# Standard Model For Machine Learning

Kaiwen Bian

December 3, 2024

Standard Model Paper.

## All About Constraint Solving

The standard model is a new perspective of looking at all types of traditional learning as a **constraint solving procedure**. RL is a search and optimization process under the reward constraint. We can also think of supervised learning not as finding patterns in data, but as a optimization or a search in the landscape of weights with the constraint of data. Think how we are doing projected gradient descent (Hard constraint) or regularized gradient descent (Lagrangian constraint) where we are projecting the steps onto a subspace following the constraint requirement. Then our constraint in supervised setting is just constraining (a strict constraint) all posible weight world steps to a data space.

## Panoramic Learning

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**. However, they all seems to fall apart when extending to generalized examples. Thus, a standardized ML formalism is highly needed where it 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.

This model try to study 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. 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. The list consists of three principled terms:

- **Experience term**: Offers a unified language to express arbitrary relevant information to supervise the learning.
- **Divergence term**: Measures the fitness of the target model to be learned.
- **Uncertainty term**: Regularizes the complexity of the system.

Hopefully this paradigm would it offer **guiding principles** for designing algorithmic approaches to new problems in a **mechanical way**. 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.

# Maximum Entropy: One Subset Case

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 $\theta$ and an **auxiliary distribution** $q$, over a **properly designed constraint space** $\mathcal{Q}$ where $q$ must reside:

$$\begin{aligned} \min_{q,\theta} \quad & \mathcal{L}(q, \theta) \\ \text{s.t.} \quad & q \in \mathcal{Q} \end{aligned}$$

We will see this type of formulation is one subset case of teh general model.

## Likelihood = Constraints Randomness

This **probabilistic and statistical learning 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, boosting, sparse learning, structure learning in **non-convex situation**.

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. We will start deriving the standard model's first component. We know MLE as:

$$\max_{\theta} \mathbb{E}_{x \sim D} \left[ \log p_\theta(x) \right]$$

$$\min_{\theta} -\mathbb{E}_{x \sim D} \left[ \log p_\theta(x) \right]$$

MLE is known to be intimately related to the maximum entropy principle, especially when this assumed distribution is the exponential distribution ($Z(\theta)$ is the normalization factor):

$$p_\theta(x) = \frac{\exp\{\theta \cdot T(x)\}}{Z(\theta)}$$

In MaxEnt, rather than assuming a specific parametric from of the target model distribution, denoted as $p(x)$, we instead impose constraints on the model distribution: in the supervised setting, the constraints require the expectation of the features $T(x)$ (sufficient statistics of the data) to be equal to the empirical expectation:

$$\mathbb{E}_p[T(x)] = \mathbb{E}_{x^* \sim \mathcal{D}}[T(x^*)]$$

The objective is to find a distribution that maximizes entropy, subject to matching certain empirical features. Intuitively, this is saying to choose the distribution that has the greatest amount of uncertainty or "spread," subject to certain known constraints. This results in a distribution that "covers" all possibilities fairly without going to much of what the data provides. **We will later see that this is just a special case of the general model**.

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

Using 2 Lagrangian solver ($\theta$ for MaxEnt constrain and $\mu$ for normalization constrain):

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

Setting zero with respect to $p$ using the Lagrangian, we get:

$$p(x) = \frac{\exp\{\theta \cdot T(x)\}}{Z(\theta)}$$

and pluging in:

$$\mathcal{L}(\theta) = \mathbb{E}_{x \sim \mathcal{D}}[\theta \cdot T(x)] - \log Z(\theta)$$

Which is exactly like MLE that we want to do earlier. 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.

## Untractable to Tractable

Unsupervised MLE is using latent variables and direct optimization of the marginal log-likelihood is typically intractable due to the summation over $y$.

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

However, it is solvable using EM and ELBO. Let $q(y|x)$ represent an arbitrary auxiliary distribution acting as a surrogate of the true posterior $p(y|x)$, known as the variational distribution:

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

We can then do E-step and M-step:

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

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

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

s

Overall, the MaxEnt 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 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 GP to wake-sleep, arise naturally as an approximation to the original solution. **We will see later that this can also be deemed a special case of the standard model**.

## Another Classic: Bayesian Inference

Interestingly, the the maximum entropy principle can also help to **reformulate Bayesian inference as a constraint optimization problem for MLE**. Bayesian approach for statistical inference treats the hypotheses (parameters $\theta$) to be inferred as random variables. Assuming a prior distribution $\pi(\theta)$ over the parameters, and considering a probabilistic model that defines a conditional distribution $p(x|\theta)$, then based on Bayes theorem:

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

and for continuous probabilistic functions:

$$p(\mathcal{D}) = \int_\theta \pi(\theta) \prod_{x^*\in\mathcal{D}} p(x^*|\theta)\, d\theta$$

By reformulating the statistical inference problem from the perspective of **information processing** and rediscovering Bayes' theorem as the **optimal information processing rule**, we can find its connections to MaxEnt.

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**. 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** ($q(\theta)$ is our structured representation of input beliefs about $\theta$) and **output** (posterior $p(\theta|\mathcal{D})$) information w.r.t. the output distribution. We need a way to measure how close $q(\theta)$ is to $p(\theta|\mathcal{D})$. We can down so by minimizing the KL divergence between $q(\theta)$ and $p(\theta|\mathcal{D})$:

$$\mathrm{KL}(q(\theta)\|p(\theta|\mathcal{D})) = \mathbb{E}_{q(\theta)}\left[\log q(\theta) - \log p(\theta|\mathcal{D})\right]$$

Expanding this expression using Bayes' theorem for $p(\theta|\mathcal{D})$, we get:

$$\mathrm{KL}(q(\theta)\|p(\theta|\mathcal{D})) = \mathbb{E}_{q(\theta)}\left[\log q(\theta) - \log \pi(\theta) - \sum_{x^*\in\mathcal{D}} \log p(x^*|\theta)\right] + \log p(\mathcal{D})$$

This KL divergence can be rewritten as:

$$\mathrm{KL}(q(\theta)\|p(\theta|\mathcal{D})) = \mathbb{E}_{q(\theta)}\left[\log q(\theta)\right] + \log p(\mathcal{D}) - \mathbb{E}_{q(\theta)}\left[\log \pi(\theta) + \sum_{x^*\in\mathcal{D}} \log p(x^*|\theta)\right]$$

Remember that the entropy term's definition is written as $\mathrm{H}(q) = -\mathbb{E}_{q(\theta)}\left[\log q(\theta)\right] = -\sum_\theta q(\theta)\log q(\theta)$. Thus, we can swap out the $\mathbb{E}_{q(\theta)}\left[\log q(\theta)\right]$ here.

$$\mathrm{KL}(q(\theta)\|p(\theta|\mathcal{D})) = -\mathrm{H}(q(\theta)) + \log p(\mathcal{D}) - \mathbb{E}_{q(\theta)}\left[\log \pi(\theta) + \sum_{x^*\in\mathcal{D}} \log p(x^*|\theta)\right]$$

Minimizing this KL divergence with respect to $q(\theta)$:

$$\begin{aligned} \min_{q(\theta)} \quad & -\mathrm{H}(q(\theta)) + \log p(\mathcal{D}) - \mathbb{E}_{q(\theta)}\left[\log \pi(\theta) + \textstyle\sum_{x^*\in\mathcal{D}} \log p(x^*|\theta)\right] \\ \text{s.t.} \quad & q(\theta) \in \mathcal{P}(\Theta) \end{aligned}$$

Here $\mathcal{P}(\Theta)$ is the space of all probability distributions over $\theta$. **This is probably one of the most similar formulation to the standard model, this is also a special case of the SE**.

## More Posterior Regularization

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. This is the posterior regularization or regularized Bayes:

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

Notice that $\log p(\mathcal{D})$ is a constant with regard to the optimization problem, so we can drop this term directly. In here, we also added constraints with $\xi$ being a vector of **slack variables** and $U(\xi)$ a **penalty function** with $\mathcal{Q}(\xi)$ being a subset of valid distributions over $\theta$ that **satisfy the constraints** determined by $\xi$.

The optimization problem is generally easy to solve when the penalty/constraints are convex and defined with respect to a linear operator (e.g., expectation) of the posterior $q$. For example, let $T(x^*;\theta)$ be a feature vector of data instance $x^* \in \mathcal{D}$, the constraint posterior set $\mathcal{Q}(\xi)$ can be defined as:

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

which bounds the feature expectations with $\xi$.

# Standard Model of Machine Learning

The previous example is demonstrating a special case of this standard formulation. This general formulation for learning a target model via a **constrained loss minimization program**. Without loss of generality:

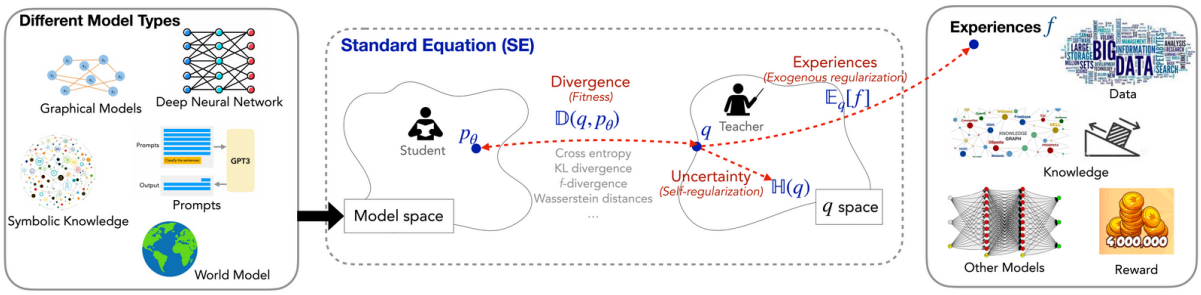
1. Let $t \in \mathcal{T}$ be the variable of interest, for example, the input-output pair $t = (x, y)$ in a prediction task, or the target variable $t = x$ in generative modeling.
2. Let $p_\theta(t)$ be the target model with parameters $\theta$ to be learned.
3. Let $q(t)$ be an auxiliary distribution or the true hidden distribution (In the previous Bayesian case the true hidden distribution is the input distribution and we try to minimize it against posterior).

The Standard Equation (SE) can be written as:

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

There are 3 key components:

- The uncertainty function $\mathbb{H}(\cdot)$ that controls the compactness of the output model, for example, by regulating the amount of allowed randomness while trying to fit experience.
- The divergence function $\mathbb{D}(\cdot,\cdot)$ that measures the distance between the target model to be trained and the auxiliary model, facilitating a teacher–student mechanism as shown below.
- The experience function, introduced by a penalty term $U(\xi)$, which incorporates the set of 'experience functions' $f_k^{(\theta)}$ that represent external experience of various kinds for training the target model.
- The hyperparameters $\alpha, \beta \geq 0$ that enable trade-offs between these components.




## Experience Function

All diverse forms of experience that can be utilized for model training (data examples, constraints, logical rules, rewards, and adversarial discriminators) can be encoded as an experience function. It provides a unified language to express all **exogenous information about the target model**. They all contribute to the optimization objective via the **penalty term** $U(\xi)$ over **slack variables** $\xi \in \mathbb{R}^K$ 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**.

## Divergence Function

Divergence function $\mathbb{D}(q, p_\theta)$ measures the 'quality' of the target model $p_\theta$ in terms of its distance with the auxiliary hidden true model $q$. The divergence function $\mathbb{D}(\cdot,\cdot)$ determines the specific optimization problem.

## Uncertainty Function

Uncertainty function $\mathbb{H}(q)$ describes the uncertainty of the auxiliary distribution $q$ and thus controls the complexity of the learning system. One should pick the most uncertain solution among those that fit all experience.

## All Subset of SE

The auxiliary hidden distribution $q$ **relaxes** the learning problem of $p_\theta$, originally only over $\theta$, to be now **alternating between** $q$ **and** $\theta$. Here $q$ acts as a **conduit between**

**the exogenous experience and the target model**:

- It subsumes the experience (by maximizing the expected $f$ value).
- It also passes incrementally to the target model (by minimizing the divergence $\mathbb{D}$).

Thus, achieving balanced learning.

**Notice that this is a standard format that can derive any of the classical algorithm and we can use numerous other algorithm to solve these problems**. There are also great flexibility of choosing the surrogate distribution $q$, ranging from the principled variational approximations for the target distribution in a properly relaxed space (e.g., mean fields), to the arbitrary neural network-based inference networks that are highly expressive and easy to compute.

Classical examples including EM, variational EM, wake-sleep, forward and backward propagation 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 $\mathbb{D}$.

## Teacher-Student (EM) Optimization of SE

The SE formulation is super good, but how do we optimize it? Again, we can do the classical EM update format to find the hidden $q$ distribution (**parameters doesn't matter, the hidden distribution is what impact all important information, then we just optimize parameter with regard to this distribution is fine**);

1. Uncertainty as $\mathbb{H}(q) = -\mathbb{E}_q[\log q]$ (Shannon Entropy). Maximization of this process is implictly represented in
2. Divergence as $\mathbb{D}(q, p_\theta) = -\mathbb{E}_q[\log p_\theta]$ (Cross Entropy).
3. Experience as $f(t)$

We will derive a EM formulation from the SE. Recall that SE is defined as:

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

We can do uncontraied optimization duality of it. The Lagrangian $\mathcal{L}$ becomes:

$$\mathcal{L}(q, \theta, \xi, \lambda) = -\alpha\mathbb{H}(q) + \beta\mathbb{D}(q, p_\theta) + U(\xi) + \sum_{k=1}^{K} \lambda_k \left(-\mathbb{E}_q[f_k^{(\theta)}] - \xi_k\right)$$

To find the optimal form of $q$, we take the derivative of $\mathcal{L}$ with respect to $q$ and set it to zero (this is taking a **Functional derivative**), a common approach in variational

| **Experience Type** | **Experience Function** $f$ | **Div.** $\mathbb{D}$ | $\alpha$ | $\beta$ | **Algorithm** |
|---|---|---|---|---|---|
| Data Instances | $f_{\text{data}}(x; \mathcal{D})$ | CE | 1 | 1 | Unsupervised MLE |
| Data Instances | $f_{\text{data}}(x, y; \mathcal{D})$ | CE | 1 | $\epsilon$ | Supervised MLE |
| Self-supervised | $f_{\text{data-self}}(x, y; \mathcal{D})$ | CE | 1 | $\epsilon$ | Self-supervised MLE |
| Re-weighting | $f_{\text{data-w}}(t; \mathcal{D})$ | CE | 1 | $\epsilon$ | Data Re-weighting |
| Augmentation | $f_{\text{data-aug}}(t; \mathcal{D})$ | CE | 1 | $\epsilon$ | Data Augmentation |
| Active Learning | $f_{\text{active}}(x, y; \mathcal{D})$ | CE | 1 | $\epsilon$ | Active Learning |
| Knowledge | $f_{\text{rule}}(x, y)$ | CE | 1 | 1 | Posterior Regularization |
| Unified EM | $f_{\text{rule}}(x, y)$ | CE | $\mathbb{R}$ | 1 | Unified EM |
| Policy Gradient | $\log Q^{\theta}(x, y)$ | CE | 1 | 1 | Policy Gradient |
| + Intrinsic Reward | $\log Q^{\theta}(x, y) + Q^{in,\theta}(x, y)$ | CE | 1 | 1 | Intrinsic Reward |
| RL as Inference | $Q^{\theta}(x, y)$ | CE | $\rho > 0$ | $\rho > 0$ | RL as Inference |
| Model Mimicking | $f^{\text{mimicking}}_{\text{model}}(x, y; \mathcal{D})$ | CE | 1 | $\epsilon$ | Knowledge Distillation |
| GAN (Vanilla) | Binary Classifier | JSD | 0 | 1 | Vanilla GAN |
| GAN (f-GAN) | Discriminator | f-divergence | 0 | 1 | f-GAN |
| WGAN | 1-Lipschitz Disc. | $W_1$ | 0 | 1 | WGAN |
| PPO-GAN | 1-Lipschitz Disc. | KL | 0 | 1 | PPO-GAN |
| Online | $f_{\tau}(t)$ | CE | $\rho > 0$ | $\rho > 0$ | Multiplicative Weights |

Table 1: Specifications of SE Components for Different Algorithms

inference. Since $\mathbb{H}(q) = -\mathbb{E}_q[\log q]$ and $\mathbb{D}(q, p_\theta) = -\mathbb{E}_q[\log p_\theta]$, the relevant terms in the Lagrangian with respect to $q$ are:

$$-\alpha\mathbb{H}(q) + \beta\mathbb{D}(q, p_\theta) + \sum_{k=1}^{K} \lambda_k \mathbb{E}_q[f_k^{(\theta)}]$$

Taking the derivative of this expression with respect to $q$ and setting it to zero yields:

$$\alpha \log q(t) = \beta \log p_\theta(t) + \sum_{k=1}^{K} \lambda_k f_k^{(\theta)}(t)$$

Rearranging terms gives:

$$q(t) = \exp\left(\frac{\beta \log p_\theta(t) + \sum_{k=1}^{K} \lambda_k f_k^{(\theta)}(t)}{\alpha}\right)$$

To ensure $q$ remains a valid probability distribution (summing to 1), we introduce a normalization factor $Z$:

$$Z = \sum_t \exp\left\{\frac{\beta \log p_{\theta^{(n)}}(t) + f(t)}{\alpha}\right\}.$$

Then we have:

$$q(t) = \frac{\exp\left(\frac{\beta \log p_\theta(t) + f(t)}{\alpha}\right)}{Z}$$

Here, we denote $f(t) = \sum_{k=1}^{K} \lambda_k f_k^{(\theta)}(t)$, representing the combined experience from multiple sources weighted by their respective multipliers.

**Now we have a ideal $q$ distribution that we can work with, such $q$ distribution comes from the SE.** Now, in the M-step, we are just doing a **MLE** under such $q$ distribution, which is the equivalent with Minimizing the **KL**.

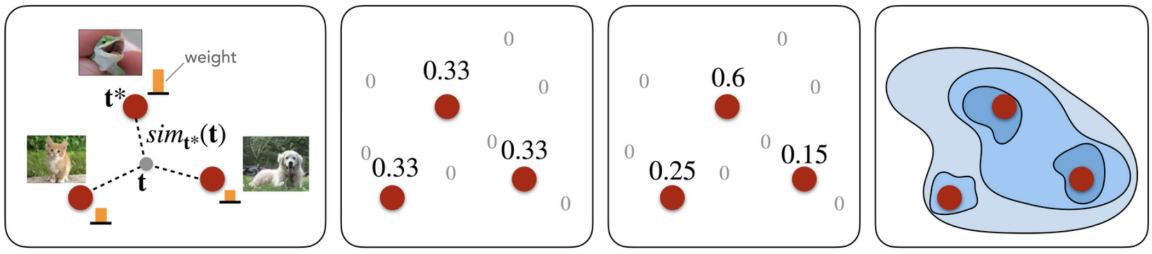
$$\begin{aligned}
\text{Teacher (E-Step):} \quad & q^{(n+1)}(t) = \exp\left\{\frac{\beta \log p_{\theta^{(n)}}(t) + f(t)}{\alpha}\right\} / Z \\
\text{Student (M-Step):} \quad & \theta^{(n+1)} = \operatorname*{argmax}_\theta \mathbb{E}_{q^{(n+1)}(t)}\big[\log p_\theta(t)\big]
\end{aligned}$$

We will use this as a running example for the next section. **The transformation from SE to more classical algorithm is not that trivial.**

# Experience Function

Different choices of $f(t)$ result in learning algorithms applied to different problems. With particular choices, the standard equation rediscovers a wide array of well-known algorithms. **We will only use the most interesting examples here to illustrate the point.**

## Supervised Data Instances




For an arbitrary configuration $(x_0, y_0)$, its probability $p_d(x_0, y_0)$ under the data distribution can be seen as measuring the **expected similarity** between $(x_0, y_0)$ and true data samples $(x^*, y^*)$, and be written as

$$p_d(x_0, y_0) = \mathbb{E}_{p_d(x^*,y^*)} \left[\mathbb{I}_{(x^*,y^*)}(x_0, y_0)\right].$$

Here the similarity measure is $\mathbb{I}_{(x^*,y^*)}(x, y)$, an indicator function that takes the value 1 if $(x, y)$ equals $(x^*, y^*)$ and 0 otherwise. The experience function would thus be defined as:

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

and plug in to the teacher model of expecting $q$. Notice that if **we don't care about the divergence** ($\beta = 0$), **we don't care about the true distribution**, rather just fit to data (**what MLE is doing**), then we have:

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

Then we maximize this (instead of true q-distribution, we directly optimized against the data distribution):

$$\max_\theta \mathbb{E}_{t\sim\tilde{p}_d(x,y)} \left[\log p_\theta(t)\right]$$

This is exactly the same as definition of MLE.

$$\max_\theta \mathbb{E}_{x^*\sim D} \left[\log p_\theta(x^*)\right]$$

## Self-Supervised Data Instances

Given an observed data instance $t^* \in \mathcal{D}$ in general, one could potentially derive various **supervision** signals based on the **structures** of the data and the target model. We can could apply a "**split**" function that artificially partitions $t^*$ into two parts $(x^*, y^*) = \mathit{split}(t^*)$. Then the two parts are treated as the **input and output** for the properly designed target model $p_\theta(x, y)$:

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

Now the target variable $y$ is not costly obtained labels or annotations, but rather part of the **massively available data instances**. The paradigm of treating part of an observed instance as the prediction target is called '**self-supervised**' learning.

## Expected Reward Based Experience

We now consider a very different learning setting commonly seen in robotic control and other sequential decision making problems. Experience is gained by the agent interacting with external environment and collecting feedback in the form of rewards. Let's say $t = (x, y) = (s, a)$ is the state-action pair. The first way to use the reward signals as the experience is by defining the experience function as the **logarithm of the expected future reward**, which leads to the classical **policy gradient** algorithm.:

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

Where:

$$Q^\theta(x, y) = \mathbb{E}\left[ \sum_{t=0}^{\infty} \gamma^t r_t \mid x_0 = x, y_0 = y \right]$$

With $\alpha = \beta = 1$, we would arrive at policy gradient.

**In E-step**, we care about divergent to true distribution fully (remember that the $q$ distribution is derived from the SE):

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

**In M-step**, we update $\theta$ with the gradient. Different from traditional MLE, we want to optimize not just the parameter to fit the (state, action) pair data, but also added extra reward to optimize. Thus, we have two gradient at hand, one with the log-likelihood of the data under the $q$-distribution, and the second is gradient of the reward function directly.

$$\nabla_\theta \mathbb{E}_{q^{(n)}(x,y)} \left[ \log p_\theta(x, y) + f^\theta_{\text{reward}}(x, y) \right] \Big|_{\theta = \theta^{(n)}}$$

We can use the log-derivative trick $g \nabla \log g = \nabla g$. We can simplify:

$$\nabla_\theta \log g(\theta) = \frac{\nabla_\theta g(\theta)}{g(\theta)},$$

we can apply it as follows:

$$\mathbb{E}_{q^{(n)}(x,y)} \left[\nabla_\theta \log p_\theta(x,y)\right] = \frac{1}{Z} \sum_{x,y} p_{\theta^{(n)}}(x,y) Q^{\theta^{(n)}}(x,y) \nabla_\theta \log p_\theta(x,y).$$

When substituting $q$ as:

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

Similarly, we can rewrite the second term as:

$$\mathbb{E}_{q^{(n)}(x,y)} \left[\nabla_\theta \log Q^\theta(x,y)\right] = \frac{1}{Z} \sum_{x,y} p_{\theta^{(n)}}(x,y) Q^{\theta^{(n)}}(x,y) \nabla_\theta \log Q^\theta(x,y).$$

Combining the two terms:

$$\frac{1}{Z} \sum_{x,y} p_{\theta^{(n)}}(x,y) Q^{\theta^{(n)}}(x,y) \left(\nabla_\theta \log p_\theta(x,y) + \nabla_\theta \log Q^\theta(x,y)\right).$$

We can rewrite this in conditional format to separate out the state and action (to match policy gradient formula). **This is the highly non-trivial part.**

$$= \frac{1}{Z} \cdot \sum_x p_0(x) \nabla_\theta \sum_y p_\theta(y|x) Q^\theta(x,y) \Big|_{\theta=\theta^{(n)}}$$

Thus, we can say that:

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

The final form is exactly the policy gradient up to a multiplication factor $1/Z$.