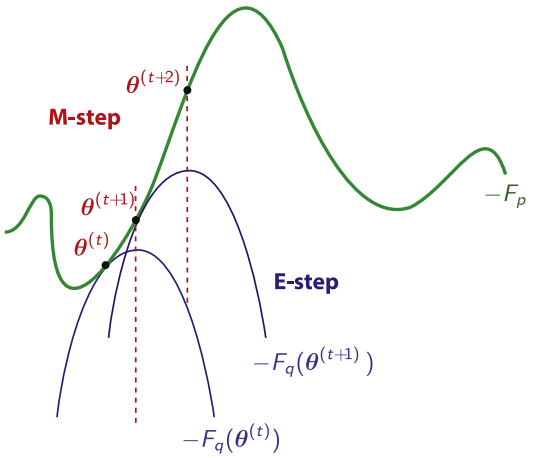

Fig. 59. Convergence of EM algorithm. Starting from $\boldsymbol{\theta}^{(t)}$, E-step (blue) establishes $-F_q(\boldsymbol{\theta}^{(t)})$ which is always a lower bound of $-F_p := \langle \log p(\boldsymbol{x}|\boldsymbol{\theta})\rangle_{P_{\mathbf{x}}}$ (green). M-step (red) is then applied to update the parameter, yielding $\boldsymbol{\theta}^{(t+1)}$. The updated parameter $\boldsymbol{\theta}^{(t+1)}$ is then used to construct $-F_q(\boldsymbol{\theta}^{(t+1)})$ in the subsequent E-step. M-step is performed again to update the parameter, etc.

and unobserved). The name "M-step" is intuitive since the parameters $\boldsymbol{\theta}$ are found by maximizing $-F_q(\boldsymbol{\theta})$. The name "E-step" comes from the fact that one usually does not need to construct the probability of missing data explicitly, but rather need only compute the "expected" sufficient statistics over these data, cf. Fig. 59.

On the practical side, EM has been demonstrated to be extremely useful in parameter estimation, particularly in hidden Markov models and Bayesian networks (see, for example, (Barber, 2012; Wainwright et al., 2008)). Some of the authors have used EM in biophysics, to design algorithms which establish the equivalence of niche theory and the Minimum Environmental Perturbation Principle (Marsland et al., 2019). One of the striking advantages of EM is that it is conceptually simple and easy to implement (see Notebook 16). In many cases, implementation of EM is guaranteed to increase the likelihood monotonically, which could be a perk during debugging. For readers interested in an overview on applications of EM, we recommend (Do and Batzoglou, 2008).

Finally for advanced readers familiar with the physics of disordered systems, we note that it is possible to construct a one-to-one dictionary between EM for latent variable models and the MFT of spin systems with quenched disorder. In a disordered spin systems, the Ising couplings $\boldsymbol{J}$ are commonly taken to be quenched random variables drawn from some underlying probability distribution. In the EM procedure, the quenched disorder is provided by the observed data points $\boldsymbol{x}$ which are drawn from some underlying probability distribution that characterizes the data. The spins $\boldsymbol{s}$ are like the hidden or latent variables $\boldsymbol{z}$. Similar analogues can be found for all the variational MFT quantities (see Table 1). This striking correspondence offers a glimpse into the deep connection between statistical mechanics and unsupervised latent variable models – a connection that we will repeatedly exploit to gain more intuition for the energy-based unsupervised models considered in the next few chapters.

15. Energy based models: Maximum entropy (MaxEnt) principle, generative models, and Boltzmann learning

Most of the models discussed in the previous sections (e.g. linear and logistic regression, ensemble models, and supervised neural networks) are *discriminative* – they are designed to perceive differences between groups or categories of data. For example, recognizing differences between images of cats and images of dogs allows a discriminative model to label an image as "cat" or "dog". Discriminative models form the core techniques of most supervised learning methods. However, discriminative methods have several limitations. First, like all supervised learning methods, they require labeled data. Second, there are tasks that discriminative approaches simply cannot accomplish, such as drawing new examples from an unknown probability distribution. A model that can learn to represent and sample from a probability distribution is called *generative*. For example, a generative model for images would learn to draw new examples of cats and dogs given a dataset of images of cats and dogs. Similarly, given samples generated from one phase of an Ising model we may want to generate new samples from that phase. Such tasks are clearly beyond the scope of discriminative models like the ensemble models and DNNs discussed so far in the review. Instead, we must turn to a new class of machine learning methods.

The goal of this section is to introduce the reader to *energy-based* generative models. As we will see, energy-based models are closely related to the kinds of models commonly encountered in statistical physics. We will draw upon many techniques that have their origin in statistical mechanics (e.g. Monte-Carlo methods). The section starts with a brief overview of generative models, highlighting the similarities and differences with the supervised learning methods encountered in earlier sections. Next, we introduce perhaps the simplest kind of generative models — Maximum Entropy (MaxEnt) models. MaxEnt models have no latent (or hidden) variables, making them ideal for introducing the key concepts

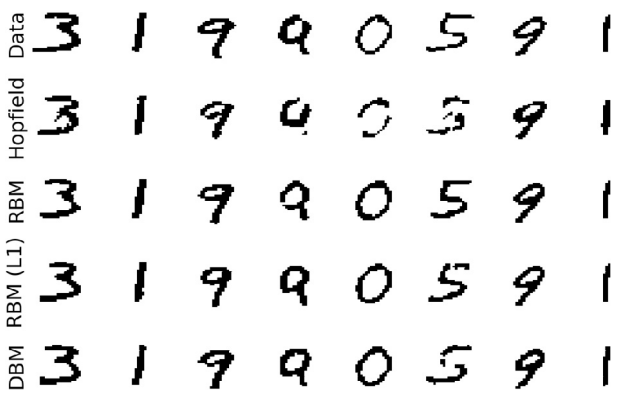

Fig. 60. Examples of handwritten digits (“reconstructions”) generated using various energy-based models using the powerful *Paysage* package for unsupervised learning. Examples from top to bottom are: the original MNIST database, an RBM with Gaussian units which is equivalent to a Hopfield Model, a Restricted Boltzmann Machine (RBM), a RBM with an L_1 penalty for regularization, and a Deep Boltzmann Machine (DBM) with 3 layers. All models have 200 hidden units. See Section 16 and corresponding notebook for details.

and tools that underlie energy-based generative models. We then present an extended discussion of how to train energy-based models. Much of this discussion will also be applicable to more complicated energy-based models such as Restricted Boltzmann Machines (RBMs) and the deep models discussed in the next section.

15.1. An overview of energy-based generative models

Generative models are a machine learning technique that allows to learn how to generate new examples similar to those found in a training dataset. The core idea of most generative models is to learn a parametric model for the probability distribution from which the data was drawn. Once we have learned a model, we can generate new examples by sampling from the learned generative model (see Fig. 60). As in statistical physics, this sampling is often done using Markov Chain Monte Carlo (MCMC) methods. A review of MCMC methods is beyond the scope of this discussion: for a concise and beautiful introduction to MCMC-inspired methods that bridges both statistical physics and ML the reader is encouraged to consult Chapters 29–32 of David MacKay’s book (MacKay, 2003) as well as the review by Michael I. Jordan and collaborators (Andrieu et al., 2003).

The added complexity of learning models directly from samples introduces many of the same fundamental tensions we encountered when discussing discriminative models. The ability to generate new examples requires models to be able to “generalize” beyond the examples they have been trained on, that is to generate new samples that are not samples of the training set. The models must be expressive enough to capture the complex correlations present in the underlying data distribution, but the amount of data we have is finite which can give rise to overfitting.

In practice, most generative models that are used in machine learning are flexible enough that, with a sufficient number of parameters, they can approximate any probability distribution. For this reason, there are three axes on which we can differentiate classes of generative models:

- The first axis is how easy the model is to train — both in terms of computational time and the complexity of writing code for the algorithm.
- The second axis is how well the model generalizes from the training set to the test set.
- The third axis is which characteristics of the data distribution the model is capable of and focuses on capturing.

All generative models must balance these competing requirements and generative models differ in the tradeoffs they choose. Simpler models capture less structure about the underlying distributions but are often easier to train. More complicated models can capture this structure but may overfit to the training data.

One of the fundamental reasons that energy-based models have been less widely-employed than their discriminative counterparts is that the training procedure for these models differs significantly from those for supervised neural networks models. Though both employ gradient-descent based procedures for minimizing a cost function (one common choice for generative models is the negative log-likelihood function), energy-based models do not use backpropagation (see Section 9.3) and automatic differentiation for computing gradients. Rather, one must turn to ideas inspired by MCMC based methods in physics and statistics that sometimes go under the name “Boltzmann Learning” (discussed below). As a result, training energy-based models requires additional tools that are not immediately available in packages such as PyTorch and TensorFlow.

The open-source package – *Paysage* – that is built on top of PyTorch bridges this gap by providing the toolset for training energy-based models (Paysage is maintained by Unlearn.AI – a company affiliated with two of the authors (CKF and PM)). Paysage makes it easy to quickly code and deploy energy-based models such as Restricted Boltzmann Machines

(RBMs) and Stacked RBMs – a "deep" unsupervised model. The package includes unpublished training methods that significantly improve the training performance, can be applied with various datatypes, and can be employed on GPUs. We make use of this package extensively in the next two sections and the accompanying Python notebooks. For example, Fig. 60 (and the accompanying Notebook 17) show how the Paysage package can be used to quickly code and train a variety of energy-based models on the MNIST handwritten digit dataset.

Finally, we note that generative models at their most basic level are complex parametrizations of the probability distribution the data is drawn from. For this reason, generative models can do much more than just generate new examples. They can be used to perform a multitude of other tasks that require sampling from a complex probability distribution including "de-noising", filling in missing data, and even discrimination (Hinton, 2012). The versatility of generative models is one of the major appeals of these unsupervised learning methods.

15.2. Maximum entropy models: the simplest energy-based generative models

Maximum Entropy (MaxEnt) models are one of the simplest classes of energy-based generative models. MaxEnt models have their origin in a series of beautiful papers by Jaynes that reformulated statistical mechanics in information theoretic terms (Jaynes, 1957a,b). Recently, the flood of new, large scale datasets has resulted in a resurgence of interest in MaxEnt models in many fields including physics (especially biological physics), computational neuroscience, and ecology (Elith et al., 2011; Schneidman et al., 2006; Weigt et al., 2009). MaxEnt models are often presented as the class of generative models that make the least assumptions about the underlying data. However, as we have tried to emphasize throughout the review, all ML and statistical models require assumptions, and MaxEnt models are no different. Overlooking this can sometimes lead to misleading conclusions, and it is important to be cognizant of these implicit assumptions (Aitchison et al., 2016; Schwab et al., 2014).

15.2.1. MaxEnt models in statistical mechanics

MaxEnt models were introduced by E. T. Jaynes in a two-part paper in 1957 entitled "Information theory and statistical mechanics" (Jaynes, 1957a,b). In these incredible papers, Jaynes showed that it was possible to re-derive the Boltzmann distribution (and the idea of generalized ensembles) entirely from information theoretic arguments. Quoting from the abstract, Jaynes considered "statistical mechanics as a form of statistical inference rather than as a physical theory" (portending the close connection between statistical physics and machine learning). Jaynes showed that the Boltzmann distribution could be viewed as resulting from a statistical inference procedure for learning probability distributions describing physical systems where one only has partial information about the system (usually the average energy).

The key quantity in MaxEnt models is the information theoretic, or Shannon, entropy, a concept introduced by Shannon in his landmark treatise on information theory (Shannon, 1949). The Shannon entropy quantifies the statistical uncertainty one has about the value of a random variable $\mathbf{x}$ drawn from a probability distribution $p(\mathbf{x})$. The Shannon entropy of the distribution is defined as

$$S_p = -\mathrm{Tr}_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x}) \tag{179}$$

where the trace is a sum/integral over all possible values a variable can take. Jaynes showed that the Boltzmann distribution follows from the Principle of Maximum Entropy. A physical system should be described by the probability distribution with the largest entropy subject to certain constraints (often provided by measuring the average value of conserved, extensive quantities such as the energy, particle number, etc.) The principle uniquely specifies a procedure for parametrizing the functional form of the probability distribution. Once we have specified and learned this form we can, of course, generate new examples by sampling this distribution.

Let us illustrate how this works in more detail. Suppose that we have chosen a set of functions $\{f_i(\mathbf{x})\}$ whose average value we want to fix to some observed values $\langle f_i \rangle_{\text{obs}}$. The Principle of Maximum Entropy states that we should choose the distribution $p(\mathbf{x})$ with the largest uncertainty (i.e. largest Shannon entropy S_p), subject to the constraints that the model averages match the observed averages:

$$\langle f_i \rangle_{\text{model}} = \int \mathrm{d}\mathbf{x} f_i(\mathbf{x}) p(\mathbf{x}) = \langle f_i \rangle_{\text{obs}}. \tag{180}$$

We can formulate the Principle of Maximum Entropy as an optimization problem using the method of Lagrange multipliers by minimizing:

$$\mathcal{L}[p] = -S_p + \sum_i \lambda_i \left(\langle f_i \rangle_{\text{obs}} - \int \mathrm{d}\mathbf{x} f_i(\mathbf{x}) p(\mathbf{x}) \right) + \gamma \left(1 - \int \mathrm{d}\mathbf{x} p(\mathbf{x}) \right),$$

where the first set of constraints enforce the requirement for the averages and the last constraint enforces the normalization that the trace over the probability distribution equals one. We can solve for $p(\mathbf{x})$ by taking the functional derivative

and setting it to zero

$$0 = \frac{\delta \mathcal{L}}{\delta p} = (\log p(\mathbf{x}) + 1) - \sum_i \lambda_i f_i(\mathbf{x}) - \gamma.$$

The general form of the maximum entropy distribution is then given by

$$p(\mathbf{x}) = \frac{1}{Z}\, \mathrm{e}^{\sum_i \lambda_i f_i(\mathbf{x})} \tag{181}$$

where $Z(\lambda_i) = \int \mathrm{d}\mathbf{x}\, \mathrm{e}^{\sum_i \lambda_i f_i(\mathbf{x})}$ is the partition function.

The maximum entropy distribution is clearly just the usual Boltzmann distribution with energy $E(\mathbf{x}) = -\sum_i \lambda_i f_i(\mathbf{x})$. The values of the Lagrange multipliers are chosen to match the observed averages for the set of functions $\{f_i(\mathbf{x})\}$ whose average value is being fixed:

$$\langle f_i \rangle_{\text{model}} = \int \mathrm{d}\mathbf{x} p(\mathbf{x}) f_i(\mathbf{x}) = \frac{\partial \log Z}{\partial \lambda_i} = \langle f_i \rangle_{\text{obs}}. \tag{182}$$

In other words, the parameters of the distribution can be chosen such that

$$\partial_{\lambda_i} \log Z = \langle f_i \rangle_{\text{data}}. \tag{183}$$

To gain more intuition for the MaxEnt distribution, it is helpful to relate the Lagrange multipliers to the familiar thermodynamic quantities we use to describe physical systems (Jaynes, 1957a). Our $\mathbf{x}$ denotes the microscopic state of the system, i.e. the MaxEnt distribution is a probability distribution over microscopic states. However, in thermodynamics we only have access to average quantities. If we know only the average energy $\langle E(\mathbf{x}) \rangle_{\text{obs}}$, the MaxEnt procedure tells us to maximize the entropy subject to the average energy constraint. This yields

$$p(\mathbf{x}) = \frac{1}{Z}\, \mathrm{e}^{-\beta E(\mathbf{x})}, \tag{184}$$

where we have identified the Lagrange multiplier conjugate to the energy $\lambda_1 = -\beta = 1/k_B T$ with the (negative) inverse temperature. Now, suppose we also constrain the particle number $\langle N(x) \rangle_{\text{obs}}$. Then, an almost identical calculation yields a MaxEnt distribution of the functional form

$$p(\mathbf{x}) = \frac{1}{Z}\, \mathrm{e}^{-\beta(E(\mathbf{x}) - \mu N(\mathbf{x}))}, \tag{185}$$

where we have rewritten our Lagrange multipliers in the familiar thermodynamic notation $\lambda_1 = -\beta$ and $\lambda_2 = \mu/\beta$. Since this is just the Boltzmann distribution, we can also relate the partition function in our MaxEnt model to the thermodynamic free-energy via $F = -\beta^{-1} \log Z$. The choice of which quantities to constrain is equivalent to working in different thermodynamic ensembles.

15.2.2. From statistical mechanics to machine learning

The MaxEnt idea also provides a general procedure for learning a generative model from data. The key difference between MaxEnt models in (theoretical) physics and ML is that in ML we have no direct access to observed values $\langle f_i \rangle_{\text{obs}}$. Instead, these averages must be directly estimated from data (samples). To denote this difference, we will call empirical averages calculated from data as $\langle f_i \rangle_{\text{data}}$. We can think of MaxEnt as a statistical inference procedure simply by replacing $\langle f_i \rangle_{\text{obs}}$ by $\langle f_i \rangle_{\text{data}}$ above.

This subtle change has important implications for training MaxEnt models. First, since we do not know these averages exactly, but must estimate them from the data, our training procedures must be careful not to overfit to the observations (our samples might not be reflective of the true values of these statistics). Second, the averages of certain functions f_i are easier to estimate from limited data than others. This is often an important consideration when formulating which MaxEnt model to fit to the data. Finally, we note that unlike in physics where conservation laws often suggest the functions f_i whose averages we hold fix, ML offers no comparable guide for how to choose the f_i we care about. For these reasons, choosing the $\{f_i\}$ is often far from straightforward. As a final point, we note that here we have presented a physics-based perspective for justifying the MaxEnt procedure. We mention in passing that the MaxEnt in ML is also closely related to ideas from Bayesian inference (Jaynes, 1968, 2003) and this latter point of view is more common in discussions of MaxEnt in the statistics and ML literature.

15.2.3. Generalized Ising models from MaxEnt

The form of a MaxEnt model is completely specified once we choose the averages $\{f_i\}$ we wish to constrain. One common choice often used in MaxEnt modeling is to constrain the first two moments of a distribution. When our random variables $\mathbf{x}$ are continuous, the corresponding MaxEnt distribution is a multi-dimensional Gaussian. If the $\mathbf{x}$ are binary (discrete), then the corresponding MaxEnt distribution is a generalized Ising (Potts) model with all-to-all couplings.

To see this, consider a random variable $\mathbf{x}$ with first and second moments $\langle x_i \rangle_{\text{data}}$ and $\langle x_i x_j \rangle_{\text{data}}$, respectively. According to the Principle of Maximum Entropy, we should choose to model this variable using a Boltzmann distribution with

constraints on the first and second moments. Let a_i be the Lagrange multiplier associated with $\langle x_i \rangle_{\text{data}}$ and $J_{ij}/2$ be the Lagrange multiplier associated with $\langle x_i x_j \rangle_{\text{data}}$. Using Eq. (182), it is easy to verify that the energy function

$$E(\mathbf{x}) = -\sum_i a_i x_i - \frac{1}{2}\sum_{ij} J_{ij} x_i x_j \tag{186}$$

satisfies the above constraints.

Partition functions for maximum entropy models are often intractable to compute. Therefore, it is helpful to consider two special cases where $\mathbf{x}$ has different support (different kinds of data). First, consider the case that the random variables $\mathbf{x} \in \mathbb{R}^n$ are real numbers. In this case we can compute the partition function directly:

$$Z = \int \mathrm{d}\mathbf{x}\, \mathrm{e}^{\mathbf{a}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T J \mathbf{x}} = \sqrt{(2\pi)^n \det J^{-1}}\, \mathrm{e}^{-\frac{1}{2}\mathbf{a}^T J^{-1}\mathbf{a}}. \tag{187}$$

The resulting probability density function is,

$$\begin{aligned} p(\mathbf{x}) &= Z^{-1}\mathrm{e}^{-E(\mathbf{x})} \\ &= \frac{1}{\sqrt{(2\pi)^n \det J^{-1}}}\, \mathrm{e}^{\frac{1}{2}\mathbf{a}^T J^{-1}\mathbf{a} + \mathbf{a}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T J \mathbf{x}} \\ &= \frac{1}{\sqrt{(2\pi)^n \det \Sigma}}\, \mathrm{e}^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})}, \end{aligned} \tag{188}$$

where $\boldsymbol{\mu} = -J^{-1}\mathbf{a}$ and $\Sigma = -J^{-1}$. This, of course, is the normalized, multi-dimensional Gaussian distribution.

Second, consider the case that the random variable $\mathbf{x}$ is binary with $x_i \in \{-1, +1\}$. The energy function takes the same form as Eq. (186), but the partition function can no longer be computed in a closed form. This model is known as the Ising model in the physics literature, and is often called a Markov Random Field in the machine learning literature. It is well known to physicists that calculating the partition function for the Ising Model is intractable. For this reason, the best we can do is estimate it using numerical techniques such MCMC methods or approximate methods like variational MFT methods, see Section 14. Finally, we note that in ML it is common to use binary variables which take on values in $x_i \in \{0, 1\}$ rather than $\{\pm 1\}$. This can sometimes be a source of confusion when translating between ML and physics literatures and can lead to confusion when using ML packages for physics problems.

15.3. Cost functions for training energy-based models

The MaxEnt procedure gives us a way of parametrizing an energy-based generative model. For any energy-based generative model, the energy function $E(\mathbf{x}, \{\theta_i\})$ depends on some parameters θ_i – couplings in the language of statistical physics – that must be inferred directly from the data. For example, for the MaxEnt models the $\{\theta_i\}$ are just the Lagrange multipliers $\{\lambda_i\}$ introduced in the last section. The goal of the training procedure is to use the available training data to fit these parameters.

Like in many other ML techniques, we will fit these couplings by minimizing a cost function using stochastic gradient descent (cf. Section 4). Such a procedure naturally separates into two parts: choosing an appropriate cost function, and calculating the gradient of the cost function with respect to the model parameters. Formulating a cost function for generative models is a little bit trickier than for supervised, discriminative models. The objective of discriminative models is straightforward — predict the label from the features. However, what we mean by a "good" generative model is much harder to define using a cost function. We would like the model to generate examples similar to those we find in the training dataset. However, we would also like the model to be able to generalize — we do not want the model to reproduce "spurious details" that are particular to the training dataset. Unlike for discriminative models, there is no straightforward idea like cross-validation on the data labels that neatly addresses this issue. For this reason, formulating cost functions for generative models is subtle and represents an important and interesting open area of research.

Calculating the gradients of energy-based models also turns out to be different than for discriminative models, such as deep neural networks. Rather than relying on automatic differentiation techniques and backpropagation (see Section 9.3), calculating the gradient requires drawing on intuitions from MCMC-based methods. Below, we provide an in-depth discussion of Boltzmann learning for energy-based generative models, focusing on MaxEnt models. We put the emphasis on training procedures that generalize to more complicated generative models with latent variables such as RBMs discussed in the next section. Therefore, we largely ignore the incredibly rich physics-based literature on fitting Ising-like MaxEnt models (see the recent reviews (Baldassi et al., 2018; Nguyen et al., 2017) and references therein).

15.3.1. Maximum likelihood

By far the most common approach used for training a generative model is to maximize the log-likelihood of the training dataset. Recall, that the log-likelihood characterizes the log-probability of generating the observed data using our generative model. By choosing the negative log-likelihood as the cost function, the learning procedure tries to find parameters that maximize the probability of the data. This cost function is intuitive and has been the work-horse of most

generative modeling. However, we note that the Maximum Likelihood estimation (MLE) procedure has some important limitations that we will return to in Section 17.

In what follows, we employ a general notation that is applicable to all energy-based models, not just the MaxEnt models introduced above. The reason for this is that much of this discussion does not rely on the specific form of the energy function but only on the fact that our generative model takes a Boltzmann form. We denote the generative model by the probability distribution $p_\theta(\mathbf{x})$ and its corresponding partition function by $\log Z(\{\theta_i\})$. In MLE, the parameters of the model are fit by maximizing the log-likelihood:

$$\begin{aligned}\mathcal{L}(\{\theta_i\}) &= \langle \log\left(p_\theta(\mathbf{x})\right)\rangle_{\text{data}} \\ &= -\langle E(\mathbf{x};\{\theta_i\})\rangle_{\text{data}} - \log Z(\{\theta_i\}),\end{aligned} \tag{189}$$

where we have set $\beta = 1$. In writing this expression we made use of two facts: (i) our generative distribution is of the Boltzmann form, and (ii) the partition function does not depend on the data:

$$\langle \log Z(\{\theta_i\})\rangle_{\text{data}} = \log Z(\{\theta_i\}). \tag{190}$$

15.3.2. Regularization

Just as for discriminative models like linear and logistic regression, it is common to supplement the log-likelihood with additional regularization terms (see Sections 6 and 7). Instead of minimizing the negative log-likelihood, one minimizes a cost function of the form

$$-\mathcal{L}(\{\theta_i\}) + E_{\text{reg}}(\{\theta_i\}), \tag{191}$$

where $E_{\text{reg}}(\{\theta_i\})$ is an additional regularization term that prevents overfitting. From a Bayesian perspective, this new term can be viewed as encoding a (negative) log-prior on model parameters and performing a maximum-a-posteriori (MAP) estimate instead of a MLE (see corresponding discussion in Section 6).

As we saw by studying linear regression, different forms of regularization give rise to different kinds of properties. A common choice for the regularization function are the sums of the L_1 or L_2 norms of the parameters

$$E_{\text{reg}}(\{\theta_i\}) = \Lambda \sum_i |\theta_i|^\alpha,\ \alpha = 1, 2 \tag{192}$$

with Λ controlling the regularization strength. For $\Lambda = 0$, there is no regularization and we are simply performing MLE. In contrast, a choice of large Λ will force many parameters to be close to or exactly zero. Just as in regression, an L1 penalty enforces sparsity, with many of the θ_i set to zero, and L2 regularization shrinks the size of the parameters towards zero.

One challenge of generative models is that it is often difficult to choose the regularization strength Λ. Recall that, for linear and logistic regression, Λ is chosen to maximize the out-of-sample performance on a validation dataset (i.e. cross-validation). However, for generative models our data are usually unlabeled. Therefore, choosing a regularization strength is more subtle and there exists no universal procedure for choosing Λ. One common strategy is to divide the data into a training set and a validation set and monitor a summary statistic such as the log-likelihood, energy distance (Székely, 2003), or variational free-energy of the generative model on the training and validation sets (the variational free-energy was discussed extensively in Section 14) (Hinton, 2012). If the gap between the training and validation datasets starts growing, one is probably overfitting the model even if the log-likelihood of the training dataset is still increasing. This also gives a procedure for "early stopping" – a regularization procedure we introduced in the context of discriminative models. In practice, when using such regularizers it is important to try many different values of Λ and then try to use a proxy statistic for overfitting to evaluate the optimal choice of Λ.

15.4. Computing gradients

We still need to specify a procedure for minimizing the cost function. One powerful and common choice that is widely employed when training energy-based models is stochastic gradient descent (SGD) (see Section 4). Performing MLE using SGD requires calculating the gradient of the log-likelihood Eq. (189) with respect to the parameters θ_i. To simplify notation and gain intuition, it is helpful to define "operators" $O_i(\mathbf{x})$, conjugate to the parameters θ_i

$$O_i(\mathbf{x}) = \frac{\partial E(\mathbf{x};\theta_i)}{\partial \theta_i}. \tag{193}$$

Since the partition function is just the cumulant generating function for the Boltzmann distribution, we know that the usual statistical mechanics relationships between expectation values and derivatives of the log-partition function hold:

$$\langle O_i(\mathbf{x})\rangle_{\text{model}} = \text{Tr}_{\mathbf{x}} p_\theta(\mathbf{x}) O_i(\mathbf{x}) = -\frac{\partial \log Z(\{\theta_i\})}{\partial \theta_i}. \tag{194}$$

In terms of the operators $\{O_i(\mathbf{x})\}$, the gradient of Eq. (189) takes the form (Ackley et al., 1987)

$$\begin{aligned}-\frac{\partial \mathcal{L}(\{\theta_i\})}{\partial \theta_i} &= \left\langle \frac{\partial E(\mathbf{x};\theta_i)}{\partial \theta_i}\right\rangle_{\text{data}} + \frac{\partial \log Z(\{\theta_i\})}{\partial \theta_i} \\ &= \langle O_i(\mathbf{x})\rangle_{\text{data}} - \langle O_i(\mathbf{x})\rangle_{\text{model}}.\end{aligned} \tag{195}$$

These equations have a simple and beautiful interpretation. The gradient of the log-likelihood with respect to a model parameter is a difference of moments — one calculated directly from the data and one calculated from our model using the current model parameters. The data-dependent term is known as the *positive phase* of the gradient and the model-dependent term is known as the *negative phase* of the gradient. This derivation also gives an intuitive explanation for likelihood-based training procedures. The gradient acts on the model to lower the energy of configurations that are near observed data points while raising the energy of configurations that are far from observed data points. Finally, we note that all information about the data only enters the training procedure through the expectations $\langle O_i(\mathbf{x})\rangle_{\text{data}}$ and our generative model is blind to information beyond what is contained in these expectations.

To use SGD, we must still calculate the expectation values that appear in Eq. (195). The positive phase of the gradient – the expectation values with respect to the data – can be easily calculated using samples from the training dataset. However, the negative phase – the expectation values with respect to the model – is generally much more difficult to compute. We will see that in almost all cases, we will have to resort to either numerical or approximate methods. The fundamental reason for this is that it is impossible to calculate the partition function exactly for most interesting models in both physics and ML.

There are exceptional cases in which we can calculate expectation values analytically. When this happens, the generative model is said to have a *Tractable Likelihood*. One example of a generative model with a Tractable Likelihood is the Gaussian MaxEnt model for real valued data discussed in Eq. (188). The parameters/Lagrange multipliers for this model are the local fields $\mathbf{a}$ and the pairwise coupling matrix J. In this case, the usual manipulations involving Gaussian integrals allow us to exactly find the parameters $\mu = -J^{-1}\mathbf{a}$ and $\Sigma = -J^{-1}$, yielding the familiar expressions $\boldsymbol{\mu} = \langle \mathbf{x}\rangle_{\text{data}}$ and $\Sigma = \langle(\mathbf{x} - \langle \mathbf{x}\rangle_{\text{data}})(\mathbf{x} - \langle \mathbf{x}\rangle_{\text{data}})^T\rangle_{\text{data}}$. These are the standard estimates for the sample mean and covariance matrix. Converting back to the Lagrange multipliers yields

$$J = -\langle(\mathbf{x} - \langle \mathbf{x}\rangle_{\text{data}})(\mathbf{x} - \langle \mathbf{x}\rangle_{\text{data}})^T\rangle^{-1}_{\text{data}}. \tag{196}$$

Returning to the generic case where most energy-based models have *intractable likelihoods*, we must estimate expectation values numerically. One way to do this is draw samples $\mathcal{S}_{\text{model}} = \{\mathbf{x}'_i\}$ from the model $p_\theta(\mathbf{x})$ and evaluate arbitrary expectation values using these samples:

$$\langle f(\mathbf{x})\rangle_{\text{model}} = \int \mathrm{d}\mathbf{x} p_\theta(\mathbf{x}) f(\mathbf{x}) \approx \sum_{\mathbf{x}'_i \in \mathcal{S}_{\text{model}}} f(\mathbf{x}'_i). \tag{197}$$

The samples from the model $\mathbf{x}'_i \in \mathcal{S}_{\text{model}}$ are often referred to as *fantasy particles* in the ML literature and can be generated using simple MCMC algorithms such as Metropolis–Hasting which are covered in most modern statistical physics classes. However, if the reader is unfamiliar with MCMC methods or wants a quick refresher, we recommend the concise and beautiful discussion of MCMC methods from both the physics and ML point-of-view in Chapters 29–32 of David MacKay's masterful book (MacKay, 2003).

Finally, we note that once we have the fantasy particles from the model, we can also easily calculate the gradient of any expectation value $\langle f(\mathbf{x})\rangle_{\text{model}}$ using what is commonly called the "log-derivative trick" in ML (Fu, 2006; Kleijnen and Rubinstein, 1996):

$$\begin{aligned}
\frac{\partial}{\partial \theta_i}\langle f(\mathbf{x})\rangle_{\text{model}} &= \int \mathrm{d}\mathbf{x} \frac{\partial p_\theta(\mathbf{x})}{\partial \theta_i} f(\mathbf{x}) \\
&= \left\langle \frac{\partial \log p_\theta(\mathbf{x})}{\partial \theta_i} f(\mathbf{x}) \right\rangle_{\text{model}} \\
&= \langle O_i(\mathbf{x}) f(\mathbf{x})\rangle_{\text{model}} \\
&\approx \sum_{\mathbf{x}'_j \in \mathcal{S}_{\text{model}}} O_i(\mathbf{x}_j) f(\mathbf{x}'_j).
\end{aligned} \tag{198}$$

This expression allows us to take gradients of more complex cost functions beyond the MLE procedure discussed here.

15.5. Summary of the training procedure

We now summarize the discussion above and present a general procedure for training an energy based model using SGD on the cost function (see Section 4). Our goal is to fit the parameters of a model $p_{\boldsymbol{\lambda}}(\{\theta_i\}) = Z^{-1}\mathrm{e}^{-E(\mathbf{x},\{\theta_i\})}$. Training the model involves the following steps:

1. Read a minibatch of data, $\{\mathbf{x}\}$.
2. Generate fantasy particles $\{\mathbf{x}'\} \sim p_{\boldsymbol{\lambda}}$ using an MCMC algorithm (e.g., Metropolis–Hastings).
3. Compute the gradient of log-likelihood using these samples and Eq. (195), where the averages are taken over the minibatch of data and the fantasy particles from the model, respectively.
4. Use the gradient as input to one of the gradient based optimizers discussed in section Section 4.

In practice, it is helpful to supplement this basic procedure with some tricks that help training. As with discriminative neural networks, it is important to initialize the parameters properly and print summary statistics during the training procedure on the training and validation sets to prevent overfitting. These and many other "cheap tricks" have been nicely summarized in a short note from the Hinton group (Hinton, 2012).

A major computational and practical limitation of these methods is that it is often hard to draw samples from generative models. MCMC methods often have long mixing-times (the time one has to run the Markov chain to get uncorrelated samples) and this can result in biased sampling. Luckily, we often do not need to know the gradients exactly for training ML models (recall that noisy gradient estimates often help the convergence of gradient descent algorithms), and we can significantly reduce the computational expense by running MCMC for a reasonable time window. We will exploit this observation extensively in the next section when we discuss how to train more complex energy-based models with hidden variables.

16. Deep generative models: Hidden variables and restricted Boltzmann machines (RBMs)

The last section introduced many of the core ideas behind energy-based generative models. Here, we extend this discussion to energy-based models that include latent or hidden variables.

Including latent variables in generative models greatly enhances their expressive power – allowing the model to represent sophisticated correlations between visible features without sacrificing trainability. By having multiple layers of latent variables, we can even construct powerful deep generative models that possess many of the same desirable properties as deep, discriminative neural networks.

We begin with a discussion that tries to provide a simple intuition for why latent variables are such a powerful tool for generative models. Next, we introduce a powerful class of latent variable models called Restricted Boltzmann Machines (RBMs) and discuss techniques for training these models. After that, we introduce Deep Boltzmann Machines (DBMs), which have multiple layers of latent variables. We then introduce the new Paysage package for training energy-based models and demonstrate how to use it on the MNIST dataset and samples from the Ising model. We conclude by discussing recent physics literature related to energy-based generative models.

16.1. Why hidden (latent) variables?

Latent or hidden variables are a powerful yet elegant way to encode sophisticated correlations between observable features. The underlying reason for this is that marginalizing over a subset of variables – "integrating out" degrees of freedom in the language of physics – induces complex interactions between the remaining variables. The idea that integrating out variables can lead to complex correlations is a familiar component of many physical theories. For example, when considering free electrons living on a lattice, integrating out phonons gives rise to higher-order electron–electron interactions (e.g. superconducting or magnetic correlations). More generally, in the Wilsonian renormalization group paradigm, all effective field theories can be thought of as arising from integrating out high-energy degrees of freedom (Wilson and Kogut, 1974).

Generative models with latent variables run this logic in reverse — encode complex interactions between visible variables by introducing additional, hidden variables that interact with visible degrees of freedom in a simple manner, yet still reproduce the complex correlations between visible degrees in the data once marginalized over (integrated out). This allows us to encode complex higher-order interactions between the visible variables using simpler interactions at the cost of introducing new latent variables/degrees of freedom. This trick is also widely exploited in physics (e.g. in the Hubbard–Stratonovich transformation (Hubbard, 1959; Stratonovich, 1957) or the introduction of ghost fields in gauge theory (Faddeev and Popov, 1967)).

To make these ideas more concrete, let us revisit the pairwise Ising model introduced in the discussion of MaxEnt models, see Eq. (186). The model is described by a Boltzmann distribution with energy

$$E(\mathbf{v}) = -\sum_i a_i v_i - \frac{1}{2}\sum_{ij} v_i J_{ij} v_j, \tag{199}$$

where J_{ij} is a symmetric coupling matrix that encodes the pairwise constraints and a_i enforce the single-variable constraint.

Our goal is to replace the complicated interactions between the visible variables v_i encoded by J_{ij}, by interactions with a new set of latent variables h_μ. In order to do this, it is helpful to rewrite the coupling matrix in a slightly different form. Using SVD, we can always express the coupling matrix in the form $J_{ij} = \sum_{\mu=1}^{N} W_{i\mu} W_{j\mu}$, where $\{W_{i\mu}\}_i$ are appropriately normalized singular vectors. In terms of $W_{i\mu}$, the energy takes the form

$$E_{\text{Hop}}(\mathbf{v}) = -\sum_i a_i v_i - \frac{1}{2}\sum_{ij\mu} v_i W_{i\mu} W_{j\mu} v_j. \tag{200}$$

We note that in the special case when both $v_i \in \{-1, +1\}$ and $W_{i\mu} \in \{-1, +1\}$ are binary variables, a model with this form of the energy function is known as the *Hopfield model* (Amit et al., 1985; Hopfield, 1982). The Hopfield model has played an extremely important role in statistical physics, computational neuroscience, and machine learning, and a