

Taskar, B., Guestrin, C., & Koller, D. (2004). Max-margin markov networks. *Advances in Neural Information Processing systems*, 25–32.

Thrun, S. (1998). Lifelong learning algorithms. In *Learning to learn* (pp. 181–209). Springer.

Titsias, M. (2009). Variational learning of inducing variables in sparse Gaussian processes. *Artificial Intelligence and Statistics*, 567–574.

Vapnik, V. N. (1998). Statistical learning theory. *John Wiley and Sons*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 6000–6010.

Wainwright, M. J., & Jordan, M. I. (n.d.). A variational principle for graphical models. *New Directions in Statistical Signal Processing*, 155.

Wainwright, M. J., & Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1–305.

Welling, M., & Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 681–688.

Wilson, A. G., Hu, Z., Salakhutdinov, R. R., & Xing, E. P. (2016). Stochastic variational deep kernel learning. *Advances in Neural Information Processing Systems*, *29*.

Wilson, A. G., Hu, Z., Salakhutdinov, R., & Xing, E. P. (2016). Deep kernel learning. *Artificial intelligence and statistics*, 370–378.

Wu, Y. N., Gao, R., Han, T., & Zhu, S.-C. (2019). A tale of three probabilistic families: Discriminative, descriptive, and generative models. *Quarterly of Applied Mathematics*, *77*(2), 423–465.

Wu, Y., Zhou, P., Wilson, A. G., Xing, E., & Hu, Z. (2020). Improving GAN training with probability ratio clipping and sample reweighting. *Advances in Neural Information Processing Systems*, *33*, 5729–5740.

Xing, E. P., Jordan, M. I., & Russell, S. (2002). A generalized mean field algorithm for variational inference in exponential families. *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, 583–591.

Yang, Z., Hu, Z., Dyer, C., Xing, E. P., & Berg-Kirkpatrick, T. (2018). Unsupervised text style transfer using language models as discriminators. *Advances in Neural Information Processing Systems*, *31*.

Yu, J., Yang, M.-S., & Lee, E. S. (2011). Sample-weighted clustering methods. *Computers & mathematics with applications*, *62*(5), 2200–2208.

Zellner, A. (1988). Optimal information processing and Bayes's theorem. *The American Statistician*, *42*(4), 278–280.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., & Torr, P. H. (2015). Conditional random fields as recurrent neural networks. *Proceedings of the IEEE International Conference on Computer Vision*, 1529–1537.

Zheng, Z., Oh, J., & Singh, S. (2018). On learning intrinsic rewards for policy gradient methods. *Advances in Neural Information Processing Systems*, *31*.

Zhu, J., Chen, N., & Xing, E. P. (2014). Bayesian inference with posterior regularization and applications to infinite latent svms. *The Journal of Machine Learning Research*, *15*(1), 1799–1847.

Zhu, J., & Xing, E. P. (2009). Maximum entropy discrimination markov networks. *Journal of Machine Learning Research*, *10*(11).

Zhu, Y., Gao, T., Fan, L., Huang, S., Edmonds, M., Liu, H., Gao, F., Zhang, C., Qi, S., Wu, Y. N., et al. (2020). Dark, beyond deep: A paradigm shift to cognitive AI with humanlike common sense. *Engineering*, *6*(3), 310–345.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., & Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. *AAAI Conference on Artificial Intelligence*, *8*, 1433–1438.