Bridging the Gap: Non-equilibrium Physics and Machine Learning

Ethan Buchman

December 12, 2013

1 The RBM

1.1 Description of Model

The classic physics inspired model in machine learning is the Boltzmann Machine (BM), or its restricted counterpart (RBM), which has been used to model a variety of data, both stationary and, in its extensions, temporal. The BM is a stochastic extension of a Hopfield net, the first instance of using statistical mechanics based models in machine learning [3]. An RBM is a simplified version of a BM, consisting of two layers: the visible layer (v) and the hidden layer (h), with N_v and N_h nodes, respectively. We use hidden units to capture higher level correlations in the data, and the visible units to mirror the data itself. Connections between nodes are restricted so that there are no visible-visible and hidden-hidden connections. There are only symmetric hidden-visible connections. Hence, we have a restricted BM. In physics parlance, An RBM is a bi-partite (visible and hidden) Ising magnet with full interactivity between visible and hidden units, and no interactivity between units of the same type. Its like a magnet made of spins that don't interact with each other, but do interact with the spins of another magnet. We're going to use pairs of magnets to model the world.

The RBM satisfies a Boltzmann-Gibbs distribution over all its units:

$$p(v,h) = \frac{1}{Z} e^{-E(v,h)}$$

where Z is a normalization constant (the partition function), $Z = \sum_{v,h} e^{-E(v,h)}$ and E is the energy,

$$E(v,h) = -h^T W v - b_h h - b_v v$$

where h is the vector of hidden units, v the vector of visible units, W the trainable matrix of connection weights, and b_h and b_v trainable biases on the hidden and visible units, respectively. In analogy to an Ising magnet, W is the (per pair) strength of interaction between spins, and the biases are magnetic fields acting on the spins. Our goal is to learn

a distribution over the visible units, $p(v)$ that minimizes some loss criteria with respect to the true data distribution, ρ_D.

To find $p(v)$, we marginalize over the hidden units:

$$p(v) = \frac{1}{Z} \sum_h e^{-E(v,h)}$$

which can also be written

$$p(v) = \frac{1}{Z} e^{-F(v)}$$

where

$$F(v) = -log \sum_h e^{-E(v,h)}$$

This, to a physicist, looks astoundingly like a free energy, and in fact it is. However, it is not the free energy of the full distribution, $p(v, h)$, nor of the marginal $p(v)$. Rather, it is the free energy of the conditional distribution over the hidden units, $p(h|v)$, properly defined as the difference between the average energy and the entropy of the distribution.

$$F(v) =< E(v,h) >_{p(h|v)} -H[p(h|v)]$$

$$= \sum_h p(h|v)(E(v,h) + log\ p(h|v))$$

$$= \sum_h p(h|v)(-log\ p(v,h) - log\ Z + log\ p(h|v)) = \sum_h p(h|v)(-log\ p(v) - log\ Z)$$

$$= -log\ p(v)Z = -log \sum_h e^{-E(v,h)}$$

Since it is a function of a single visible vector, and all possible hidden vectors, we call it the free energy of v. We will have occasion to return to a discussion of free energy later on. For now, it is a mathematical convenience.

To draw samples from this distribution, we need a sampling procedure. To facilitate this, we note that the structure of the RBM implies that visible units are conditionally independent of each other, given the hidden units, and hidden units are conditionally independent of each other, given the visible units. Hence, if we can derive $p(v|h)$ and $p(h|v)$, we can initialize the RBM in some random state and update the layers one at a time for as long as we like. After some initial equilibration period (which in practice can be quite long), we will be drawing samples from the RBM's equilibrium distribution, as desired. We call this procedure 'block Gibbs sampling.' We note that if we allow visible-visible and hidden-hidden connections, sampling is much more difficult and cannot be done so neatly in block steps.

Let us derive $p(v|h)$ and $p(h|v)$. Since the RBM is symmetric, we need only derive one, knowing that the same form will apply to the other. We have:

$$p(h|v) = \frac{p(v,h)}{p(v)} = \frac{e^{-E(v,h)}}{\sum_h e^{-E(v,h)}}$$

where,

$$\sum_h e^{-E(v,h)} = e^{b_v v} \sum_h e^{h^T(Wv+b_h)} = e^{b_v v} \sum_h \prod_{k=1}^{N_h} e^{h_k(Wv+b_h)_k}.$$

Now, the sum over all possible states of h factors into sums over the binary states of its individual units, which are $h_k = 0$ and $h_k = 1$. Hence the sum over products collapses into a product over sums:

$$\sum_h \prod_{k=1}^{N_h} e^{h_k(Wv+b_h)_k} = \prod_{k=1}^{N_h} (1 + e^{(Wv+b_h)_k})$$

So,

$$p(h|v) = \frac{e^{h^T(Wv+b_h)}}{\prod_{k=1}^{N_h}(1+e^{(Wv+b_h)_k})} = \prod_{k=1}^{N_h} \frac{1}{(1+e^{-}(Wv+b_h)_k)}.$$

Due to the restricted connectivity, we know the distribution factors, $p(h|v) = \prod_{k=1}^{N_h} p(h_k|v)$. Thus, we find that:

$$p(h_k|v) = \frac{1}{(1+e^{-(Wv+b_h)_k})} = \sigma((Wv+b_h)_k)$$

where $\sigma()$ is the logistic function. By symmetry, we also have

$$p(v_k|h) = \sigma((h^T W + b_v)_k)$$

From these update rules, we see that positive weights between units will encourage both units to be on together, and negative weights will encourage them to both be off. This function, ie. the sigmoid of a linear transformation, is identical to the function used in feed-forward neural networks. It permits the RBM to learn a non-linear re-presentation of the data in its hidden units, and to use those units to reconstruct the data in its visible units.

1.2 Learning in an RBM

Now, let us consider the learning. We wish to learn a distribution $p(v)$ to model the data distribution, ρ_D. To do so, we will present examples from the data distribution (the

dataset) and adjust our parameters (weights and biases) in turn. If we are successful, then upon running the RBM by alternating Gibbs sampling on the hidden and visible units, as described above, we will see that the visible units resemble the data. Intuitively, we would like the model to assign high probability to states of the visible units which resemble the data, or, equivalently, to assign low free energies to such states. One approach to achieving this is to use gradient descent on the negative logarithm of the probability the model assigns to training data, $-\log p(d)$, known as the negative log-likelihood. To do so, we need the gradient $\frac{\partial(-\log p(d))}{\partial\theta}$, where θ represents our tuneable parameters (weights and biases). Then we will make parameter updates as follows:

$$\theta := \theta - \eta\frac{\partial(-\log p(d))}{\partial\theta}$$

where η is a hyper-parameter known as the learning rate. The gradient has a particularly nice form:

$$\begin{aligned}\frac{\partial(-\log p(d))}{\partial\theta} &= -\frac{\partial}{\partial\theta}(-F(d) - \log Z) \\ &= \frac{\partial F(d)}{\partial\theta} - \sum_v p(v)\frac{\partial F(v)}{\partial\theta}\end{aligned}$$

The first term is simple to compute, and is called the"positive phase". It forces the free energy of the current data vector down, making it more favourable. The second term, however, is an expectation of the free-energy gradient over the entire model, which is intractable to compute for all but the simplest models. Hence, we require an approximation scheme. This term we call the "negative phase". It forces the free energy of states of the visible units produced by the model during sampling to be larger, causing them to be disfavoured. In effect, this learning algorithm proceeds by increasing the probability of data vectors and decreasing the probability of visible vectors the RBM dreams up on its own in equilibrium. Notice that, in either case, the free energy over the full model distribution $p(v, h)$ remains a Lyaupunov function of the dynamics, and is minimized as the RBM samples towards equilibrium. It is the free energy of a given visible vector, defined by the distribution $p(h|v)$, that is directly manipulated by the training procedure, in order to favour states resembling the data and disfavour states dreamt up by the model.

The standard technique for estimating the negative phase is called contrastive divergence (CD) [2]. The visible units are initialized with a data vector and a single Gibbs sampling step is taken, producing a new visible vector. This is considered to be a sample from the model's equilibrium distribution, and is used to compute the gradient. Of course it is not a sample from the model's equilibrium distribution, since it is only one step away from a data vector, but it works, in practice. Since we typically train in mini-batches (ie. we present multiple data instances at the same time and average the corresponding parameter updates), we are employing an average over such samples. While this is the kind of trick that would inspire a mathematician's revolt (it has been shown that contrastive

divergence corresponds to the gradient of no real function), it happens to work, and has caused a proliferation in uses and applications of the RBM. It does, however, present an incongruity with the situation of a non-equilibrium physicist.

2 The Non-Equilibrium Analogy

The essential analogy between machine learning and non-equilibrium physics is that machine learning studies bots driven by data, while non-equilibrium physics studies physical systems driven by some energetically coupled driving signal . In a typical example, a system starts in an equilibrium state, determined by an external parameter λ. The external parameter is coupled to the system's Hamiltonian, such that for each value of λ there is a corresponding equilibrium distribution. In a non-equilibrium protocol, the parameter is varied as a function of time, $\lambda(t)$. Were λ to vary infinitely slowly, the system would remain in an equilibrium state the entire time, with the particular equilibrium state determined by the current value of the parameter. However, since λ varies at a finite rate, the system is driven out of equilibrium, and carried along a non-equilibrium trajectory, $z(t)$. $z(t)$ is determined by the protocol, $\lambda(t)$, the initial condition, $z(0)$, and random fluctuations that speckle stochasticity along the non-equilibrium trajectory.

We can conceive of this non-equilibrium process occurring by a repetition of two discrete steps: first, the external parameter changes, changing the energy of the system. We call this change in energy 'work'. Next, the system responds to the change in parameter, typically by evolving toward the new equilibrium state defined by the new parameter value. The resulting change in energy is called 'heat'. The total change in energy is the sum of the work and the heat. This is of course the first law of thermodynamics, and the tendency for the system to evolve towards equilibrium is a manifestation of the second law of thermodynamics.

In contrast, an RBM running on its own has no λ. It has a single equilibrium distribution, which was learned to model some data distribution, through the course of a training procedure that involved a driving signal. However, in the CD algorithm, we introduce a discontinuity in the evolution of the RBM when we clamp the visible units to the current data vector, and hence break from the kind of process studied by a non-equilibrium physicist. When the visible units are clamped, the connection to the RBM's past is broken, and the interpretation of work is confused. What we would like, instead, is for the data to be energetically coupled to the RBM, such that each instance of the data defines a different equilibrium distribution. The data would then be a vector valued version of λ. By varying the data, we take the RBM through a series of non-equilibrium states. Then, to train for predictiveness, we may attempt to minimize some loss function between the next instantiation of the visible units and the next state of the data.

3 The FD-RBM

This is the basic idea for an extension of the RBM we call a "Field Driven"-RBM. The FD-RBM has the following energy function:

$$E(v,h) = -h^T W v - b_h h - b_v v - d^T v$$

where d is the data vector. Thus, instead of presenting data by clamping visible units, we present it in the form of a 'field' in the energy function, akin to a magnetic field driving the spins in an Ising magnet. This is very similar to the 'Conditional' RBM (CRBM) [7], but differs in the fact that the visible units are *never* clamped.

The update rules follow identically as for the RBM, but with the data serving as an additional bias on v. Hence, the equilibrium distribution is a function of the current data vector itself, or, in the terminology of the preceding section, the data vector takes on the role of λ.

To train an FD-RBM, we would like to exploit the non-equilibrium circumstance directly. We envision the FD-RBM in direct analogy to a non-equilibrium physical system. It begins in equilibrium, with some initial state of the data vector. At each time step, we present a new data vector, doing work on the system. The system then takes a single relaxation step, emitting some heat. (Both work and heat are well defined in terms of changes in the energy of the system). In so doing, we take the system through a non-equilibrium protocol equivalent to that studied in the non-equilibrium physics literature. Ideally, we would like to use results from this literature pertaining to the relationship between energy and information to train the FD-RBM.

In particular, work by Still et al. [6] deduced a non-equilibrium equivalence relation between the work dissipated by a system responding to a stochastic driving signal, and the information the system maintains about the driving signal's past state that is not useful for predicting its future state. The equivalence is given as follows:

$$< W_{diss}[d_t \rightarrow d_{t+1}] >= I_{mem}(t) - I_{pred}(t)$$

which says that the average work dissipated in a work step is equal to the difference between the mutual information of the current state of the system and the previous state of the environment (I_{mem}, or information memory) and that of the current state of the system and the next state of the environment (I_{pred}, or predictive information). For details, see [6]

To use this relation to train an FD-RBM, we could minimize the W_{diss} term, and rest assured that we are maximizing our system's information efficiency. However, we have only a guarantee that mutual information between our system and the next state of the data will be maximized, not that the system will actually sample states resembling that of the data (like in an RBM or a CRBM). Hence we would be required to figure out a decoding procedure to 'read out' the information from the FD-RBM. This, however, resembles much more closely the circumstance of the organism, which encodes in its internal

state predictions about the next state of the environment. When coupled to behavioural algorithms, this makes for highly effective inference machines.

To train on the W_{diss} term would require statistical methods capable of approximating the FD-RBM's partition function. Though in general this problem is intractable, a variety of techniques have emerged recently which make this problem more manageable. Of these, annealed importance sampling [5] has a particularly interesting relationship with non-equilibrium physics, as it follows from the same mathematics which give rise to the Jarzynski equality [4], though the relationship between the two is seldom commented upon and is little known in either community, despite the two results being of paramount importance in their respective communities. Another approach proposes to track changes to the partition function during training, keeping its estimation manageable [1]. Whether or not these methods will be suitable for training the FD-RBM remain to be seen, and the development of newer methods are much anticipated. The FD-RBM offers a unique opportunity to bridge machine learning and non-equilibrium physics in the development of algorithms which may begin to truly parallel the information capacity of organisms.

References

[1] Guillaume Desjardins, Yoshua Bengio, and Aaron C Courville. On tracking the partition function. In *Advances in Neural Information Processing Systems*, pages 2501–2509, 2011.

[2] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[3] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[4] Christopher Jarzynski. A nonequilibrium equality for free energy differences. *arXiv preprint cond-mat/9610209*, 1996.

[5] Radford M Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

[6] Susanne Still, David A Sivak, Anthony J Bell, and Gavin E Crooks. Thermodynamics of prediction. *Physical Review Letters*, 109(12):120604, 2012.

[7] Graham W Taylor and Geoffrey E Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *Proceedings of the 26th annual international conference on machine learning*, pages 1025–1032. ACM, 2009.