March 27-31: Advanced machine learning and data analysis for the physical sciences

Morten Hjorth-Jensen[1,2]

[1]Department of Physics and Center for Computing in Science Education, University of Oslo, Norway
[2]Department of Physics and Astronomy and Facility for Rare Isotope Beams, Michigan State University, East Lansing, Michigan, USA

March 27-31, 2023

Plans for the week March 27-31

Deep generative models.

1. Monte Carlo methods and structured probabilistic models for deep learning
2. Partition function and Boltzmann machines
3. Reading recommendation: Goodfellow et al chapters 16-18
4. Video of lecture
5. Whiteboard notes

Probabilistic models and Bayesian statistics

See also whiteboard notes March 29

A central theorem in statistics is Bayes' theorem. This theorem plays a similar role as the good old Pythagoras' theorem in geometry. Bayes' theorem is extremely simple to derive. But to do so we need some basic axioms from statistics.

Assume we have two domains of events $X = [x_0, x_1, \dots, x_{n-1}]$ and $Y = [y_0, y_1, \dots, y_{n-1}]$.

We define also the likelihood for X and Y as $p(X)$ and $p(Y)$ respectively. The likelihood of a specific event x_i (or y_i) is then written as $p(X = x_i)$ or just $p(x_i) = p_i$.

Union of events is given by.

$$p(X \cup Y) = p(X) + p(Y) - p(X \cap Y).$$

The product rule (aka joint probability) is given by.

$$p(X \cup Y) = p(X, Y) = p(X|Y)p(Y) = p(Y|X)p(X),$$

where we read $p(X|Y)$ as the likelihood of obtaining X given Y.

If we have independent events then $p(X, Y) = p(X)p(Y)$.

Marginal Probability

The marginal probability is defined in terms of only one of the set of variables X, Y. For a discrete probability we have

$$p(X) = \sum_{i=0}^{n-1} p(X, Y = y_i) = \sum_{i=0}^{n-1} p(X|Y = y_i)p(Y = y_i) = \sum_{i=0}^{n-1} p(X|y_i)p(y_i).$$

Conditional Probability

The conditional probability, if $p(Y) > 0$, is

$$p(X|Y) = \frac{p(X, Y)}{p(Y)} = \frac{p(X, Y)}{\sum_{i=0}^{n-1} p(Y|X = x_i)p(x_i)}.$$

Bayes' Theorem

If we combine the conditional probability with the marginal probability and the standard product rule, we have

$$p(X|Y) = \frac{p(X, Y)}{p(Y)},$$

which we can rewrite as

$$p(X|Y) = \frac{p(X, Y)}{\sum_{i=0}^{n-1} p(Y|X = x_i)p(x_i)} = \frac{p(Y|X)p(X)}{\sum_{i=0}^{n-1} p(Y|X = x_i)p(x_i)},$$

which is Bayes' theorem. It allows us to evaluate the uncertainty in in X after we have observed Y. We can easily interchange X with Y.

Interpretations of Bayes' Theorem

The quantity $p(Y|X)$ on the right-hand side of the theorem is evaluated for the observed data Y and can be viewed as a function of the parameter space represented by X. This function is not necesseraly normalized and is normally called the likelihood function.

The function $p(X)$ on the right hand side is called the prior while the function on the left hand side is the called the posterior probability. The denominator on the right hand side serves as a normalization factor for the posterior distribution.

Let us try to illustrate Bayes' theorem through an example.

Example of Usage of Bayes' theorem

Let us suppose that you are undergoing a series of mammography scans in order to rule out possible breast cancer cases. We define the sensitivity for a positive event by the variable X. It takes binary values with $X = 1$ representing a positive event and $X = 0$ being a negative event. We reserve Y as a classification parameter for either a negative or a positive breast cancer confirmation. (Short note on wordings: positive here means having breast cancer, although none of us would consider this being a positive thing).

We let $Y = 1$ represent the the case of having breast cancer and $Y = 0$ as not.

Let us assume that if you have breast cancer, the test will be positive with a probability of 0.8, that is we have

$$p(X = 1|Y = 1) = 0.8.$$

This obviously sounds scary since many would conclude that if the test is positive, there is a likelihood of 80% for having cancer. It is however not correct, as the following Bayesian analysis shows.

Doing it correctly

If we look at various national surveys on breast cancer, the general likelihood of developing breast cancer is a very small number. Let us assume that the prior probability in the population as a whole is

$$p(Y = 1) = 0.004.$$

We need also to account for the fact that the test may produce a false positive result (false alarm). Let us here assume that we have

$$p(X = 1|Y = 0) = 0.1.$$

Using Bayes' theorem we can then find the posterior probability that the person has breast cancer in case of a positive test, that is we can compute

$$p(Y=1|X=1) = \frac{p(X=1|Y=1)p(Y=1)}{p(X=1|Y=1)p(Y=1)+p(X=1|Y=0)p(Y=0)} = \frac{0.8\times 0.004}{0.8\times 0.004+0.1\times 0.996} = 0.031.$$

That is, in case of a positive test, there is only a 3% chance of having breast cancer!

Why Markov chains, Brownian motion and the Metropolis algorithm

- We want to study a physical system which evolves towards equilibrium, from given initial conditions.
- We start with a PDF $w(x_0, t_0)$ and we want to understand how the system evolves with time.
- We want to reach a situation where after a given number of time steps we obtain a steady state. This means that the system reaches its most likely state (equilibrium situation)
- Our PDF is normally a multidimensional object whose normalization constant is impossible to find.
- Analytical calculations from $w(x, t)$ are not possible.
- To sample directly from from $w(x, t)$ is not possible/difficult.
- The transition probability W is also not known.
- How can we establish that we have reached a steady state? Sounds impossible!

Use Markov chain Monte Carlo

Brownian motion and Markov processes

A Markov process is a random walk with a selected probability for making a move. The new move is independent of the previous history of the system.

The Markov process is used repeatedly in Monte Carlo simulations in order to generate new random states.

The reason for choosing a Markov process is that when it is run for a long enough time starting with a random state, we will eventually reach the most likely state of the system.

In thermodynamics, this means that after a certain number of Markov processes we reach an equilibrium distribution.

This mimicks the way a real system reaches its most likely state at a given temperature of the surroundings.

To reach this distribution, the Markov process needs to obey two important conditions, that of **ergodicity** and **detailed balance**. These conditions impose then constraints on our algorithms for accepting or rejecting new random states.

The Metropolis algorithm discussed here abides to both these constraints.

The Metropolis algorithm is widely used in Monte Carlo simulations and the understanding of it rests within the interpretation of random walks and Markov processes.

In a random walk one defines a mathematical entity called a **walker**, whose attributes completely define the state of the system in question.

The state of the system can refer to any physical quantities, from the vibrational state of a molecule specified by a set of quantum numbers, to the brands of coffee in your favourite supermarket.

The walker moves in an appropriate state space by a combination of deterministic and random displacements from its previous position.

This sequence of steps forms a **chain**.

- We want to study a physical system which evolves towards equilibrium, from given initial conditions.
- Markov chains are intimately linked with the physical process of diffusion.
- From a Markov chain we can then derive the conditions for detailed balance and ergodicity. These are the conditions needed for obtaining a steady state.
- The widely used algorithm for doing this is the so-called Metropolis algorithm, in its refined form the Metropolis-Hastings algorithm.

Applications: almost every field in science

- Financial engineering, see for example Patriarca *et al*, Physica **340**, page 334 (2004).
- Neuroscience, see for example Lipinski, Physics Medical Biology **35**, page 441 (1990) or Farnell and Gibson, Journal of Computational Physics **208**, page 253 (2005)
- Tons of applications in physics
- and chemistry
- and biology, medicine
- Nobel prize in economy to Black and Scholes

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0.$$

The Black and Scholes equation is a partial differential equation, which describes the price of the option over time. It is a diffusion equation with a random term.

The list of applications is endless

The obvious case is that of a random walker on a one-, or two- or three-dimensional lattice (dubbed coordinate space hereafter).

Consider a system whose energy is defined by the orientation of single spins. Consider the state i, with given energy E_i represented by the following N spins

$$\begin{array}{cccccccccc} \uparrow & \uparrow & \uparrow & \dots & \uparrow & \downarrow & \uparrow & \dots & \uparrow & \downarrow \\ 1 & 2 & 3 & \dots & k-1 & k & k+1 & \dots & N-1 & N \end{array}$$

We may be interested in the transition with one single spinflip to a new state j with energy E_j

$$\begin{array}{cccccccccc} \uparrow & \uparrow & \uparrow & \dots & \uparrow & \uparrow & \uparrow & \dots & \uparrow & \downarrow \\ 1 & 2 & 3 & \dots & k-1 & k & k+1 & \dots & N-1 & N \end{array}$$

This change from one microstate i (or spin configuration) to another microstate j is the **configuration space** analogue to a random walk on a lattice. Instead of jumping from one place to another in space, we 'jump' from one microstate to another.

Markov processes

A Markov process allows in principle for a microscopic description of Brownian motion. As with the random walk studied in the previous section, we consider a particle which moves along the x-axis in the form of a series of jumps with step length $\Delta x = l$. Time and space are discretized and the subsequent moves are statistically independent, i.e., the new move depends only on the previous step and not on the results from earlier trials. We start at a position $x = jl = j\Delta x$ and move to a new position $x = i\Delta x$ during a step $\Delta t = \epsilon$, where $i \geq 0$ and $j \geq 0$ are integers. The original probability distribution function (PDF) of the particles is given by $w_i(t = 0)$ where i refers to a specific position on the grid in

The function $w_i(t = 0)$ is now the discretized version of $w(x, t)$. We can regard the discretized PDF as a vector.

For the Markov process we have a transition probability from a position $x = jl$ to a position $x = il$ given by

$$W_{ij}(\epsilon) = W(il - jl, \epsilon) = \begin{cases} \frac{1}{2} & |i - j| = 1 \\ 0 & \text{else} \end{cases},$$

where W_{ij} is normally called the transition probability and we can represent it, see below, as a matrix. **Here we have specialized to a case where the transition probability is known**.

Our new PDF $w_i(t = \epsilon)$ is now related to the PDF at $t = 0$ through the relation

$$w_i(t = \epsilon) = \sum_j W(j \to i) w_j(t = 0).$$

This equation represents the discretized time-development of an original PDF with equal probability of jumping left or right.

Since both W and w represent probabilities, they have to be normalized, i.e., we require that at each time step we have

$$\sum_i w_i(t) = 1,$$

and

$$\sum_j W(j \to i) = 1,$$

which applies for all j-values. The further constraints are $0 \leq W_{ij} \leq 1$ and $0 \leq w_j \leq 1$. Note that the probability for remaining at the same place is in general not necessarily equal zero.

The time development of our initial PDF can now be represented through the action of the transition probability matrix applied n times. At a time $t_n = n\epsilon$ our initial distribution has developed into

$$w_i(t_n) = \sum_j W_{ij}(t_n) w_j(0),$$

and defining

$$W(il - jl, n\epsilon) = (W^n(\epsilon))_{ij}$$

we obtain

$$w_i(n\epsilon) = \sum_j (W^n(\epsilon))_{ij} w_j(0),$$

or in matrix form

$$\hat{w}(n\epsilon) = \hat{W}^n(\epsilon)\hat{w}(0). \tag{1}$$

An Illustrative Example

The following simple example may help in understanding the meaning of the transition matrix $\hat{W}$ and the vector $\hat{w}$. Consider the 4×4 matrix $\hat{W}$

$$\hat{W} = \begin{pmatrix} 1/4 & 1/9 & 3/8 & 1/3 \\ 2/4 & 2/9 & 0 & 1/3 \\ 0 & 1/9 & 3/8 & 0 \\ 1/4 & 5/9 & 2/8 & 1/3 \end{pmatrix},$$

and we choose our initial state as

$$\hat{w}(t = 0) = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

We note that both the vector and the matrix are properly normalized. Summing the vector elements gives one and summing over columns for the matrix results also in one. Furthermore, the largest eigenvalue is one. We act then on $\hat{w}$ with $\hat{W}$. The first iteration is

$$\hat{w}(t=\epsilon) = \hat{W}\hat{w}(t=0),$$

resulting in

$$\hat{w}(t=\epsilon) = \begin{pmatrix} 1/4 \\ 1/2 \\ 0 \\ 1/4 \end{pmatrix}.$$

The next iteration results in

$$\hat{w}(t=2\epsilon) = \hat{W}\hat{w}(t=\epsilon),$$

resulting in

$$\hat{w}(t=2\epsilon) = \begin{pmatrix} 0.201389 \\ 0.319444 \\ 0.055556 \\ 0.423611 \end{pmatrix}.$$

Note that the vector $\hat{w}$ is always normalized to 1.

We find the steady state of the system by solving the set of equations

$$w(t=\infty) = Ww(t=\infty),$$

which is an eigenvalue problem with eigenvalue equal to **one**! This set of equations reads

$$\begin{aligned}
W_{11}w_1(t=\infty) + W_{12}w_2(t=\infty) + W_{13}w_3(t=\infty) + W_{14}w_4(t=\infty) &= w_1(t=\infty) \\
W_{21}w_1(t=\infty) + W_{22}w_2(t=\infty) + W_{23}w_3(t=\infty) + W_{24}w_4(t=\infty) &= w_2(t=\infty) \\
W_{31}w_1(t=\infty) + W_{32}w_2(t=\infty) + W_{33}w_3(t=\infty) + W_{34}w_4(t=\infty) &= w_3(t=\infty) \\
W_{41}w_1(t=\infty) + W_{42}w_2(t=\infty) + W_{43}w_3(t=\infty) + W_{44}w_4(t=\infty) &= w_4(t=\infty)
\end{aligned} \tag{2}$$

with the constraint that

$$\sum_i w_i(t=\infty) = 1,$$

yielding as solution

$$\hat{w}(t=\infty) = \begin{pmatrix} 0.244318 \\ 0.319602 \\ 0.056818 \\ 0.379261 \end{pmatrix}.$$

The table here demonstrates the convergence as a function of the number of iterations or time steps. After twelve iterations we have reached the exact value with six leading digits.

Iteration	w_1	w_2	w_3	w_4
0	1.000000	0.000000	0.000000	0.000000
1	0.250000	0.500000	0.000000	0.250000
2	0.201389	0.319444	0.055556	0.423611
3	0.247878	0.312886	0.056327	0.382909
4	0.245494	0.321106	0.055888	0.377513
5	0.243847	0.319941	0.056636	0.379575
6	0.244274	0.319547	0.056788	0.379391
7	0.244333	0.319611	0.056801	0.379255
8	0.244314	0.319610	0.056813	0.379264
9	0.244317	0.319603	0.056817	0.379264
10	0.244318	0.319602	0.056818	0.379262
11	0.244318	0.319602	0.056818	0.379261
12	0.244318	0.319602	0.056818	0.379261
$\hat{w}(t=\infty)$	0.244318	0.319602	0.056818	0.379261

We have after t-steps

$$\hat{w}(t) = \hat{W}^t \hat{w}(0),$$

with $\hat{w}(0)$ the distribution at $t = 0$ and $\hat{W}$ representing the transition probability matrix.

We can always expand $\hat{w}(0)$ in terms of the right eigenvectors $\hat{v}$ of $\hat{W}$ as

$$\hat{w}(0) = \sum_i \alpha_i \hat{v}_i,$$

resulting in

$$\hat{w}(t) = \hat{W}^t \hat{w}(0) = \hat{W}^t \sum_i \alpha_i \hat{v}_i = \sum_i \lambda_i^t \alpha_i \hat{v}_i,$$

with λ_i the i^{th} eigenvalue corresponding to the eigenvector $\hat{v}_i$.

If we assume that λ_0 is the largest eigenvector we see that in the limit $t \to \infty$, $\hat{w}(t)$ becomes proportional to the corresponding eigenvector $\hat{v}_0$. This is our steady state or final distribution.

Let us recapitulate some of our results about Markov chains and random walks.

- The time development of our PDF $w(t)$, after

one time-step from $t = 0$ is given by

$$w_i(t = \epsilon) = W(j \to i) w_j(t = 0).$$

This equation represents the discretized time-development of an original PDF. We can rewrite this as a

$$w_i(t = \epsilon) = W_{ij} w_j(t = 0).$$

with the transition matrix W for a random walk given by

$$W_{ij}(\epsilon) = W(il - jl, \epsilon) = \begin{cases} \frac{1}{2} & |i - j| = 1 \\ 0 & \text{else} \end{cases}$$

We call W_{ij} for the transition probability and we represent it as a matrix.

- Both W and w represent probabilities and they have to be normalized, meaning that at each time step we have

$$\sum_i w_i(t) = 1,$$

and

$$\sum_j W(j \to i) = 1.$$

Here we have written the previous matrix $W_{ij} = W(j \to i)$.

The further constraints are $0 \leq W_{ij} \leq 1$ and $0 \leq w_j \leq 1$.

- We can thus write the action of W as

$$w_i(t + 1) = \sum_j W_{ij} w_j(t),$$

or as vector-matrix relation

$$\hat{w}(t + 1) = \hat{W}\hat{w}(t),$$

and if we have that $||\hat{w}(t + 1) - \hat{w}(t)|| \to 0$, we say that we have reached the most likely state of the system, the so-called steady state or equilibrium state.

Another way of phrasing this is

$$w(t = \infty) = W w(t = \infty). \tag{3}$$

The question then is how can we model anything under such a severe lack of knowledge? The Metropolis algorithm comes to our rescue here. Since $W(j \to i)$ is unknown, we model it as the product of two probabilities, a probability for accepting the proposed move from the state j to the state j, and a probability for making the transition to the state i being in the state j. We label these probabilities $A(j \to i)$ and $T(j \to i)$, respectively. Our total transition probability is then

$$W(j \to i) = T(j \to i)A(j \to i).$$

The algorithm can then be expressed as

- We make a suggested move to the new state i with some transition or moving probability $T_{j\rightarrow i}$.
- We accept this move to the new state with an acceptance probability $A_{j\rightarrow i}$. The new state i is in turn used as our new starting point for the next move. We reject this proposed moved with a $1 - A_{j\rightarrow i}$ and the original state j is used again as a sample.

We wish to derive the required properties of the probabilities T and A such that $w_i^{(t\rightarrow\infty)} \rightarrow w_i$, starting from any distribution, will lead us to the correct distribution.

We can now derive the dynamical process towards equilibrium. To obtain this equation we note that after t time steps the probability for being in a state i is related to the probability of being in a state j and performing a transition to the new state together with the probability of actually being in the state i and making a move to any of the possible states j from the previous time step.

We can express this as, assuming that T and A are time-independent,

$$w_i(t+1) = \sum_j \left[w_j(t) T_{j\rightarrow i} A_{j\rightarrow i} + w_i(t) T_{i\rightarrow j} \left(1 - A_{i\rightarrow j}\right) \right] .$$

All probabilities are normalized, meaning that $\sum_j T_{i\rightarrow j} = 1$. Using the latter, we can rewrite the previous equation as

$$w_i(t+1) = w_i(t) + \sum_j \left[w_j(t) T_{j\rightarrow i} A_{j\rightarrow i} - w_i(t) T_{i\rightarrow j} A_{i\rightarrow j} \right] ,$$

which can be rewritten as

$$w_i(t+1) - w_i(t) = \sum_j \left[w_j(t) T_{j\rightarrow i} A_{j\rightarrow i} - w_i(t) T_{i\rightarrow j} A_{i\rightarrow j} \right] .$$

The last equation is very similar to the so-called Master equation, which relates the temporal dependence of a PDF $w_i(t)$ to various transition rates. The equation can be derived from the so-called Chapman-Einstein-Enskog-Kolmogorov equation. The equation is given as

$$\frac{dw_i(t)}{dt} = \sum_j \left[W(j \rightarrow i) w_j - W(i \rightarrow j) w_i \right] , \tag{4}$$

which simply states that the rate at which the systems moves from a state j to a final state i (the first term on the right-hand side of the last equation) is balanced by the rate at which the system undergoes transitions from the state i to a state j (the second term). If we have reached the so-called steady state, then the temporal development is zero. This means that in equilibrium we have

$$\frac{dw_i(t)}{dt} = 0.$$

In the limit $t \to \infty$ we require that the two distributions $w_i(t+1) = w_i$ and $w_i(t) = w_i$ and we have

$$\sum_j w_j T_{j\to i} A_{j\to i} = \sum_j w_i T_{i\to j} A_{i\to j},$$

which is the condition for balance when the most likely state (or steady state) has been reached. We see also that the right-hand side can be rewritten as

$$\sum_j w_i T_{i\to j} A_{i\to j} = \sum_j w_i W_{i\to j},$$

and using the property that $\sum_j W_{i\to j} = 1$, we can rewrite our equation as

$$w_i = \sum_j w_j T_{j\to i} A_{j\to i} = \sum_j w_j W_{j\to i},$$

which is nothing but the standard equation for a Markov chain when the steady state has been reached.

However, the condition that the rates should equal each other is in general not sufficient to guarantee that we, after many simulations, generate the correct distribution. We may risk to end up with so-called cyclic solutions. To avoid this we therefore introduce an additional condition, namely that of detailed balance

$$W(j \to i) w_j = W(i \to j) w_i.$$

These equations were derived by Lars Onsager when studying irreversible processes. At equilibrium detailed balance gives thus

$$\frac{W(j \to i)}{W(i \to j)} = \frac{w_i}{w_j}.$$

Rewriting the last equation in terms of our transition probabilities T and acceptance probobalities A we obtain

$$w_j(t) T_{j\to i} A_{j\to i} = w_i(t) T_{i\to j} A_{i\to j}.$$

Since we normally have an expression for the probability distribution functions w_i, we can rewrite the last equation as

$$\frac{T_{j\to i} A_{j\to i}}{T_{i\to j} A_{i\to j}} = \frac{w_i}{w_j}.$$

In statistical physics this condition ensures that it is e.g., the Boltzmann distribution which is generated when equilibrium is reached.

We introduce now the Boltzmann distribution

$$w_i = \frac{\exp\left(-\beta(E_i)\right)}{Z},$$

which states that the probability of finding the system in a state i with energy E_i at an inverse temperature $\beta = 1/k_B T$ is $w_i \propto \exp\left(-\beta(E_i)\right)$. The denominator

Z is a normalization constant which ensures that the sum of all probabilities is normalized to one. It is defined as the sum of probabilities over all microstates j of the system

$$Z = \sum_j \exp\left(-\beta(E_i)\right).$$

From the partition function we can in principle generate all interesting quantities for a given system in equilibrium with its surroundings at a temperature T.

With the probability distribution given by the Boltzmann distribution we are now in a position where we can generate expectation values for a given variable A through the definition

$$\langle A\rangle = \sum_j A_j w_j = \frac{\sum_j A_j \exp\left(-\beta(E_j)\right.}{Z}.$$

In general, most systems have an infinity of microstates making thereby the computation of Z practically impossible and a brute force Monte Carlo calculation over a given number of randomly selected microstates may therefore not yield those microstates which are important at equilibrium. To select the most important contributions we need to use the condition for detailed balance. Since this is just given by the ratios of probabilities, we never need to evaluate the partition function Z.

For the Boltzmann distribution, detailed balance results in

$$\frac{w_i}{w_j} = \exp\left(-\beta(E_i - E_j)\right).$$

Let us now specialize to a system whose energy is defined by the orientation of single spins. Consider the state i, with given energy E_i represented by the following N spins

$$\begin{array}{cccccccccc} \uparrow & \uparrow & \uparrow & \dots & \uparrow & \downarrow & \uparrow & \dots & \uparrow & \downarrow \\ 1 & 2 & 3 & \dots & k-1 & k & k+1 & \dots & N-1 & N \end{array}$$

We are interested in the transition with one single spinflip to a new state j with energy E_j

$$\begin{array}{cccccccccc} \uparrow & \uparrow & \uparrow & \dots & \uparrow & \uparrow & \uparrow & \dots & \uparrow & \downarrow \\ 1 & 2 & 3 & \dots & k-1 & k & k+1 & \dots & N-1 & N \end{array}$$

This change from one microstate i (or spin configuration) to another microstate j is the configuration space analogue to a random walk on a lattice. Instead of jumping from one place to another in space, we 'jump' from one microstate to another.

However, the selection of states has to generate a final distribution which is the Boltzmann distribution. This is again the same we saw for a random walker, for the discrete case we had always a binomial distribution, whereas for the

continuous case we had a normal distribution. The way we sample configurations should result, when equilibrium is established, in the Boltzmann distribution. Else, our algorithm for selecting microstates is wrong.

As stated above, we do in general not know the closed-form expression of the transition rate and we are free to model it as $W(i \to j) = T(i \to j)A(i \to j)$. Our ratio between probabilities gives us

$$\frac{A_{j\to i}}{A_{i\to j}} = \frac{w_i T_{i\to j}}{w_j T_{j\to i}}.$$

The simplest form of the Metropolis algorithm (sometimes called for brute force Metropolis) assumes that the transition probability $T(i \to j)$ is symmetric, implying that $T(i \to j) = T(j \to i)$.

We obtain then (using the Boltzmann distribution)

$$\frac{A(j \to i)}{A(i \to j)} = \exp\left(-\beta(E_i - E_j)\right).$$

We are in this case interested in a new state E_j whose energy is lower than E_i, viz., $\Delta E = E_j - E_i \leq 0$. A simple test would then be to accept only those microstates which lower the energy. Suppose we have ten microstates with energy $E_0 \leq E_1 \leq E_2 \leq E_3 \leq \cdots \leq E_9$. Our desired energy is E_0.

At a given temperature T we start our simulation by randomly choosing state E_9. Flipping spins we may then find a path from $E_9 \to E_8 \to E_7 \cdots \to E_1 \to E_0$. This would however lead to biased statistical averages since it would violate the ergodic hypothesis discussed in the previous section. This principle states that it should be possible for any Markov process to reach every possible state of the system from any starting point if the simulations is carried out for a long enough time.

Any state in a Boltzmann distribution has a probability different from zero and if such a state cannot be reached from a given starting point, then the system is not ergodic. This means that another possible path to E_0 could be $E_9 \to E_7 \to E_8 \cdots \to E_9 \to E_5 \to E_0$ and so forth. Even though such a path could have a negligible probability it is still a possibility, and if we simulate long enough it should be included in our computation of an expectation value.

Thus, we require that our algorithm should satisfy the principle of detailed balance and be ergodic. The problem with our ratio

$$\frac{A(j \to i)}{A(i \to j)} = \exp\left(-\beta(E_i - E_j)\right),$$

is that we do not know the acceptance probability. This equation only specifies the ratio of pairs of probabilities. Normally we want an algorithm which is as efficient as possible and maximizes the number of accepted moves. Moreover, we know that the acceptance probability has 0 as its smallest value and 1 as its largest. If we assume that the largest possible acceptance probability is 1, we adjust thereafter the other acceptance probability to this constraint.

To understand this better, assume that we have two energies, E_i and E_j, with $E_i < E_j$. This means that the largest acceptance value must be $A(j \rightarrow i)$ since we move to a state with lower energy. It follows from also from the fact that the probability w_i is larger than w_j. The trick then is to fix this value to $A(j \rightarrow i) = 1$. It means that the other acceptance probability has to be

$$A(i \rightarrow j) = \exp\left(-\beta(E_j - E_i)\right).$$

One possible way to encode this equation reads

$$A(j \rightarrow i) = \begin{cases} \exp\left(-\beta(E_i - E_j)\right) & E_i - E_j > 0 \\ 1 & else \end{cases},$$

implying that if we move to a state with a lower energy, we always accept this move with acceptance probability $A(j \rightarrow i) = 1$. If the energy is higher, we need to check this acceptance probability with the ratio between the probabilities from our PDF. From a practical point of view, the above ratio is compared with a random number. If the ratio is smaller than a given random number we accept the move to a higher energy, else we stay in the same state.

Nothing hinders us obviously in choosing another acceptance ratio, like a weighting of the two energies via

$$A(j \rightarrow i) = \exp\left(-\frac{1}{2}\beta(E_i - E_j)\right).$$

However, it is easy to see that such an acceptance ratio would result in fewer accepted moves.

The Monte Carlo approach, combined with the theory for Markov chains can be summarized as follows: A Markov chain Monte Carlo method for the simulation of a distribution w is any method producing an ergodic Markov chain of events x whose stationary distribution is w. The Metropolis algorithm can be phrased as

- Generate an initial value $x^{(i)}$.
- Generate a trial value y_t with probability $T(y_t|x^{(i)})$. The latter quantity represents the probability of generating y_t given $x^{(i)}$.
- Take a new value

$$x^{(i+1)} = \begin{cases} y_t & \text{with probability} = A(x^{(i)} \rightarrow y_t) \\ x^{(i)} & \text{with probability} = 1 - A(x^{(i)} \rightarrow y_t) \end{cases}$$

- We have defined the transition (acceptance) probability as

$$A(x \rightarrow y) = \min\left\{\frac{w(y)T(x|y)}{w(x)T(y|x)}, 1\right\}.$$

Boltzmann Machines

Why use a generative model rather than the more well known discriminative deep neural networks (DNN)?

- Discriminitave methods have several limitations: They are mainly supervised learning methods, thus requiring labeled data. And there are tasks they cannot accomplish, like drawing new examples from an unknown probability distribution.
- A generative model can learn to represent and sample from a probability distribution. The core idea is to learn a parametric model of the probability distribution from which the training data was drawn. As an example
    1. A model for images could learn to draw new examples of cats and dogs, given a training dataset of images of cats and dogs.
    2. Generate a sample of an ordered or disordered Ising model phase, having been given samples of such phases.
    3. Model the trial function for Monte Carlo calculations
- Both use gradient-descent based learning procedures for minimizing cost functions
- Energy based models don't use backpropagation and automatic differentiation for computing gradients, instead turning to Markov Chain Monte Carlo methods.
- DNNs often have several hidden layers. A restricted Boltzmann machine has only one hidden layer, however several RBMs can be stacked to make up Deep Belief Networks, of which they constitute the building blocks.

History: The RBM was developed by amongst others Geoffrey Hinton, called by some the "Godfather of Deep Learning", working with the University of Toronto and Google.

A BM is what we would call an undirected probabilistic graphical model with stochastic continuous or discrete units.

It is interpreted as a stochastic recurrent neural network where the state of each unit(neurons/nodes) depends on the units it is connected to. The weights in the network represent thus the strength of the interaction between various units/nodes.

It turns into a Hopfield network if we choose deterministic rather than stochastic units. In contrast to a Hopfield network, a BM is a so-called generative model. It allows us to generate new samples from the learned distribution.

A standard BM network is divided into a set of observable and visible units $\hat{x}$ and a set of unknown hidden units/nodes $\hat{h}$.

Additionally there can be bias nodes for the hidden and visible layers. These biases are normally set to 1.

BMs are stackable, meaning they cwe can train a BM which serves as input to another BM. We can construct deep networks for learning complex PDFs. The layers can be trained one after another, a feature which makes them popular in deep learning

However, they are often hard to train. This leads to the introduction of so-called restricted BMs, or RBMS. Here we take away all lateral connections between nodes in the visible layer as well as connections between nodes in the hidden layer. The network is illustrated in the figure below.

The network

The network layers:

1. A function **x** that represents the visible layer, a vector of M elements (nodes). This layer represents both what the RBM might be given as training input, and what we want it to be able to reconstruct. This might for example be the pixels of an image, the spin values of the Ising model, or coefficients representing speech.

2. The function **h** represents the hidden, or latent, layer. A vector of N elements (nodes). Also called "feature detectors".

The goal of the hidden layer is to increase the model's expressive power. We 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).

Examples of this trick being employed in physics:

1. The Hubbard-Stratonovich transformation

2. The introduction of ghost fields in gauge theory

3. Shadow wave functions in Quantum Monte Carlo simulations

The network parameters, to be optimized/learned:

1. **a** represents the visible bias, a vector of same length as **x**.

2. **b** represents the hidden bias, a vector of same lenght as **h**.

3. W represents the interaction weights, a matrix of size $M \times N$.

Joint distribution. T he restricted Boltzmann machine is described by a Bolztmann distribution

$$P_{rbm}(\mathbf{x}, \mathbf{h}) = \frac{1}{Z} e^{-\frac{1}{T_0} E(\mathbf{x}, \mathbf{h})}, \tag{5}$$

where Z is the normalization constant or partition function, defined as

$$Z = \int \int e^{-\frac{1}{T_0} E(\mathbf{x}, \mathbf{h})} d\mathbf{x} d\mathbf{h}. \tag{6}$$

It is common to ignore T_0 by setting it to one.

Network Elements, the energy function. The function $E(\mathbf{x}, \mathbf{h})$ gives the **energy** of a configuration (pair of vectors) $(\mathbf{x}, \mathbf{h})$. The lower the energy of a configuration, the higher the probability of it. This function also depends on the parameters $\mathbf{a}$, $\mathbf{b}$ and W. Thus, when we adjust them during the learning procedure, we are adjusting the energy function to best fit our problem.

An expression for the energy function is

$$E(\hat{x}, \hat{h}) = -\sum_{ia}^{NA} b_i^a \alpha_i^a(x_i) - \sum_{jd}^{MD} c_j^d \beta_j^d(h_j) - \sum_{ijad}^{NAMD} b_i^a \alpha_i^a(x_i) c_j^d \beta_j^d(h_j) w_{ij}^{ad}.$$

Here $\beta_j^d(h_j)$ and $\alpha_i^a(x_j)$ are so-called transfer functions that map a given input value to a desired feature value. The labels a and d denote that there can be multiple transfer functions per variable. The first sum depends only on the visible units. The second on the hidden ones. **Note** that there is no connection between nodes in a layer.

The quantities b and c can be interpreted as the visible and hidden biases, respectively.

The connection between the nodes in the two layers is given by the weights w_{ij}.

Defining different types of RBMs. There are different variants of RBMs, and the differences lie in the types of visible and hidden units we choose as well as in the implementation of the energy function $E(\mathbf{x}, \mathbf{h})$.

Binary-Binary RBM: RBMs were first developed using binary units in both the visible and hidden layer. The corresponding energy function is defined as follows:

$$E(\mathbf{x}, \mathbf{h}) = -\sum_{i}^{M} x_i a_i - \sum_{j}^{N} b_j h_j - \sum_{i,j}^{M,N} x_i w_{ij} h_j, \tag{7}$$

where the binary values taken on by the nodes are most commonly 0 and 1.

Gaussian-Binary RBM: Another varient is the RBM where the visible units are Gaussian while the hidden units remain binary:

$$E(\mathbf{x}, \mathbf{h}) = \sum_{i}^{M} \frac{(x_i - a_i)^2}{2\sigma_i^2} - \sum_{j}^{N} b_j h_j - \sum_{i,j}^{M,N} \frac{x_i w_{ij} h_j}{\sigma_i^2}. \tag{8}$$

1. RBMs are Useful when we model continuous data (i.e., we wish $\mathbf{x}$ to be continuous)
2. Requires a smaller learning rate, since there's no upper bound to the value a component might take in the reconstruction

Other types of units include:

1. Softmax and multinomial units
2. Gaussian visible and hidden units
3. Binomial units
4. Rectified linear units

Cost function. When working with a training dataset, the most common training approach is maximizing the log-likelihood of the training data. The log likelihood characterizes the log-probability of generating the observed data using our generative model. Using this method our cost function is chosen as the negative log-likelihood. The learning then consists of trying to find parameters that maximize the probability of the dataset, and is known as Maximum Likelihood Estimation (MLE). Denoting the parameters as $\boldsymbol{\theta} = a_1, ..., a_M, b_1, ..., b_N, w_{11}, ..., w_{MN}$, the log-likelihood is given by

$$\mathcal{L}(\{\theta_i\}) = \langle \log P_\theta(\boldsymbol{x}) \rangle_{data} \tag{9}$$

$$= -\langle E(\boldsymbol{x}; \{\theta_i\}) \rangle_{data} - \log Z(\{\theta_i\}), \tag{10}$$

where we used that the normalization constant does not depend on the data, $\langle \log Z(\{\theta_i\}) \rangle = \log Z(\{\theta_i\})$ Our cost function is the negative log-likelihood, $\mathcal{C}(\{\theta_i\}) = -\mathcal{L}(\{\theta_i\})$

Optimization / Training. The training procedure of choice often is Stochastic Gradient Descent (SGD). It consists of a series of iterations where we update the parameters according to the equation

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \nabla \mathcal{C}(\boldsymbol{\theta}_k) \tag{11}$$

at each k-th iteration. There are a range of variants of the algorithm which aim at making the learning rate η more adaptive so the method might be more efficient while remaining stable.

We now need the gradient of the cost function in order to minimize it. We find that

$$\frac{\partial \mathcal{C}(\{\theta_i\})}{\partial \theta_i} = \langle \frac{\partial E(\boldsymbol{x};\theta_i)}{\partial \theta_i} \rangle_{data} + \frac{\partial \log Z(\{\theta_i\})}{\partial \theta_i} \tag{12}$$

$$= \langle O_i(\boldsymbol{x}) \rangle_{data} - \langle O_i(\boldsymbol{x}) \rangle_{model}, \tag{13}$$

where in order to simplify notation we defined the "operator"

$$O_i(\boldsymbol{x}) = \frac{\partial E(\boldsymbol{x};\theta_i)}{\partial \theta_i}, \tag{14}$$

and used the statistical mechanics relationship between expectation values and the log-partition function:

$$\langle O_i(\boldsymbol{x}) \rangle_{model} = \mathrm{Tr} P_\theta(\boldsymbol{x}) O_i(\boldsymbol{x}) = -\frac{\partial \log Z(\{\theta_i\})}{\partial \theta_i}. \tag{15}$$

The data-dependent term in the gradient is known as the positive phase of the gradient, while the model-dependent term is known as the negative phase of the gradient. The aim of the training is to lower the energy of configurations that are near observed data points (increasing their probability), and raising the energy of configurations that are far from observed data points (decreasing their probability).

The gradient of the negative log-likelihood cost function of a Binary-Binary RBM is then

$$\frac{\partial \mathcal{C}(w_{ij}, a_i, b_j)}{\partial w_{ij}} = \langle x_i h_j \rangle_{data} - \langle x_i h_j \rangle_{model} \tag{16}$$

$$\frac{\partial \mathcal{C}(w_{ij}, a_i, b_j)}{\partial a_{ij}} = \langle x_i \rangle_{data} - \langle x_i \rangle_{model} \tag{17}$$

$$\frac{\partial \mathcal{C}(w_{ij}, a_i, b_j)}{\partial b_{ij}} = \langle h_i \rangle_{data} - \langle h_i \rangle_{model}. \tag{18}$$

$$\tag{19}$$

To get the expectation values with respect to the *data*, we set the visible units to each of the observed samples in the training data, then update the hidden units according to the conditional probability found before. We then average over all samples in the training data to calculate expectation values with respect to the data.

Kullback-Leibler relative entropy. When the goal of the training is to approximate a probability distribution, as it is in generative modeling, another relevant measure is the **Kullback-Leibler divergence**, also known as the relative entropy or Shannon entropy. It is a non-symmetric measure of the dissimilarity between two probability density functions p and q. If p is the

unkown probability which we approximate with q, we can measure the difference by

$$\mathrm{KL}(p||q) = \int_{-\infty}^{\infty} p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} d\boldsymbol{x}. \tag{20}$$

Thus, the Kullback-Leibler divergence between the distribution of the training data $f(\boldsymbol{x})$ and the model distribution $p(\boldsymbol{x}|\boldsymbol{\theta})$ is

$$\begin{aligned}
\mathrm{KL}(f(\boldsymbol{x})||p(\boldsymbol{x}|\boldsymbol{\theta})) &= \int_{-\infty}^{\infty} f(\boldsymbol{x}) \log \frac{f(\boldsymbol{x})}{p(\boldsymbol{x}|\boldsymbol{\theta})} d\boldsymbol{x} && (21)\\
&= \int_{-\infty}^{\infty} f(\boldsymbol{x}) \log f(\boldsymbol{x}) d\boldsymbol{x} - \int_{-\infty}^{\infty} f(\boldsymbol{x}) \log p(\boldsymbol{x}|\boldsymbol{\theta}) d\boldsymbol{x} && (22)\\
&= \langle \log f(\boldsymbol{x}) \rangle_{f(\boldsymbol{x})} - \langle \log p(\boldsymbol{x}|\boldsymbol{\theta}) \rangle_{f(\boldsymbol{x})} && (23)\\
&= \langle \log f(\boldsymbol{x}) \rangle_{data} + \langle E(\boldsymbol{x}) \rangle_{data} + \log Z && (24)\\
&= \langle \log f(\boldsymbol{x}) \rangle_{data} + \mathcal{C}_{LL}. && (25)
\end{aligned}$$

The first term is constant with respect to $\boldsymbol{\theta}$ since $f(\boldsymbol{x})$ is independent of $\boldsymbol{\theta}$. Thus the Kullback-Leibler Divergence is minimal when the second term is minimal. The second term is the log-likelihood cost function, hence minimizing the Kullback-Leibler divergence is equivalent to maximizing the log-likelihood.

To further understand generative models it is useful to study the gradient of the cost function which is needed in order to minimize it using methods like stochastic gradient descent.

The partition function is the generating function of expectation values, in particular there are mathematical relationships between expectation values and the log-partition function. In this case we have

$$\langle \frac{\partial E(\boldsymbol{x}; \theta_i)}{\partial \theta_i} \rangle_{model} = \int p(\boldsymbol{x}|\boldsymbol{\theta}) \frac{\partial E(\boldsymbol{x}; \theta_i)}{\partial \theta_i} d\boldsymbol{x} = -\frac{\partial \log Z(\theta_i)}{\partial \theta_i}. \tag{26}$$

Here $\langle \cdot \rangle_{model}$ is the expectation value over the model probability distribution $p(\boldsymbol{x}|\boldsymbol{\theta})$.

Setting up for gradient descent calculations

Using the previous relationship we can express the gradient of the cost function as

$$\begin{aligned}
\frac{\partial \mathcal{C}_{LL}}{\partial \theta_i} &= \langle \frac{\partial E(\boldsymbol{x}; \theta_i)}{\partial \theta_i} \rangle_{data} + \frac{\partial \log Z(\theta_i)}{\partial \theta_i} && (27)\\
&= \langle \frac{\partial E(\boldsymbol{x}; \theta_i)}{\partial \theta_i} \rangle_{data} - \langle \frac{\partial E(\boldsymbol{x}; \theta_i)}{\partial \theta_i} \rangle_{model} && (28)\\
& && (29)
\end{aligned}$$

This expression shows that the gradient of the log-likelihood cost function is a **difference of moments**, with one calculated from the data and one calculated from the model. The data-dependent term is called the **positive phase** and the model-dependent term is called the **negative phase** of the gradient. We see now that minimizing the cost function results in lowering the energy of configurations $\boldsymbol{x}$ near points in the training data and increasing the energy of configurations not observed in the training data. That means we increase the model's probability of configurations similar to those in the training data.

The gradient of the cost function also demonstrates why gradients of unsupervised, generative models must be computed differently from for those of for example FNNs. While the data-dependent expectation value is easily calculated based on the samples $\boldsymbol{x}_i$ in the training data, we must sample from the model in order to generate samples from which to caclulate the model-dependent term. We sample from the model by using MCMC-based methods. We can not sample from the model directly because the partition function Z is generally intractable.

As in supervised machine learning problems, the goal is also here to perform well on **unseen** data, that is to have good generalization from the training data. The distribution $f(x)$ we approximate is not the **true** distribution we wish to estimate, it is limited to the training data. Hence, in unsupervised training as well it is important to prevent overfitting to the training data. Thus it is common to add regularizers to the cost function in the same manner as we discussed for say linear regression.

Because we are restricted to potential functions which are positive it is convenient to express them as exponentials, so that

$$\phi_C(\boldsymbol{x}_C) = e^{-E_C(\boldsymbol{x}_C)} \tag{30}$$

where $E(\boldsymbol{x}_C)$ is called an *energy function*, and the exponential representation is the *Boltzmann distribution*. The joint distribution is defined as the product of potentials.

The joint distribution of the random variables is then

$$\begin{aligned} p(\boldsymbol{x}) =& \frac{1}{Z}\prod_C \phi_C(\boldsymbol{x}_C) \\ =& \frac{1}{Z}\prod_C e^{-E_C(\boldsymbol{x}_C)} \\ =& \frac{1}{Z} e^{-\sum_C E_C(\boldsymbol{x}_C)} \\ =& \frac{1}{Z} e^{-E(\boldsymbol{x})}. \end{aligned} \tag{31}$$

$$p_{BM}(\boldsymbol{x}, \boldsymbol{h}) = \frac{1}{Z_{BM}} e^{-\frac{1}{T}E_{BM}(\boldsymbol{x},\boldsymbol{h})}, \tag{32}$$

with the partition function

$$Z_{BM} = \int\int e^{-\frac{1}{T}E_{BM}(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{h}})} d\tilde{\boldsymbol{x}} d\tilde{\boldsymbol{h}}. \tag{33}$$

T is a physics-inspired parameter named temperature and will be assumed to be 1 unless otherwise stated. The energy function of the Boltzmann machine determines the interactions between the nodes and is defined

$$\begin{aligned} E_{BM}(\boldsymbol{x},\boldsymbol{h}) = &- \sum_{i,k}^{M,K} a_i^k \alpha_i^k(x_i) - \sum_{j,l}^{N,L} b_j^l \beta_j^l(h_j) - \sum_{i,j,k,l}^{M,N,K,L} \alpha_i^k(x_i) w_{ij}^{kl} \beta_j^l(h_j) \\ &- \sum_{i,m=i+1,k}^{M,M,K} \alpha_i^k(x_i) v_{im}^k \alpha_m^k(x_m) - \sum_{j,n=j+1,l}^{N,N,L} \beta_j^l(h_j) u_{jn}^l \beta_n^l(h_n). \end{aligned} \tag{34}$$

Here $\alpha_i^k(x_i)$ and $\beta_j^l(h_j)$ are one-dimensional transfer functions or mappings from the given input value to the desired feature value. They can be arbitrary functions of the input variables and are independent of the parameterization (parameters referring to weight and biases), meaning they are not affected by training of the model. The indices k and l indicate that there can be multiple transfer functions per variable. Furthermore, a_i^k and b_j^l are the visible and hidden bias. w_{ij}^{kl} are weights of the **inter-layer** connection terms which connect visible and hidden units. v_{im}^k and u_{jn}^l are weights of the **intra-layer** connection terms which connect the visible units to each other and the hidden units to each other, respectively.

We remove the intra-layer connections by setting v_{im} and u_{jn} to zero. The expression for the energy of the RBM is then

$$E_{RBM}(\boldsymbol{x},\boldsymbol{h}) = - \sum_{i,k}^{M,K} a_i^k \alpha_i^k(x_i) - \sum_{j,l}^{N,L} b_j^l \beta_j^l(h_j) - \sum_{i,j,k,l}^{M,N,K,L} \alpha_i^k(x_i) w_{ij}^{kl} \beta_j^l(h_j). \tag{35}$$

resulting in

$$
\begin{aligned}
P_{RBM}(\boldsymbol{x}) =& \int P_{RBM}(\boldsymbol{x}, \tilde{\boldsymbol{h}}) d\tilde{\boldsymbol{h}} \\
=& \frac{1}{Z_{RBM}} \int e^{-E_{RBM}(\boldsymbol{x}, \tilde{\boldsymbol{h}})} d\tilde{\boldsymbol{h}} \\
=& \frac{1}{Z_{RBM}} \int e^{\sum_{i,k} a_i^k \alpha_i^k(x_i) + \sum_{j,l} b_j^l \beta_j^l(\tilde{h}_j) + \sum_{i,j,k,l} \alpha_i^k(x_i) w_{ij}^{kl} \beta_j^l(\tilde{h}_j)} d\tilde{\boldsymbol{h}} \\
=& \frac{1}{Z_{RBM}} e^{\sum_{i,k} a_i^k \alpha_i^k(x_i)} \int \prod_j^N e^{\sum_l b_j^l \beta_j^l(\tilde{h}_j) + \sum_{i,k,l} \alpha_i^k(x_i) w_{ij}^{kl} \beta_j^l(\tilde{h}_j)} d\tilde{\boldsymbol{h}} \\
=& \frac{1}{Z_{RBM}} e^{\sum_{i,k} a_i^k \alpha_i^k(x_i)} \Bigg(\int e^{\sum_l b_1^l \beta_1^l(\tilde{h}_1) + \sum_{i,k,l} \alpha_i^k(x_i) w_{i1}^{kl} \beta_1^l(\tilde{h}_1)} d\tilde{h}_1 \\
& \times \int e^{\sum_l b_2^l \beta_2^l(\tilde{h}_2) + \sum_{i,k,l} \alpha_i^k(x_i) w_{i2}^{kl} \beta_2^l(\tilde{h}_2)} d\tilde{h}_2 \\
& \times ... \\
& \times \int e^{\sum_l b_N^l \beta_N^l(\tilde{h}_N) + \sum_{i,k,l} \alpha_i^k(x_i) w_{iN}^{kl} \beta_N^l(\tilde{h}_N)} d\tilde{h}_N \Bigg) \\
=& \frac{1}{Z_{RBM}} e^{\sum_{i,k} a_i^k \alpha_i^k(x_i)} \prod_j^N \int e^{\sum_l b_j^l \beta_j^l(\tilde{h}_j) + \sum_{i,k,l} \alpha_i^k(x_i) w_{ij}^{kl} \beta_j^l(\tilde{h}_j)} d\tilde{h}_j
\end{aligned}
\tag{36}
$$

Similarly

$$
\begin{aligned}
P_{RBM}(\boldsymbol{h}) =& \frac{1}{Z_{RBM}} \int e^{-E_{RBM}(\tilde{\boldsymbol{x}}, \boldsymbol{h})} d\tilde{\boldsymbol{x}} \\
=& \frac{1}{Z_{RBM}} e^{\sum_{j,l} b_j^l \beta_j^l(h_j)} \prod_i^M \int e^{\sum_k a_i^k \alpha_i^k(\tilde{x}_i) + \sum_{j,k,l} \alpha_i^k(\tilde{x}_i) w_{ij}^{kl} \beta_j^l(h_j)} d\tilde{x}_i
\end{aligned}
\tag{37}
$$

Using Bayes theorem

$$
\begin{aligned}
P_{RBM}(\boldsymbol{h}|\boldsymbol{x}) =& \frac{P_{RBM}(\boldsymbol{x}, \boldsymbol{h})}{P_{RBM}(\boldsymbol{x})} \\
=& \frac{\frac{1}{Z_{RBM}} e^{\sum_{i,k} a_i^k \alpha_i^k(x_i) + \sum_{j,l} b_j^l \beta_j^l(h_j) + \sum_{i,j,k,l} \alpha_i^k(x_i) w_{ij}^{kl} \beta_j^l(h_j)}}{\frac{1}{Z_{RBM}} e^{\sum_{i,k} a_i^k \alpha_i^k(x_i)} \prod_j^N \int e^{\sum_l b_j^l \beta_j^l(\tilde{h}_j) + \sum_{i,k,l} \alpha_i^k(x_i) w_{ij}^{kl} \beta_j^l(\tilde{h}_j)} d\tilde{h}_j} \\
=& \prod_j^N \frac{e^{\sum_l b_j^l \beta_j^l(h_j) + \sum_{i,k,l} \alpha_i^k(x_i) w_{ij}^{kl} \beta_j^l(h_j)}}{\int e^{\sum_l b_j^l \beta_j^l(\tilde{h}_j) + \sum_{i,k,l} \alpha_i^k(x_i) w_{ij}^{kl} \beta_j^l(\tilde{h}_j)} d\tilde{h}_j}
\end{aligned}
\tag{38}
$$

Similarly

$$P_{RBM}(\boldsymbol{x}|\boldsymbol{h}) = \frac{P_{RBM}(\boldsymbol{x},\boldsymbol{h})}{P_{RBM}(\boldsymbol{h})}$$
$$= \prod_i^M \frac{e^{\sum_k a_i^k \alpha_i^k(x_i) + \sum_{j,k,l} \alpha_i^k(x_i) w_{ij}^{kl} \beta_j^l(h_j)}}{\int e^{\sum_k a_i^k \alpha_i^k(\tilde{x}_i) + \sum_{j,k,l} \alpha_i^k(\tilde{x}_i) w_{ij}^{kl} \beta_j^l(h_j)} d\tilde{x}_i} \tag{39}$$

The original RBM had binary visible and hidden nodes. They were showned to be universal approximators of discrete distributions. It was also shown that adding hidden units yields strictly improved modelling power. The common choice of binary values are 0 and 1. However, in some physics applications, -1 and 1 might be a more natural choice. We will here use 0 and 1.

$$E_{BB}(\boldsymbol{x}, \mathbf{h}) = -\sum_i^M x_i a_i - \sum_j^N b_j h_j - \sum_{i,j}^{M,N} x_i w_{ij} h_j. \tag{40}$$

$$p_{BB}(\boldsymbol{x},\boldsymbol{h}) = \frac{1}{Z_{BB}} e^{\sum_i^M a_i x_i + \sum_j^N b_j h_j + \sum_{ij}^{M,N} x_i w_{ij} h_j} \tag{41}$$
$$= \frac{1}{Z_{BB}} e^{\boldsymbol{x}^T \boldsymbol{a} + \boldsymbol{b}^T \boldsymbol{h} + \boldsymbol{x}^T \boldsymbol{W} \boldsymbol{h}} \tag{42}$$

with the partition function

$$Z_{BB} = \sum_{\boldsymbol{x},\boldsymbol{h}} e^{\boldsymbol{x}^T \boldsymbol{a} + \boldsymbol{b}^T \boldsymbol{h} + \boldsymbol{x}^T \boldsymbol{W} \boldsymbol{h}}. \tag{43}$$

Marginal Probability Density Functions. In order to find the probability of any configuration of the visible units we derive the marginal probability density function.

$$
\begin{aligned}
p_{BB}(\boldsymbol{x}) =& \sum_{\boldsymbol{h}} p_{BB}(\boldsymbol{x}, \boldsymbol{h}) && (44)\\
=& \frac{1}{Z_{BB}} \sum_{\boldsymbol{h}} e^{\boldsymbol{x}^T \boldsymbol{a} + \boldsymbol{b}^T \boldsymbol{h} + \boldsymbol{x}^T \boldsymbol{W} \boldsymbol{h}} \\
=& \frac{1}{Z_{BB}} e^{\boldsymbol{x}^T \boldsymbol{a}} \sum_{\boldsymbol{h}} e^{\sum_j^N (b_j + \boldsymbol{x}^T \boldsymbol{w}_{*j}) h_j} \\
=& \frac{1}{Z_{BB}} e^{\boldsymbol{x}^T \boldsymbol{a}} \sum_{\boldsymbol{h}} \prod_j^N e^{(b_j + \boldsymbol{x}^T \boldsymbol{w}_{*j}) h_j} \\
=& \frac{1}{Z_{BB}} e^{\boldsymbol{x}^T \boldsymbol{a}} \Bigg(\sum_{h_1} e^{(b_1 + \boldsymbol{x}^T \boldsymbol{w}_{*1}) h_1} \times \sum_{h_2} e^{(b_2 + \boldsymbol{x}^T \boldsymbol{w}_{*2}) h_2} \times \\
& \ldots \times \sum_{h_2} e^{(b_N + \boldsymbol{x}^T \boldsymbol{w}_{*N}) h_N} \Bigg) \\
=& \frac{1}{Z_{BB}} e^{\boldsymbol{x}^T \boldsymbol{a}} \prod_j^N \sum_{h_j} e^{(b_j + \boldsymbol{x}^T \boldsymbol{w}_{*j}) h_j} \\
=& \frac{1}{Z_{BB}} e^{\boldsymbol{x}^T \boldsymbol{a}} \prod_j^N (1 + e^{b_j + \boldsymbol{x}^T \boldsymbol{w}_{*j}}). && (45)
\end{aligned}
$$

A similar derivation yields the marginal probability of the hidden units

$$
p_{BB}(\boldsymbol{h}) = \frac{1}{Z_{BB}} e^{\boldsymbol{b}^T \boldsymbol{h}} \prod_i^M (1 + e^{a_i + \boldsymbol{w}_{i*}^T \boldsymbol{h}}). \tag{46}
$$

Conditional Probability Density Functions. We derive the probability of the hidden units given the visible units using Bayes' rule

$$
\begin{aligned}
p_{BB}(\boldsymbol{h}|\boldsymbol{x}) =& \frac{p_{BB}(\boldsymbol{x},\boldsymbol{h})}{p_{BB}(\boldsymbol{x})} \\
=& \frac{\frac{1}{Z_{BB}} e^{\boldsymbol{x}^T\boldsymbol{a}+\boldsymbol{b}^T\boldsymbol{h}+\boldsymbol{x}^T\boldsymbol{W}\boldsymbol{h}}}{\frac{1}{Z_{BB}} e^{\boldsymbol{x}^T\boldsymbol{a}} \prod_j^N (1+e^{b_j+\boldsymbol{x}^T\boldsymbol{w}_{*j}})} \\
=& \frac{e^{\boldsymbol{x}^T\boldsymbol{a}} e^{\sum_j^N (b_j+\boldsymbol{x}^T\boldsymbol{w}_{*j})h_j}}{e^{\boldsymbol{x}^T\boldsymbol{a}} \prod_j^N (1+e^{b_j+\boldsymbol{x}^T\boldsymbol{w}_{*j}})} \\
=& \prod_j^N \frac{e^{(b_j+\boldsymbol{x}^T\boldsymbol{w}_{*j})h_j}}{1+e^{b_j+\boldsymbol{x}^T\boldsymbol{w}_{*j}}} \\
=& \prod_j^N p_{BB}(h_j|\boldsymbol{x}). \qquad (47)
\end{aligned}
$$

From this we find the probability of a hidden unit being "on" or "off":

$$
\begin{aligned}
p_{BB}(h_j=1|\boldsymbol{x}) =& \frac{e^{(b_j+\boldsymbol{x}^T\boldsymbol{w}_{*j})h_j}}{1+e^{b_j+\boldsymbol{x}^T\boldsymbol{w}_{*j}}} \qquad (48) \\
=& \frac{e^{(b_j+\boldsymbol{x}^T\boldsymbol{w}_{*j})}}{1+e^{b_j+\boldsymbol{x}^T\boldsymbol{w}_{*j}}} \qquad (49) \\
=& \frac{1}{1+e^{-(b_j+\boldsymbol{x}^T\boldsymbol{w}_{*j})}}, \qquad (50)
\end{aligned}
$$

and

$$
p_{BB}(h_j=0|\boldsymbol{x}) = \frac{1}{1+e^{b_j+\boldsymbol{x}^T\boldsymbol{w}_{*j}}}. \qquad (51)
$$

Similarly we have that the conditional probability of the visible units given the hidden are

$$
\begin{aligned}
p_{BB}(\boldsymbol{x}|\boldsymbol{h}) =& \prod_i^M \frac{e^{(a_i+\boldsymbol{w}_{i*}^T\boldsymbol{h})x_i}}{1+e^{a_i+\boldsymbol{w}_{i*}^T\boldsymbol{h}}} \qquad (52) \\
=& \prod_i^M p_{BB}(x_i|\boldsymbol{h}). \qquad (53)
\end{aligned}
$$

$$
\begin{aligned}
p_{BB}(x_i=1|\boldsymbol{h}) =& \frac{1}{1+e^{-(a_i+\boldsymbol{w}_{i*}^T\boldsymbol{h})}} \qquad (54) \\
p_{BB}(x_i=0|\boldsymbol{h}) =& \frac{1}{1+e^{a_i+\boldsymbol{w}_{i*}^T\boldsymbol{h}}}. \qquad (55)
\end{aligned}
$$

Gaussian-Binary Restricted Boltzmann Machines. Inserting into the expression for $E_{RBM}(\boldsymbol{x}, \boldsymbol{h})$ in equation results in the energy

$$\begin{aligned} E_{GB}(\boldsymbol{x}, \boldsymbol{h}) =& \sum_{i}^{M} \frac{(x_i - a_i)^2}{2\sigma_i^2} - \sum_{j}^{N} b_j h_j - \sum_{ij}^{M,N} \frac{x_i w_{ij} h_j}{\sigma_i^2} \\ =& ||\frac{\boldsymbol{x} - \boldsymbol{a}}{2\boldsymbol{\sigma}}||^2 - \boldsymbol{b}^T \boldsymbol{h} - (\frac{\boldsymbol{x}}{\boldsymbol{\sigma}^2})^T \boldsymbol{W} \boldsymbol{h}. \end{aligned} \tag{56}$$

Joint Probability Density Function.

$$\begin{aligned} p_{GB}(\boldsymbol{x}, \boldsymbol{h}) =& \frac{1}{Z_{GB}} e^{-||\frac{\boldsymbol{x}-\boldsymbol{a}}{2\boldsymbol{\sigma}}||^2 + \boldsymbol{b}^T \boldsymbol{h} + (\frac{\boldsymbol{x}}{\boldsymbol{\sigma}^2})^T \boldsymbol{W} \boldsymbol{h}} \\ =& \frac{1}{Z_{GB}} e^{-\sum_i^M \frac{(x_i - a_i)^2}{2\sigma_i^2} + \sum_j^N b_j h_j + \sum_{ij}^{M,N} \frac{x_i w_{ij} h_j}{\sigma_i^2}} \\ =& \frac{1}{Z_{GB}} \prod_{ij}^{M,N} e^{-\frac{(x_i - a_i)^2}{2\sigma_i^2} + b_j h_j + \frac{x_i w_{ij} h_j}{\sigma_i^2}}, \end{aligned} \tag{57}$$

with the partition function given by

$$Z_{GB} = \int \sum_{\tilde{\boldsymbol{h}}}^{\tilde{\boldsymbol{H}}} e^{-||\frac{\tilde{\boldsymbol{x}}-\boldsymbol{a}}{2\boldsymbol{\sigma}}||^2 + \boldsymbol{b}^T \tilde{\boldsymbol{h}} + (\frac{\tilde{\boldsymbol{x}}}{\boldsymbol{\sigma}^2})^T \boldsymbol{W} \tilde{\boldsymbol{h}}} d\tilde{\boldsymbol{x}}. \tag{58}$$

Marginal Probability Density Functions. We proceed to find the marginal probability densitites of the Gaussian-binary RBM. We first marginalize over the binary hidden units to find $p_{GB}(\boldsymbol{x})$

$$\begin{aligned} p_{GB}(\boldsymbol{x}) =& \sum_{\tilde{\boldsymbol{h}}}^{\tilde{\boldsymbol{H}}} p_{GB}(\boldsymbol{x}, \tilde{\boldsymbol{h}}) \\ =& \frac{1}{Z_{GB}} \sum_{\tilde{\boldsymbol{h}}}^{\tilde{\boldsymbol{H}}} e^{-||\frac{\boldsymbol{x}-\boldsymbol{a}}{2\boldsymbol{\sigma}}||^2 + \boldsymbol{b}^T \tilde{\boldsymbol{h}} + (\frac{\boldsymbol{x}}{\boldsymbol{\sigma}^2})^T \boldsymbol{W} \tilde{\boldsymbol{h}}} \\ =& \frac{1}{Z_{GB}} e^{-||\frac{\boldsymbol{x}-\boldsymbol{a}}{2\boldsymbol{\sigma}}||^2} \prod_{j}^{N} (1 + e^{b_j + (\frac{\boldsymbol{x}}{\boldsymbol{\sigma}^2})^T \boldsymbol{w}_{*j}}). \end{aligned} \tag{59}$$

We next marginalize over the visible units. This is the first time we marginalize over continuous values. We rewrite the exponential factor dependent on $\boldsymbol{x}$ as a Gaussian function before we integrate in the last step.

$$
\begin{aligned}
p_{GB}(\boldsymbol{h}) =& \int p_{GB}(\tilde{\boldsymbol{x}}, \boldsymbol{h}) d\tilde{\boldsymbol{x}} \\
=& \frac{1}{Z_{GB}} \int e^{-||\frac{\tilde{\boldsymbol{x}}-\boldsymbol{a}}{2\boldsymbol{\sigma}}||^2 + \boldsymbol{b}^T\boldsymbol{h} + (\frac{\tilde{\boldsymbol{x}}}{\boldsymbol{\sigma}^2})^T \boldsymbol{W}\boldsymbol{h}} d\tilde{\boldsymbol{x}} \\
=& \frac{1}{Z_{GB}} e^{\boldsymbol{b}^T\boldsymbol{h}} \int \prod_i^M e^{-\frac{(\tilde{x}_i - a_i)^2}{2\sigma_i^2} + \frac{\tilde{x}_i \boldsymbol{w}_{i*}^T \boldsymbol{h}}{\sigma_i^2}} d\tilde{\boldsymbol{x}} \\
=& \frac{1}{Z_{GB}} e^{\boldsymbol{b}^T\boldsymbol{h}} \Bigg(\int e^{-\frac{(\tilde{x}_1 - a_1)^2}{2\sigma_1^2} + \frac{\tilde{x}_1 \boldsymbol{w}_{1*}^T \boldsymbol{h}}{\sigma_1^2}} d\tilde{x}_1 \\
& \times \int e^{-\frac{(\tilde{x}_2 - a_2)^2}{2\sigma_2^2} + \frac{\tilde{x}_2 \boldsymbol{w}_{2*}^T \boldsymbol{h}}{\sigma_2^2}} d\tilde{x}_2 \\
& \times ... \\
& \times \int e^{-\frac{(\tilde{x}_M - a_M)^2}{2\sigma_M^2} + \frac{\tilde{x}_M \boldsymbol{w}_{M*}^T \boldsymbol{h}}{\sigma_M^2}} d\tilde{x}_M \Bigg) \\
=& \frac{1}{Z_{GB}} e^{\boldsymbol{b}^T\boldsymbol{h}} \prod_i^M \int e^{-\frac{(\tilde{x}_i - a_i)^2 - 2\tilde{x}_i \boldsymbol{w}_{i*}^T \boldsymbol{h}}{2\sigma_i^2}} d\tilde{x}_i \\
=& \frac{1}{Z_{GB}} e^{\boldsymbol{b}^T\boldsymbol{h}} \prod_i^M \int e^{-\frac{\tilde{x}_i^2 - 2\tilde{x}_i(a_i + \tilde{x}_i \boldsymbol{w}_{i*}^T \boldsymbol{h}) + a_i^2}{2\sigma_i^2}} d\tilde{x}_i \\
=& \frac{1}{Z_{GB}} e^{\boldsymbol{b}^T\boldsymbol{h}} \prod_i^M \int e^{-\frac{\tilde{x}_i^2 - 2\tilde{x}_i(a_i + \boldsymbol{w}_{i*}^T \boldsymbol{h}) + (a_i + \boldsymbol{w}_{i*}^T \boldsymbol{h})^2 - (a_i + \boldsymbol{w}_{i*}^T \boldsymbol{h})^2 + a_i^2}{2\sigma_i^2}} d\tilde{x}_i \\
=& \frac{1}{Z_{GB}} e^{\boldsymbol{b}^T\boldsymbol{h}} \prod_i^M \int e^{-\frac{(\tilde{x}_i - (a_i + \boldsymbol{w}_{i*}^T \boldsymbol{h}))^2 - a_i^2 - 2a_i \boldsymbol{w}_{i*}^T \boldsymbol{h} - (\boldsymbol{w}_{i*}^T \boldsymbol{h})^2 + a_i^2}{2\sigma_i^2}} d\tilde{x}_i \\
=& \frac{1}{Z_{GB}} e^{\boldsymbol{b}^T\boldsymbol{h}} \prod_i^M e^{\frac{2a_i \boldsymbol{w}_{i*}^T \boldsymbol{h} + (\boldsymbol{w}_{i*}^T \boldsymbol{h})^2}{2\sigma_i^2}} \int e^{-\frac{(\tilde{x}_i - a_i - \boldsymbol{w}_{i*}^T \boldsymbol{h})^2}{2\sigma_i^2}} d\tilde{x}_i \\
=& \frac{1}{Z_{GB}} e^{\boldsymbol{b}^T\boldsymbol{h}} \prod_i^M \sqrt{2\pi\sigma_i^2} e^{\frac{2a_i \boldsymbol{w}_{i*}^T \boldsymbol{h} + (\boldsymbol{w}_{i*}^T \boldsymbol{h})^2}{2\sigma_i^2}}. \qquad (60)
\end{aligned}
$$

Conditional Probability Density Functions. We finish by deriving the conditional probabilities.

$$\begin{aligned} p_{GB}(\boldsymbol{h}|\boldsymbol{x}) =& \frac{p_{GB}(\boldsymbol{x},\boldsymbol{h})}{p_{GB}(\boldsymbol{x})} \\ =& \frac{\frac{1}{Z_{GB}} e^{-||\frac{\boldsymbol{x}-\boldsymbol{a}}{2\boldsymbol{\sigma}}||^2 + \boldsymbol{b}^T\boldsymbol{h} + (\frac{\boldsymbol{x}}{\boldsymbol{\sigma}^2})^T \boldsymbol{W}\boldsymbol{h}}}{\frac{1}{Z_{GB}} e^{-||\frac{\boldsymbol{x}-\boldsymbol{a}}{2\boldsymbol{\sigma}}||^2} \prod_j^N (1 + e^{b_j + (\frac{\boldsymbol{x}}{\boldsymbol{\sigma}^2})^T \boldsymbol{w}_{*j}})} \\ =& \prod_j^N \frac{e^{(b_j + (\frac{\boldsymbol{x}}{\boldsymbol{\sigma}^2})^T \boldsymbol{w}_{*j}) h_j}}{1 + e^{b_j + (\frac{\boldsymbol{x}}{\boldsymbol{\sigma}^2})^T \boldsymbol{w}_{*j}}} \\ =& \prod_j^N p_{GB}(h_j|\boldsymbol{x}). \end{aligned} \tag{61}$$

The conditional probability of a binary hidden unit h_j being on or off again takes the form of a sigmoid function

$$\begin{aligned} p_{GB}(h_j = 1|\boldsymbol{x}) =& \frac{e^{b_j + (\frac{\boldsymbol{x}}{\boldsymbol{\sigma}^2})^T \boldsymbol{w}_{*j}}}{1 + e^{b_j + (\frac{\boldsymbol{x}}{\boldsymbol{\sigma}^2})^T \boldsymbol{w}_{*j}}} \\ =& \frac{1}{1 + e^{-b_j - (\frac{\boldsymbol{x}}{\boldsymbol{\sigma}^2})^T \boldsymbol{w}_{*j}}} \end{aligned} \tag{62}$$

$$p_{GB}(h_j = 0|\boldsymbol{x}) = \frac{1}{1 + e^{b_j + (\frac{\boldsymbol{x}}{\boldsymbol{\sigma}^2})^T \boldsymbol{w}_{*j}}}. \tag{63}$$

The conditional probability of the continuous $\boldsymbol{x}$ now has another form, however.

$$
\begin{aligned}
p_{GB}(\boldsymbol{x}|\boldsymbol{h}) =& \frac{p_{GB}(\boldsymbol{x},\boldsymbol{h})}{p_{GB}(\boldsymbol{h})} \\
=& \frac{\frac{1}{Z_{GB}} e^{-||\frac{\boldsymbol{x}-\boldsymbol{a}}{2\boldsymbol{\sigma}}||^2 + \boldsymbol{b}^T\boldsymbol{h} + (\frac{\boldsymbol{x}}{\boldsymbol{\sigma}^2})^T \boldsymbol{W}\boldsymbol{h}}}{\frac{1}{Z_{GB}} e^{\boldsymbol{b}^T\boldsymbol{h}} \prod_i^M \sqrt{2\pi\sigma_i^2} e^{\frac{2a_i \boldsymbol{w}_{i*}^T\boldsymbol{h} + (\boldsymbol{w}_{i*}^T\boldsymbol{h})^2}{2\sigma_i^2}}} \\
=& \prod_i^M \frac{1}{\sqrt{2\pi\sigma_i^2}} \frac{e^{-\frac{(x_i-a_i)^2}{2\sigma_i^2} + \frac{x_i \boldsymbol{w}_{i*}^T\boldsymbol{h}}{2\sigma_i^2}}}{e^{\frac{2a_i \boldsymbol{w}_{i*}^T\boldsymbol{h} + (\boldsymbol{w}_{i*}^T\boldsymbol{h})^2}{2\sigma_i^2}}} \\
=& \prod_i^M \frac{1}{\sqrt{2\pi\sigma_i^2}} \frac{e^{-\frac{x_i^2 - 2a_i x_i + a_i^2 - 2x_i \boldsymbol{w}_{i*}^T\boldsymbol{h}}{2\sigma_i^2}}}{e^{\frac{2a_i \boldsymbol{w}_{i*}^T\boldsymbol{h} + (\boldsymbol{w}_{i*}^T\boldsymbol{h})^2}{2\sigma_i^2}}} \\
=& \prod_i^M \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{x_i^2 - 2a_i x_i + a_i^2 - 2x_i \boldsymbol{w}_{i*}^T\boldsymbol{h} + 2a_i \boldsymbol{w}_{i*}^T\boldsymbol{h} + (\boldsymbol{w}_{i*}^T\boldsymbol{h})^2}{2\sigma_i^2}} \\
=& \prod_i^M \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(x_i - b_i - \boldsymbol{w}_{i*}^T\boldsymbol{h})^2}{2\sigma_i^2}} \\
=& \prod_i^M \mathcal{N}(x_i|b_i + \boldsymbol{w}_{i*}^T\boldsymbol{h}, \sigma_i^2) \qquad (64) \\
\Rightarrow p_{GB}(x_i|\boldsymbol{h}) =& \mathcal{N}(x_i|b_i + \boldsymbol{w}_{i*}^T\boldsymbol{h}, \sigma_i^2). \qquad (65)
\end{aligned}
$$

The form of these conditional probabilities explains the name "Gaussian" and the form of the Gaussian-binary energy function. We see that the conditional probability of x_i given $\boldsymbol{h}$ is a normal distribution with mean $b_i + \boldsymbol{w}_{i*}^T\boldsymbol{h}$ and variance σ_i^2.