17 Restricted Boltzmann Machine (RBM)

Goal

Gibbs sampling, Boltzmann Machine and Restricted Boltzmann Machine.

Alert 17.1: Convention

Gray boxes are not required hence can be omitted for unenthusiastic readers.
This note is likely to be updated again soon.

Algorithm 17.2: The Metropolis-Hastings algorithm (Metropolis et al. 1953; Hastings 1970)

Suppose we want to take a sample from a density p, where direct sampling is costly. Instead, we resort to an iterative algorithm:

Algorithm: The Metropolis-Hastings Algorithm

Input: proposal (conditional) density $\mathsf{q}(\mathbf{y}|\mathbf{x})$, symmetric function $s(\mathbf{x},\mathbf{y})$, target density $\mathsf{p}(\mathbf{x})$
Output: approximate sample $\mathbf{X} \sim \mathsf{p}$

1 choose $\mathbf{X}$
2 **repeat**
3 sample $\mathbf{Y} \sim \mathsf{q}(\cdot|\mathbf{X})$
4 $\alpha(\mathbf{X},\mathbf{Y}) \leftarrow \frac{s(\mathbf{X},\mathbf{Y})}{1+\frac{\mathsf{p}(\mathbf{X})\mathsf{q}(\mathbf{Y}|\mathbf{X})}{\mathsf{p}(\mathbf{Y})\mathsf{q}(\mathbf{X}|\mathbf{Y})}} = s(\mathbf{X},\mathbf{Y})\frac{\mathsf{p}(\mathbf{Y})\mathsf{q}(\mathbf{X}|\mathbf{Y})}{\mathsf{p}(\mathbf{Y})\mathsf{q}(\mathbf{X}|\mathbf{Y})+\mathsf{p}(\mathbf{X})\mathsf{q}(\mathbf{Y}|\mathbf{X})}$
5 with probability $\alpha(\mathbf{X},\mathbf{Y})$: $\mathbf{X} \leftarrow \mathbf{Y}$
6 **until** *until convergence*

Obviously, the symmetric function s must be chosen so that $\alpha \in [0,1]$. Popular choices include:

- Metropolis-Hastings (Hastings 1970):

$$s(\mathbf{x},\mathbf{y}) = \frac{\mathsf{p}(\mathbf{x})\mathsf{q}(\mathbf{y}|\mathbf{x}) + \mathsf{p}(\mathbf{y})\mathsf{q}(\mathbf{x}|\mathbf{y})}{\mathsf{p}(\mathbf{x})\mathsf{q}(\mathbf{y}|\mathbf{x}) \vee \mathsf{p}(\mathbf{y})\mathsf{q}(\mathbf{x}|\mathbf{y})} \implies \alpha(\mathbf{x},\mathbf{y}) = 1 \wedge \frac{\mathsf{p}(\mathbf{y})\mathsf{q}(\mathbf{x}|\mathbf{y})}{\mathsf{p}(\mathbf{x})\mathsf{q}(\mathbf{y}|\mathbf{x})} \tag{17.1}$$

- Barker (Barker 1965):

$$s(\mathbf{x},\mathbf{y}) = 1 \implies \alpha(\mathbf{x},\mathbf{y}) = \frac{\mathsf{p}(\mathbf{y})\mathsf{q}(\mathbf{x}|\mathbf{y})}{\mathsf{p}(\mathbf{y})\mathsf{q}(\mathbf{x}|\mathbf{y}) + \mathsf{p}(\mathbf{x})\mathsf{q}(\mathbf{y}|\mathbf{x})}$$

The algorithm simplifies considerably if the proposal q is symmetric, i.e. $\mathsf{q}(\mathbf{x}|\mathbf{y}) = \mathsf{q}(\mathbf{y}|\mathbf{x})$, which is the original setting in (Metropolis et al. 1953):

Algorithm: The Symmetric Metropolis-Hastings Algorithm

Input: symmetric proposal density $\mathsf{q}(\mathbf{y}|\mathbf{x})$, symmetric function $s(\mathbf{x},\mathbf{y})$, target density $\mathsf{p}(\mathbf{x})$
Output: approximate sample $\mathbf{X} \sim \mathsf{p}$

1 choose $\mathbf{X}$
2 **repeat**
3 sample $\mathbf{Y} \sim \mathsf{q}(\cdot|\mathbf{X})$
4 $\alpha(\mathbf{X},\mathbf{Y}) \leftarrow s(\mathbf{X},\mathbf{Y})\frac{\mathsf{p}(\mathbf{Y})}{\mathsf{p}(\mathbf{Y})+\mathsf{p}(\mathbf{X})}$
5 with probability $\alpha(\mathbf{X},\mathbf{Y})$: $\mathbf{X} \leftarrow \mathbf{Y}$
6 **until** *until convergence*

For MH's rule (17.1), we now have

$$s(\mathbf{x},\mathbf{y}) = \frac{\mathsf{p}(\mathbf{x})}{\mathsf{p}(\mathbf{y})} \wedge \frac{\mathsf{p}(\mathbf{y})}{\mathsf{p}(\mathbf{x})} \implies \alpha(\mathbf{x},\mathbf{y}) = 1 \wedge \frac{\mathsf{p}(\mathbf{y})}{\mathsf{p}(\mathbf{x})}$$

while Barker's rule (6) reduces to

$$s(\mathbf{x},\mathbf{y})=1 \implies \alpha(\mathbf{x},\mathbf{y})=\frac{\mathsf{p}(\mathbf{y})}{\mathsf{p}(\mathbf{y})+\mathsf{p}(\mathbf{x})}.$$

In particular, if $\mathsf{p}(\mathbf{Y}) \geq \mathsf{p}(\mathbf{X})$, then MH always moves to the new position $\mathbf{Y}$ while Barker's rule may still reject and repeat over.

Hastings, W. Keith (1970). "Monte Carlo sampling methods using Markov chains and their applications". *Biometrika*, vol. 57, pp. 97–109.
Barker, A. A. (1965). "Monte Carlo calculations of the radial distribution functions for a proton-electron plasma". *Australian Journal of Physics*, vol. 18, no. 2, pp. 119–134.
Metropolis, Nicholas, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller (1953). "Equation of state calculations by fast computing machines". *Journal of Chemical Physics*, vol. 21, pp. 1087–1092.

Alert 17.3: Significance of MH

To appreciate the significance of MH, let us point out that:

- There is immense flexibility in choosing the proposal q!
- We need only know the target density p up to a constant!

Both are crucial for our application to (restricted) Boltzmann machines, as we will see.

Algorithm 17.4: Gibbs sampling (Hastings 1970; Geman and Geman 1984)

If we choose the proposal density q so that $\mathsf{q}(\mathbf{y}|\mathbf{x}) \neq 0$ only if the new position $\mathbf{y}$ and the original position $\mathbf{x}$ do not differ much (e.g. agree on all but 1 coordinate), then we obtain the so-called Gibbs sampler. Variations include:

- randomized: randomly choose a (block of) coordinate(s) j in $\mathbf{x}$ and change it (them) according to q_j.
- cyclic: loop over each (block of) coordinate(s) j in $\mathbf{x}$ and change it (them) according to q_j.

If we choose $\mathsf{q}(\mathbf{y}|\mathbf{x})=\mathsf{p}(\mathbf{y}|\mathbf{x})$, then for MH's rule $\alpha \equiv 1$ while for Barker's rule $\alpha \equiv \frac{1}{2}$.

Hastings, W. Keith (1970). "Monte Carlo sampling methods using Markov chains and their applications". *Biometrika*, vol. 57, pp. 97–109.
Geman, Stuart and Donald Geman (1984). "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, pp. 721–741.

Remark 17.5: Optimality of MH

Peskun (1973) showed that the MH rule is optimal in terms of asymptotic variance.

Peskun, P. H. (1973). "Optimum Monte-Carlo Sampling Using Markov Chains". *Biometrika*, vol. 60, no. 3, pp. 607–612.

Definition 17.6: Boltzmann distribution (e.g. Hopfield 1982)

We say a (discrete) random variable $\mathbf{S} \in \{\pm 1\}^m$ follows a Boltzmann distribution p iff there exists a symmetric matrix $W \in \mathbb{S}^{m+1}$ such that

$$\forall \mathbf{s} \in \{\pm 1\}^m, \quad \mathsf{p}_W(\mathbf{S}=\mathbf{s})=\exp(\mathbf{s}^\top W \mathbf{s}-A(W)), \quad \text{where} \quad A(W)=\log \sum_{\mathbf{s}\in\{\pm 1\}^m} \exp(\mathbf{s}^\top W \mathbf{s}) \tag{17.2}$$

is the log-partition function. It is clear that Boltzmann distributions belong to the exponential family Defini-

tion 16.4, with sufficient statistics

$$T(\mathbf{s}) = \mathbf{s}\mathbf{s}^\top.$$

We remind that we have appended the constant 1 in $\mathbf{s}$ so that W contains the bias term too.

Hopfield, John J. (1982). "Neural networks and physical systems with emergent collective computational abilities". *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558.

Alert 17.7: Coding convention

We used the encoding $\{\pm 1\}$ to represent a binary value above. As a consequence, the diagonal entries in W only contribute a constant (independent of the realization $\mathbf{s}$) in (17.2). Thus, w.l.o.g. we may absorb $\operatorname{diag}(W)$ into $A(W)$ so that $\operatorname{diag}(W) = \mathbf{0}$.

On the other hand, if we use the encoding $\{0, 1\}$, while conceptually being equivalent, we will no longer need to perform padding, since the bias term can now be stored in the diagonal of W.

Alert 17.8: Intractability of Boltzmann distributions

Despite the innocent form (17.2), Boltzmann distributions are in general intractable (for large m), since the log-partition function involves summation over 2^m terms (Long and Servedio 2010). This is common in Bayesian analysis where we know a distribution only up to an intractable normalization constant.

Long, Philip M. and Rocco A. Servedio (2010). "Restricted Boltzmann Machines Are Hard to Approximately Evaluate or Simulate". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 703–710.

Example 17.9: m=1 reduces to (binary) logistic

For $m = 1$ we have

$$\mathsf{p}_W(S = 1) = \frac{\exp(2w_{12})}{\exp(2w_{12}) + \exp(-2w_{12})} = \mathtt{sgm}(w), \quad \text{where} \quad w := 4w_{12}$$

and recall the sigmoid function $\mathtt{sgm}(t) = \frac{1}{1+\exp(-t)}$.

This example confirms that even if we can choose any W, the resulting set of Boltzmann distributions forms a strict subset of all discrete distributions over the cube $\{\pm 1\}^m$.

Definition 17.10: Boltzmann machine (BM) (Ackley et al. 1985; Hinton and Sejnowski 1986)

Now let us partition the Boltzmann random variable $\mathbf{S}$ into the concatenation of an observed random variable $\mathbf{X} \in \{\pm 1\}^d$ and a latent random variable $\mathbf{Z} \in \{\pm 1\}^t$. We call the marginal distribution over $\mathbf{X}$ a Boltzmann machine. Note that $\mathbf{X}$ no longer belongs to the exponential family!

Given a sample $\mathbf{X}_1, \ldots, \mathbf{X}_n$, we are interested in learning the Boltzmann machine, namely the marginal density $\mathsf{p}_W(\mathbf{x})$. We achieve this goal by learning the symmetric matrix W that defines the joint Boltzmann distribution $\mathsf{p}_W(\mathbf{s}) = \mathsf{p}_W(\mathbf{x}, \mathbf{z})$ in (17.2).

Ackley, David H., Geoffrey E. Hinton, and Terrence J. Sejnowski (1985). "A learning algorithm for boltzmann machines". *Cognitive Science*, vol. 9, no. 1, pp. 147–169.

Hinton, Geoffrey E. and T. J. Sejnowski (1986). "Learning and Relearning in Boltzmann Machines". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*. Ed. by David E. Rumelhart, James L. McClelland, and the PDP Research Group. The MIT Press, pp. 282–317.

Definition 17.11: Restricted Boltzmann Machine (RBM) (Smolensky 1986)

Let us consider the partition of the symmetric matrix

$$W = \begin{bmatrix} W_{xx} & W_{xz} \\ W_{xz}^\top & W_{zz} \end{bmatrix}.$$

If we require $W_{xx} = \mathbf{0}$ and $W_{zz} = \mathbf{0}$, then we obtain the restricted Boltzmann machine:

$$\mathsf{p}_W(\mathbf{x}, \mathbf{z}) \propto \exp\left(\mathbf{x}^\top W_{xz} \mathbf{z}\right), \tag{17.3}$$

i.e., only cross products are allowed.

Similarly, we will consider learning RBM through estimating the (rectangular) matrix $W_{xz} \in \mathbb{R}^{(d+1)\times(t+1)}$.

Smolensky, Paul (1986). "Information Processing in Dynamical Systems: Foundations of Harmony Theory". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*. The MIT Press, pp. 194–281.

Example 17.12: $m = 1, t = 1$

Let $m = 1$ and $t = 1$. We have for (R)BM:

$$\begin{aligned} \mathsf{p}(X = x, Z = z) &\propto \exp\left(2xzw_{12} + 2xw_{13} + 2zw_{23}\right) \\ \mathsf{p}(X = 1) &\propto \exp\left(2w_{12} + 2w_{13} + 2w_{23}\right) + \exp\left(-2w_{12} + 2w_{13} - 2w_{23}\right), \end{aligned}$$

In general, RBM is a strict subset of BM.

Remark 17.13: Representation power of (R)BM—the power of latent variables

Freund and Haussler (1992) and Neal (1992) are among the first to prove that RBM and BM can approximate any discrete distribution on $\{\pm 1\}^d$ arbitrarily well if the number t of latent variables is large (approaching 2^d). More refined results appeared later in (Le Roux and Bengio 2008; Le Roux and Bengio 2010; Montúfar and Ay 2011; Montúfar 2014).

In essence, when we marginalize out the latent variables in a (restricted) Boltzmann distribution, we create a mixture of many components on the remaining variables, hence the ability to approximate any discrete distribution.

Freund, Yoav and David Haussler (1992). "Unsupervised learning of distributions on binary vectors using two layer networks". In: *Advances in Neural Information Processing Systems 4*. Ed. by J. E. Moody, S. J. Hanson, and R. P. Lippmann, pp. 912–919.

Neal, Radford M. (1992). "Connectionist learning of belief networks". *Artificial Intelligence*, vol. 56, no. 1, pp. 71–113.

Le Roux, Nicolas and Yoshua Bengio (2008). "Representational Power of Restricted Boltzmann Machines and Deep Belief Networks". *Neural Computation*, vol. 20, no. 6, pp. 1631–1649.

— (2010). "Deep Belief Networks Are Compact Universal Approximators". *Neural Computation*, vol. 22, no. 8, pp. 2192–2207.

Montúfar, Guido and Nihat Ay (2011). "Refinements of Universal Approximation Results for Deep Belief Networks and Restricted Boltzmann Machines". *Neural Computation*, vol. 23, no. 5, pp. 1306–1319.

Montúfar, Guido F. (2014). "Universal Approximation Depth and Errors of Narrow Belief Networks with Discrete Units". *Neural Computation*, vol. 26, no. 7, pp. 1386–1407.

Remark 17.14: Computing the gradient

We now apply the same maximum likelihood idea in Algorithm 16.12 for learning an (R)BM:

$$\min_W \ \mathsf{KL}(\hat{\chi}(\mathbf{x}) \| \mathsf{p}_W(\mathbf{x})) \quad \equiv \quad \min_W \ -\mathsf{E}_{\hat{\chi}} \log \sum_{\mathbf{z} \in \{\pm 1\}^t} \mathsf{p}_W(\mathbf{X}, \mathbf{z}).$$

Taking derivative w.r.t. W we obtain:

$$\begin{aligned}\frac{\partial}{\partial W} &= -\mathsf{E}_{\hat{\chi}} \frac{\sum_{\mathbf{z}} \frac{\partial \mathsf{p}_W(\mathbf{X},\mathbf{z})}{\partial W}}{\sum_{\mathbf{z}} \mathsf{p}_W(X,\mathbf{z})} \\ &= -\mathsf{E}_{\hat{\chi}} \sum_{\mathbf{z}} \mathsf{p}_W(\mathbf{z}|\mathbf{X}) \frac{\partial \log \mathsf{p}_W(\mathbf{X},\mathbf{z})}{\partial W} \\ &= -\mathsf{E}_{\hat{\chi}} \sum_{\mathbf{z}} \mathsf{p}_W(\mathbf{z}|\mathbf{X}) \frac{\partial (\mathbf{s}^\top W \mathbf{s} - A(W))}{\partial W} \\ &= -\mathsf{E}_{\hat{\chi}} \sum_{\mathbf{z}} \mathsf{p}_W(\mathbf{z}|\mathbf{X}) \mathbf{s}\mathbf{s}^\top + \nabla A(W).\end{aligned}$$

Denoting $\hat{\mathsf{p}}_W(\mathbf{x},\mathbf{z}) = \hat{\chi}(\mathrm{d}\mathbf{x})\mathsf{p}_W(\mathbf{z}|\mathbf{x})$ and applying Exercise 16.7 we obtain the beautiful formula:

$$\frac{\partial}{\partial W} = -\mathsf{E}_{\hat{\mathsf{p}}_W} \mathbf{s}\mathbf{s}^\top + \mathsf{E}_{\mathsf{p}_W} \mathbf{s}\mathbf{s}^\top, \quad \text{where} \quad \mathbf{s} = (\mathbf{x};\mathbf{z};1).$$

The same result, with the restriction that $W_{xx} = \mathbf{0}$ and $W_{xz} = \mathbf{0}$, holds for RBM.

Therefore, we may apply (stochastic) gradient descent to find W, provided that we can evaluate the two expectations above. This is where we need the Gibbs sampling algorithm in Algorithm 17.4.

Alert 17.15: Failure of EM

Let us attempt to apply the EM Algorithm 16.3 for estimating W:

- E-step: $\mathcal{E}_{t+1}(\mathbf{z}|\mathbf{x}) = \mathsf{p}_{W_t}(\mathbf{z}|\mathbf{x})$.
- M-step: $W_{t+1} = \operatorname{argmin}_W \ \mathsf{KL}(\hat{\mathsf{p}}_{t+1} \| \mathsf{p}_W)$, where recall that $\hat{\mathsf{p}}_{t+1}(\mathbf{x},\mathbf{z}) := \hat{\chi}(\mathrm{d}\mathbf{x}) \cdot \mathsf{p}_{W_t}(\mathbf{z}|\mathbf{x})$.

It follows then from Exercise 16.9 (or more directly from Exercise 16.10) that

$$W_{t+1} = \nabla A^{-1}(\mathsf{E}_{\hat{\mathsf{p}}_{t+1}} T(\mathbf{X})), \quad i.e. \ \ \mathsf{E}_{\mathsf{p}_{t+1}} T(\mathbf{X}) = \mathsf{E}_{\hat{\mathsf{p}}_{t+1}} T(\mathbf{X}), \quad \text{where} \quad \mathsf{p}_{t+1} := \mathsf{p}_{W_{t+1}}. \tag{17.4}$$

However, we cannot implement (17.4) since the log-partition function A hence also its gradient ∇A is not tractable!

In fact, the gradient algorithm in Remark 17.14 is some explicit form that bypasses the difficulty in EM. Namely, to solve a nonlinear equation

$$\mathbf{f}(W) = 0, \qquad [\text{ for our case, } \mathbf{f}(W) = \mathsf{E}_{\mathsf{p}_W} T(\mathbf{X}) - \mathsf{E}_{\hat{\mathsf{p}}_{t+1}} T(\mathbf{X})]$$

we apply the fixed-point iteration:

$$W \leftarrow W - \eta \cdot \mathbf{f}(W),$$

which converges (if at all) iff $\mathbf{f}(W) = 0$.

Algorithm 17.16: Learning BM

To estimate $\mathsf{E}_{\mathsf{p}_W} \mathbf{s}\mathbf{s}^\top$, we need to be able to draw a sample $\mathbf{S} \sim \mathsf{p}_W$. This can be achieved by the Gibbs sampling Algorithm 17.4. Indeed, we know (recall that conditional of exponential family is again in exponential family, Exercise 16.8)

$$\mathsf{p}_W(S_j = s_j | \mathbf{S}_{\setminus j} = \mathbf{s}_{\setminus j}) \propto \exp\left(2 s_j \left\langle W_{\setminus j,j}, \mathbf{s}_{\setminus j} \right\rangle\right), \quad i.e. \ \ \mathsf{p}_W(S_j = s_j | \mathbf{S}_{\setminus j} = \mathbf{s}_{\setminus j}) = \mathtt{sgm}(4 s_j \left\langle W_{\setminus j,j}, \mathbf{s}_{\setminus j} \right\rangle), \tag{17.5}$$

where the subscript $_{\setminus j}$ indicates removal of the j-th entry or row.

Algorithm: Gibbs sampling from Boltzmann distribution p_W

Input: symmetric matrix $W \in \mathbb{S}^{m+1}$
Output: approximate sample $\mathbf{s} \sim \mathsf{p}_W$
initialize $\mathbf{s} \in \{\pm 1\}^m$
repeat
  for $j = 1, \dots, m$ **do**
    $p_j \leftarrow \mathtt{sgm}(4 \langle W_{\setminus j,j}, \mathbf{s}_{\setminus j} \rangle)$
    with probability p_j set $s_j = 1$, otherwise set $s_j = -1$
until *until convergence*

Similarly, to estimate $\mathsf{E}_{\hat{\mathsf{p}}_W} \mathbf{s}\mathbf{s}^\top$ we first draw a training sample $\mathbf{x} \sim \hat{\chi}$, and then draw $\mathbf{z} \sim \mathsf{p}_W(\cdot|\mathbf{x})$. For the latter, we fix $\mathbf{x}$ and apply the Gibbs sampling algorithm:

Algorithm: Gibbs sampling from conditional Boltzmann distribution $\mathsf{p}_W(\cdot|\mathbf{x})$

Input: symmetric matrix $W \in \mathbb{S}^{m+1}$, training sample $\mathbf{x}$
Output: approximate sample $\mathbf{z} \sim \mathsf{p}_W(\cdot|\mathbf{x})$
initialize $\mathbf{z} \in \{\pm 1\}^t$ and set $\mathbf{s} = (\mathbf{x}; \mathbf{z}; 1)$
repeat
  for $j = d+1, \dots, m$ **do**
    $p_j \leftarrow \mathtt{sgm}(4 \langle W_{\setminus j,j}, \mathbf{s}_{\setminus j} \rangle)$
    with probability p_j set $s_j = 1$, otherwise set $s_j = -1$
until *until convergence*

The above algorithms are inherently sequential hence extremely slow.

Algorithm 17.17: Learning RBM

We can now appreciate a big advantage in RBM:

$$\mathsf{p}_W(Z_j = z_j | \mathbf{Z}_{\setminus j} = \mathbf{z}_{\setminus j}, \mathbf{X} = \mathbf{x}) \propto \exp\left(z_j \langle W_{x,j}, \mathbf{x} \rangle\right), \quad i.e. \quad \mathsf{p}_W(Z_j = z_j | \mathbf{Z}_{\setminus j} = \mathbf{z}_{\setminus j}, \mathbf{X} = \mathbf{x}) = \mathtt{sgm}(2 z_j \langle W_{x,j}, \mathbf{x} \rangle),$$

and similarly

$$\mathsf{p}_W(X_j = x_j | \mathbf{X}_{\setminus j} = \mathbf{x}_{\setminus j}, \mathbf{Z} = \mathbf{z}) = \mathtt{sgm}(2 x_j \langle W_{j,z}, \mathbf{z} \rangle).$$

This is possible since in RBM (17.3), $\mathbf{X}$ only interacts with $\mathbf{Z}$ but there is no interaction within either $\mathbf{X}$ or $\mathbf{Z}$. Thus, we may apply block Gibbs sampling:

Algorithm: Block Gibbs sampling from RBM p_W

Input: rectangular matrix $W \in \mathbb{R}^{(d+1)\times(t+1)}$
Output: approximate sample $\mathbf{s} \sim \mathsf{p}_W$
initialize $\mathbf{s} = (\mathbf{x}, \mathbf{z}) \in \{\pm 1\}^{d+t}$
repeat
  $\mathbf{p} \leftarrow \mathtt{sgm}(2W\mathbf{z})$
  for $j = 1, \dots, d$, *in parallel* **do**
    with probability p_j set $x_j = 1$, otherwise set $x_j = -1$
  $\mathbf{q} \leftarrow \mathtt{sgm}(2W^\top \mathbf{x})$
  for $j = 1, \dots, t$, *in parallel* **do**
    with probability q_j set $z_j = 1$, otherwise set $z_j = -1$
until *until convergence*

Similarly, to estimate $\mathsf{E}_{\hat{\mathsf{p}}_W} \mathbf{s}\mathbf{s}^\top$ we first draw a training sample $\mathbf{x} \sim \hat{\chi}$, and then draw $\mathbf{z} \sim \mathsf{p}_W(\cdot|\mathbf{x})$. For the

latter, we fix $\mathbf{x}$ and apply the Block Gibbs sampling algorithm:

Algorithm: Block Gibbs sampling from conditional RBM $\mathsf{p}_W(\cdot|\mathbf{x})$

Input: rectangular matrix $W \in \mathbb{R}^{(d+1)\times(t+1)}$
Output: approximate sample $\mathbf{z} \sim \mathsf{p}_W(\cdot|\mathbf{x})$
initialize $\mathbf{z} \in \{\pm 1\}^t$
repeat
  $\mathbf{q} \leftarrow \mathtt{sgm}(2W^\top \mathbf{x})$
  for $j = 1, \ldots, t$, *in parallel* **do**
    with probability q_j set $z_j = 1$, otherwise set $z_j = -1$
until *until convergence*

Remark 17.18: Sampling and marginalization

After we have learned the parameter matrix W, we can draw a new sample $(\mathbf{X}, \mathbf{Z})$ from $\mathsf{p}_W(\mathbf{x}, \mathbf{z})$ using the same unconditioned (block) Gibbs sampling algorithm. Simply dropping $\mathbf{Z}$ we obtain a sample $\mathbf{X}$ from the marginal distribution $\mathsf{p}_W(\mathbf{x})$.

For RBM, we can actually derive the marginal density (up to a constant):

$$\begin{aligned} \mathsf{p}_W(\mathbf{X} = \mathbf{x}) &\propto \sum_{\mathbf{z}\in\{\pm1\}^t} \exp\left(\mathbf{x}^\top W \mathbf{z}\right) \\ &= \sum_{\mathbf{z}\in\{\pm1\}^t} \prod_{j=1}^{t+1} \exp(\mathbf{x}^\top W_{:j} z_j) \\ &= \exp(\mathbf{x}^\top W_{:,t+1}) \prod_{j=1}^{t} \left[\exp(\langle \mathbf{x}, W_{:j}\rangle) + \exp(-\langle \mathbf{x}, W_{:j}\rangle)\right]. \end{aligned} \tag{17.6}$$

A similar formula for $\mathsf{p}_W(\mathbf{Z} = \mathbf{z})$ obviously holds as well. Thus, for RBM, if we need only draw a sample $\mathbf{X}$, we can and perhaps should *directly* apply Gibbs sampling to the marginal density (17.6).

Exercise 17.19: Conditional independence in RBM

Prove or disprove the following for BM and RBM:

$$\mathsf{p}_W(\mathbf{z}|\mathbf{x}) = \prod_{j=1}^{t} \mathsf{p}_W(z_j|\mathbf{x}), \quad \mathsf{p}_W(\mathbf{x}|\mathbf{z}) = \prod_{j=1}^{d} \mathsf{p}_W(x_j|\mathbf{z}).$$

Remark 17.20: RBM as stochastic/feed-forward neural network

RBM is often referred to as a two-layer stochastic neural network, where $\mathbf{X}$ is the input layer while $\mathbf{Z}$ is the output layer. It also defines an underlying nonlinear, deterministic, feed-forward network. Indeed, let

$$y_j = \mathsf{p}_W(Z_j = 1|\mathbf{X} = \mathbf{x}) = \mathtt{sgm}(2\langle W_{:j}, \mathbf{x}\rangle)$$

we obtain a nonlinear feedforward network that takes an input $\mathbf{x} \in \{\pm 1\}^d$ and maps it non-linearly to an output $\mathbf{y} \in [0,1]^t$, through:

$$\begin{aligned} \mathbf{h} &= 2W^\top \mathbf{x} \\ \mathbf{y} &= \mathtt{sgm}(\mathbf{h}). \end{aligned}$$

Of course, we can stack RBMs on top of each other and go "deep" (Hinton and Salakhutdinov 2006; Salakhutdinov and Hinton 2012). Applications can be found in (Mohamed et al. 2012; Sarikaya et al. 2014).

Hinton, G. E. and R. R. Salakhutdinov (2006). "Reducing the Dimensionality of Data with Neural Networks". *Science*, vol. 313, pp. 504–507.
Salakhutdinov, Ruslan and Geoffrey Hinton (2012). "An Efficient Learning Procedure for Deep Boltzmann Machines". *Neural Computation*, vol. 24, no. 8, pp. 1967–2006.
Mohamed, A., G. E. Dahl, and G. Hinton (2012). "Acoustic Modeling Using Deep Belief Networks". *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22.
Sarikaya, R., G. E. Hinton, and A. Deoras (2014). "Application of Deep Belief Networks for Natural Language Understanding". *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 4, pp. 778–784.

Remark 17.21: Not necessarily binary data

It is possible to extend (R)BM to handle other types of data, see for instance (Welling et al. 2005).

In fact, let $\mathsf{p}_W(\mathbf{x}, \mathbf{z}) = \exp(\langle T(\mathbf{x}, \mathbf{z}), W\rangle - A(W))$ be any joint density in the exponential family. Then, we can estimate the parameter matrix W as before:

$$\min_W \quad \mathsf{KL}(\hat{\chi}(\mathbf{x}) \| \mathsf{p}_W(\mathbf{x})) \qquad \equiv \qquad \min_W \quad -\mathsf{E}_{\hat{\chi}} \log \int_{\mathbf{z}} \mathsf{p}_W(\mathbf{X}, \mathbf{z})\, \mathrm{d}\mathbf{z}.$$

Denoting $\hat{\mathsf{p}}_W(\mathbf{x}, \mathbf{z}) = \hat{\chi}(\mathrm{d}\mathbf{x})\mathsf{p}_W(\mathbf{z}|\mathbf{x})$ and following the same steps as in Remark 17.14 we obtain:

$$\frac{\partial}{\partial W} = -\mathsf{E}_{\hat{\mathsf{p}}_W} T(\mathbf{X}, \mathbf{Z}) + \mathsf{E}_{\mathsf{p}_W} T(\mathbf{X}, \mathbf{Z}).$$

A *restricted* counterpart corresponds to

$$\langle T(\mathbf{x}, \mathbf{z}), W\rangle = T_1(\mathbf{x})^\top W_{xz} T_2(\mathbf{z}).$$

Similar Gibbs sampling algorithms can be derived to approximate the expectations.

Welling, Max, Michal Rosen-zvi, and Geoffrey E. Hinton (2005). "Exponential Family Harmoniums with an Application to Information Retrieval". In: *Advances in Neural Information Processing Systems 17*, pp. 1481–1488.