Some notes about BB- and GB-RBM implementation

Tomas Ukkonen, 2015 tomas.ukkonen@iki.fi

Restricted Boltzmann machines are tricky to implement as there doesn't seem to be very good reference material around describing all the details of the implementation. Here are some notes about RBM implementation in `..src/neuralnetwork/RBM.h` and `..src/neuralnetwork/RBM.cpp`. The learning method to be used is CD-10.

I couldn't find update rules for bias-terms used to calculate approximate gradient descent for (contrastive divergence) so I used the following approach:

- there are no separate bias-terms
- instead visible $\boldsymbol{v}$ and hidden $\boldsymbol{h}$ states are always *extended* to have one extra 1 after the data terms, so the actual terms given to the RBM are $[\boldsymbol{v}, 1]$ and $[\boldsymbol{h}, 1]$

This then makes the calculations relatively simple as you will only have to care about the weight matrix $\boldsymbol{W}$ and can ignore other terms. How to handle these equal to 1 states during the stimulation phase of RBM in CD? The correct approach seems to handle them similarly to all other variables, in other words we just simulate and simulate their states normally. This then leads to situation where the bias terms automatically find rules that always set those values to 1.

TOY PROBLEM ANALYSIS

The function `rbm_test()` in `src/neuralnetwork/tst/test.cpp` generates a toy problem for which RBM CD-10 algorithm is tested. The data is generated as follows:

- hidden states (2) are selected randomly between 0 and 1
- after this visible states (4) are generated from hidden states by directly setting the k:th state exactly same as the $k/2$:th state in a hidden state
- in practice $[0, 1] \Rightarrow [0, 0, 1, 1]$,$[1, 0] \Rightarrow [1, 1, 0, 0]$, $[1, 1] \Rightarrow [1, 1, 1, 1]$ and $[0, 0] \Rightarrow [0, 0, 0, 0]$
- so the relationship between hidden states and input states is really simple, the easiest way is to simply to calculate mean value of every 2 items in visible layer and used it as a hidden layer

The results of for running CD-10 algorithm which discretizes or samples all values to 0 and 1 are always the same, the weight matrix for hidden to visible layers have the form:

$$\boldsymbol{W} = \begin{array}{|c|c|c|c|c|} \hline 4.6 & 4.4 & 4.3 & 4.5 & 5.6 \\ \hline 0.0 & -0.1 & -9.7 & -9.8 & 8.9 \\ \hline -9.7 & -9.7 & 0.09 & 0.2 & 8.7 \\ \hline \end{array}$$

This weight matrix clearly generates three hidden states where the first one is equal to the state 1 which is always on. Because binary values only take values 0 and 1 the sigmoid function only gets positive values (the last term is 5.6 meaning that the all 1 term in input layer sets value to be extremely likely to be 1 even when other terms are 0). After this, the second term sets the term to be 0 when the second part of the inputs are 1 and 0 when the inputs are 1 because in this case the sigmoidal value will be negative. And similarly for the last term.

Given this intepretation, the reversed weight matrix $\boldsymbol{W}^T$ makes now also sense:

$$\boldsymbol{W}^T = \begin{array}{|c|c|c|} \hline 4.6 & +0.1 & -9.7 \\ \hline 4.4 & -0.1 & -9.7 \\ \hline 4.3 & -9.7 & 0.0 \\ \hline 4.5 & -9.7 & 0.2 \\ \hline 5.6 & 8.9 & 8.7 \\ \hline \end{array}$$

If we keep in mind that the first term is always 1 in our reversed. Then the first two terms are 1 only when the third term is zero and zero only when the third term is one (negative sigmoid value). Similar is true for the second set of variables and finally the 5th variable is always one due to fact that state values are always non-negative meaning that sigmoidal value will be always be positive.

This then proofs and shows that the toy problem is properly solved by **the binary RBM with the CD-10** *learning rule. It can find proper two states and the one artificial always one state using the simple contrastive divergence algorithm.* This shows RBM is capable of data reduction.

Energy based model of RBM

The continuous RBM is more complicated to implement and understand. It seems that Gaussian-Bernoulli RBM could be the right choice (or maybe Beta-Bernoulli RBM) to model continuous input values. Because of this, I try to derive the whole BB-RBM and then continuous GB-RBM theory from energy based models (EBMs). It is important(?) to remember that Bernoulli-Bernoulli RBM model is related to Ising models and statistical physics but hidden variable models in general are not.

The energy of the restricted boltzman machine is:

$E_{BB}(v,h) = -a^T v - v^T W h - b^T h$

$E_{GB}(v,h) = \frac{1}{2}\|v-a\|^2 - v^T W h - b^T h$

Note that we could have added additional constant c term to these equations but it is not needed because probability distributions are normalized to have unit probability mass.

Now the probability of the (v,h) and only observed variables v is:

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

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

Now the hidden variables are Bernoulli distributed, that is, they take only values 0 and 1 which is a strong regularizer for the system.

We want to calculate probabilities of hidden h and visible v neurons:

$P_{BB}(h|v) = \frac{P(v,h)}{P(v)} = \frac{\frac{1}{Z} e^{-E(v,h)}}{\frac{1}{Z}\sum_h e^{-E(v,h)}} = \frac{e^{v^T W h + a^T v + b^T h}}{\sum_h e^{v^T W h + a^T v + b^T h}}$

$P_{BB}(h_i = 1|v) = \frac{e^{+v^T w_i 1 + a^T v + b_i 1}}{e^{+v^T w_i 0 + a^T v + b_i 0} + e^{+v^T w_i 1 + a^T v + b_i 1}} = \frac{e^{+v^T w_i + b_i}}{1 + e^{+v^T w_i + b_i}} = \frac{1}{1 + e^{-v^T w_i - b_i}} = \text{sigmoid}(w_i^T v + b_i)$

$P_{BB}(\boldsymbol{h} = \mathbf{1}|\boldsymbol{v}) = \text{sigmoid}(\boldsymbol{W}\boldsymbol{v} + \boldsymbol{b})$

$P_{BB}(v_i = 1|h) = \frac{e^{+1 w_i^{(T)} h + a_i 1 + b^T h}}{e^{+0 w_i^{(T)} h + a_i 0 + b^T h} + e^{+1 w_i^{(T)} h + a_i 1 + b^T h}} = \frac{1}{1 + e^{-w_i^{(T)} h - a_i}} = \text{sigmoid}(w_i^{(T)} h + a_i)$

$P_{BB}(\boldsymbol{v} = \mathbf{1}|\boldsymbol{h}) = \text{sigmoid}(\boldsymbol{W}^T\boldsymbol{h} + \boldsymbol{a})$

Free Energy

Next we use the definition of free energy and calculate its derivates on parameters W, a and b. Free energy is defined to be:

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

And the related probability distribution of data (visible states) is

$P(v) = \frac{1}{Z} e^{-F(v)} = \frac{1}{Z}\sum_h e^{-E(v,h)}$, $Z = \int e^{-F(v)}\, d\, v$

Now we want to calculate gradient with respect to parameters of the distribution in order to maximize likelihood of the data $P(v)$:

$$\frac{\partial - \log p(v)}{\partial \theta} = -\frac{\partial p(v)/\partial \theta}{p(v)} = -\frac{p(v)(-\partial F(v)/\partial \theta) - p(v)(\partial Z/\partial \theta)/Z}{p(v)} = \frac{\partial F(v)}{\partial \theta} + \frac{1}{Z}\frac{\partial Z}{\partial \theta}$$

We calculate the term $\frac{1}{Z}\frac{\partial Z}{\partial \theta}$ separatedly:

$$\frac{1}{Z}\frac{\partial Z}{\partial \theta} = \int \; -\frac{\partial F(v)}{\partial \theta}\frac{1}{Z}e^{-F(v)}d\,v = -\int \; p(v)\,\frac{\partial F(v)}{\partial \theta}\,d\,v$$

The general form of the derivate is then:

$$\frac{\partial \log p(v)}{\partial \theta} = -\Big(\frac{\partial F(v)}{\partial \theta} - \int \; p(v)\,\frac{\partial F(v)}{\partial \theta}\,d\,v\Big) = -\Big(\frac{\partial F(v)}{\partial \theta} - E_v[\frac{\partial F(v)}{\partial \theta}]\Big)$$

And the latter term can be approximated using contrastive divergence algorithm. We use the training data to produce $p(h|v)$ and then sample $p(v|h)$ and repeat the procedure to get sample (v_i, h_i) and only keep v_i to get the approximate sample from distribution $p(v)$. This *maybe* special case of Gibbs sampling. Altenatively, it is possible to use MCMC, AIS, PT-MCMC or some other sampling technique to sample directly from $P(v)$ instead using CD-k Gibbs sampling.

Gradient descent

Parameters of the distribution are optimized using gradient descent algorithm (maximum likelihood method) so it is important to calculate actual derivates of $p(v)$ for Bernoulli-Bernoulli RBM. (Also note that gradients are needed by approximative second order methods like L-BFGS).

First we further simplify the $F(v)$ terms

$$F(v) = -\log \sum_h \; e^{-E_{BB}(v,h)} = -a^T v - \log \sum_h \; e^{(W^T v + b)^T h} = -a^T v - \log \sum_h \; e^{\sum_{i,j} h_i (v_j w_{ij} + b_i)}$$

$$F(v) = -a^T v - \log \sum_h \; \prod_i \; e^{h_i (\sum_j v_j w_{ij} + b_i)} = -a^T v - \sum_i \; \log \sum_{h_i} \; e^{h_i (\sum_j v_j w_{ij} + b_i)}$$

If we decide $h = \{0, 1\}$ then the equation simplifies further into

$$F(v) = -a^T v - \sum_i \; \log\,(1 + e^{\sum_j v_j w_{ij} + b_i})$$

Calculating gradients leads into eqs:

$$\frac{\partial F(v)}{\partial a} = -v$$

$$\frac{\partial F(v)}{\partial b_i} = -\frac{e^{\sum_j v_j w_{ij} + b_i}}{1 + e^{\sum_j v_j w_{ij} + b_i}} = -\text{sigmoid}(\textstyle\sum_j \; v_j\, w_{ij} + b_i)$$

$$\frac{\partial\, F(v)}{\partial\, w_{ij}} = -\frac{e^{\sum_j v_j w_{ij} + b_i}\, v_j}{1 + e^{\sum_j v_j w_{ij} + b_i}} = -v_j\, \text{sigmoid}(\textstyle\sum_j \; v_j\, w_{ij} + b_i)$$

Gaussian distribution (Gaussian-Bernoulli RBM) - Continuous RBM

So far we have only discussed about Bernoulli-Bernoulli RBM. But what we really want is to process continuous valued input data (and then maybe use BB-RBM to further process its hidden variables). The possible models to use are Gaussian-Bernoulli and Beta-Bernoulli (*note that Beta and Bernoulli distributions are conjugate distributions, this might be useful..*). It seems that gaussian distribution is far more popular so I try to calculate Gaussian-Bernoulli RBM instead.

If we ignore (or fix it to unit value) the variance term of normal distribution, the logical energy function seem to be:

$$E_{GB}(v, h) = \frac{1}{2}\|v - a\|^2 - v^T W h - b^T h$$

To further justify this model, let's calculate marginalized distributions $p(v|h)$ and $p(h|v)$ for this model.

$$P(v|h) = \frac{P(v,h)}{P(h)} = \frac{e^{-E_{GB}(v,h)}}{\int_v e^{-E_{GB}(v,h)} d\,v} = \frac{e^{v^T W h - \frac{1}{2}\|v-a\|^2}}{\int_v e^{v^T W h - \frac{1}{2}\|v-a\|^2} d\,v}$$

$$\int_v e^{v^T W h - \frac{1}{2}\|v-a\|^2 + b^T h} d\,v = e^{b^T h} \int e^{v^T W h - \frac{1}{2}\|v-a\|^2} d\,v = \int e^{-\frac{1}{2}\|v\|^2 + v^T(Wh+a) - \frac{1}{2}\|a\|^2} d\,v$$

$$\int e^{-\frac{1}{2}\|v\|^2 + v^T(Wh+a) - \frac{1}{2}\|a\|^2} d\,v = e^{\frac{1}{2}\|Wh+a\|^2 - \frac{1}{2}\|a\|^2} \int e^{-\frac{1}{2}\|v-(Wh+a)\|^2} d\,v$$

$$P(v|h) = \frac{e^{\frac{1}{2}\|Wh+a\|^2 - \frac{1}{2}\|a\|^2} e^{-\frac{1}{2}\|v-(Wh+a)\|^2}}{e^{\frac{1}{2}\|Wh+a\|^2 - \frac{1}{2}\|a\|^2} \int e^{-\frac{1}{2}\|v-(Wh+a)\|^2} d\,v} = \frac{1}{Z} e^{-\frac{1}{2}\|v-(Wh+a)\|^2} \sim \text{Normal}(Wh+a, I)$$

And similar calculations can be done to calculate: $P(h|v) = \text{sigmoid}(v^T W + b)$.

The related free energy model is:

$$F(v) = -\log \sum_h e^{-E_{GB}(v,h)} = -\log \sum_h e^{-\frac{1}{2}\|v-a\|^2 + v^T W h + b^T h} = \frac{1}{2}\|v-a\|^2 - \log \sum_h e^{(W^T v + b)^T h}$$

And the related gradient of normal distribution parameter a is:

$$\frac{\partial F}{\partial a} = a - v$$

And the other gradients should be the same as in the Bernoulli-Bernoulli model.

Non-unit variance

Unit variance assumption of Gaussian-Bernoulli RBM can be a problem when the input data is not properly preprocessed (in practice we assume $\text{Var}[f(x) + e] = \text{Var}[f(x)] + N(0, 1)$). For example, maximum variance of Bernoulli distributed hidden variables is 0.25 which means that the noise in input data should be modelled to be much smaller.

NOTE: We should add a inverse-Wishart prior for Σ term assuming it to be diagnonal I initially (a regularizer). After this we can normalize variance of each dimension to have unit variance meaning that prior is hopefully close to noise prior and integrate over the $p(\Sigma)$ prior.

The energy model for the non-unit variance that can make sense is:

$$E_{GB}(v,h) = \frac{1}{2}(v-a)^T \Sigma^{-1}(v-a) - (\Sigma^{-0.5} v)^T W h - b^T h - \log\left(|\Sigma|^{-(v+p+1)/2}\right) + \frac{1}{2}\text{tr}(\Sigma^{-1})$$

which can be further justified by calculating $p(v|h)$ and $p(h|v)$ distributions:

$$P(v|h) = \frac{P(v,h)}{P(h)} = \frac{e^{-E_{GB}(v,h)}}{\int_v e^{-E_{GB}(v,h)} d\,v} = \frac{e^{(\Sigma^{-0.5} v)^T W h - \frac{1}{2}(v-a)^T \Sigma^{-1}(v-a)}}{\int_v e^{(\Sigma^{-0.5} v)^T W h - \frac{1}{2}(v-a)^T \Sigma^{-1}(v-a)} d\,v}$$

$$\int_v e^{(\Sigma^{-0.5} v)^T W h - \frac{1}{2}(v-a)^T \Sigma^{-1}(v-a)} d\,v = \int e^{-\frac{1}{2} v^T \Sigma^{-1} v + v^T \Sigma^{-1}(\Sigma^{0.5} W h + a) - \frac{1}{2}\|a\|^2} d\,v$$

$$\int e^{-\frac{1}{2} v^T \Sigma^{-1} v + v^T \Sigma^{-1}(\Sigma^{0.5} W h + a) - \frac{1}{2}\|a\|^2} d\,v = e^{\frac{1}{2}\|\Sigma^{-0.5} W h + a\|_\Sigma^2 - \frac{1}{2}\|a\|^2} \int e^{-\frac{1}{2}\|v-(\Sigma^{0.5} W h + a)\|_\Sigma^2} d\,v$$

$$P(v|h) = \frac{1}{Z(\Sigma)} e^{-\frac{1}{2}\|v-(\Sigma^{0.5} W h + a)\|_\Sigma^2} \sim \text{Normal}(v|\Sigma^{1/2} W h + a, \Sigma)$$

$$P(\boldsymbol{v}|\boldsymbol{h}, \Sigma) \sim \text{Normal}\left(v \middle| \boldsymbol{\Sigma}^{1/2} \boldsymbol{W h} + \boldsymbol{a}, \boldsymbol{\Sigma}\right)$$

To generate normally distributed variables with wanted covariance matrix we notice that for $x \sim N(0, I)$ data we have:

$$\Sigma_x = \text{Cov}[A\,x] = E[A\,x\,x^T A^T] = A\,I\,A^T = A\,A^T = X \Lambda\, X^T \Rightarrow A = X\,\Lambda^{1/2}$$

And similarly we calculate

$$P(h|v) = \frac{P(v,h)}{P(v)} = \frac{e^{-E_{GB}(v,h)}}{\sum_h e^{-E_{GB}(v,h)}} = \frac{e^{(\Sigma^{-0.5}v)^T W h + b^T h}}{\sum_h e^{(\Sigma^{-0.5}v)^T W h + b^T h}} = \frac{e^{(\Sigma^{-0.5}v)^T W h + b^T h}}{\sum_h e^{(\Sigma^{-0.5}v)^T W h + b^T h}}$$

The term can be further rewritten as:

$$e^{(\Sigma^{-0.5}v)^T W h + b^T h} = e^{\sum_d ((\Sigma^{-0.5}v)^T w_d + b_d) h_d} = \prod_d e^{((\Sigma^{-0.5}v)^T w_d + b_d) h_d}$$

And if we only restrict to a single variable we have

$$P(h_d|v) = \frac{e^{((\Sigma^{-0.5}v)^T w_d + b_d) h_d}}{\sum_{h_d} e^{((\Sigma^{-0.5}v)^T w_d + b_d) h_d}} = \text{sigmoid}((\Sigma^{-0.5}\, v)^T w_d + b_d).$$

$$P(\boldsymbol{h}|\boldsymbol{v}, \Sigma) = \text{sigmoid}((\boldsymbol{\Sigma}^{-0.5}\, \boldsymbol{v})^T \boldsymbol{W} + \boldsymbol{b}\,)$$

Free Energy gradient of Σ

Next we want to calculate gradient to Σ which can be very tricky. In the continuous RBM paper I managed to find, the authors didn't use matrix form and only calculated gradient of σ and then *approximated* it. This result further shows that calculation of gradent of full covariance matrix Σ in this case is probably analytically extremely difficult or impossible.

Initially, we notice that Σ is a symmetric matrix. This mean that if we do a change of basis we can always find a space where Σ is a diagonal matrix, $\boldsymbol{\Sigma = \text{diag}(\sigma_1 ... \sigma_D)}$. Additionally, the RBM model implicitly assumes that variables are independent when given hidden or visible variables. Therefore, to continue to make this independence assumption (note: RBM can be seen as somewhat similar to ICA where we estimate "independent" or "semi-independent" components from the data), we assume there is no correlations between elements of the visible units given hidden units.

$$F(v) = -\log \sum_h e^{-E_{GB}(v,h)} = -\log \sum_h e^{-\frac{1}{2}\|v-a\|^2_\Sigma + (\Sigma^{-0.5}v)^T W h + b^T h + \log\left(|\Sigma|^{-(v+p+1)/2}\right) - \frac{1}{2}\text{tr}(\Sigma^{-1})}$$

$$= \frac{1}{2}(v-a)^T \Sigma^{-1}(v-a) - \log\left(|\Sigma|^{-(v+p+1)/2}\right) + \frac{1}{2}\text{tr}(\Sigma^{-1}) - \log \sum_h e^{(W^T \Sigma^{-0.5} v + b)^T h}$$

$$= \frac{1}{2}(v-a)^T \Sigma^{-1}(v-a) - \log\left(|\Sigma|^{-(v+p+1)/2}\right) + \frac{1}{2}\text{tr}(\Sigma^{-1}) - \log \prod_i \sum_{h_i} e^{(W^T \Sigma^{-0.5} v + b)_i h_i}$$

$$= \frac{1}{2}(v-a)^T \Sigma^{-1}(v-a) + ((v+p+1)/2)\log(|\Sigma|) + \frac{1}{2}\text{tr}(\Sigma^{-1}) - \sum_i \log\left(1 + e^{(W^T \Sigma^{-0.5} v + b)_i}\right)$$

Its derivate, assuming the covariance matrix is diagonal, is:

$$\frac{\partial F}{\partial \sigma_k} = -\frac{(v_k - a_k)^2}{\sigma_k^3} + (v+p+1)\sum_i \frac{\partial}{\partial \sigma_k}\log(\sigma_i) - \frac{1}{\sigma_k^3} - \sum_i \frac{e^{(W^T \Sigma^{-0.5} v + b)_i}}{1 + e^{(W^T \Sigma^{-0.5} v + b)_i}} \frac{\partial}{\partial \sigma_k}\left(\sum_k w_{ki} \frac{v_k}{\sigma_k}\right)$$

$$= -\frac{(v_k - a_k)^2}{\sigma_k^3} + (v+p+1)\frac{1}{\sigma_k} - \frac{1}{\sigma_k^3} + v_k/\sigma_k^2 \sum_i w_{ki}\, \text{sigmoid}(W^T \Sigma^{-0.5} v + b)_i$$

$$\text{diag}\left[\frac{\partial F}{\partial \Sigma}\right] =$$

$$\text{diag}\left[-\Sigma^{-3/2}((v-a)\,(v-a)^T + I) + (v+p+1)\Sigma^{-1/2} + (\Sigma^{-1} v)\,(W\, \text{sigmoid}(W^T \Sigma^{-0.5} v + b))^T\right]$$

Other derivates of the free energy are:

$$\frac{\partial F}{\partial a} = -\Sigma^{-1}(v-a)$$

$$\frac{\partial F(v)}{\partial b_i} = -\text{sigmoid}\left(\sum_j w_{ij} v_j / \sigma_j + b_i\right),\ \frac{\partial F}{\partial b} = -\text{sigmoid}(W^T \Sigma^{-1/2} v + b)$$

$$\frac{\partial F(v)}{\partial w_{ij}} = -\frac{v_j}{\sigma_j}\,\text{sigmoid}\left(\sum_j w_{ij}\, v_j / \sigma_j + b_i\right), \frac{\partial F}{\partial W} = -\text{sigmoid}(W^T \Sigma^{-1/2} v + b)\,(\Sigma^{-1/2}\, v)^T$$

Now the open question is whether these derivates are valid for ***any*** covariance matrix Σ as it would mean that we could then estimate correlations between visible layer inputs and not expect them to be statistically independent given hidden variables.

RESULTS: This does not seem to work for any Σ matrix, but it is uncertain as the primary problem here is that variances can become negative meaning that the method is not robust or a complex method is needed to dynamically adjust learning rates so that Σ matrix (eigenvalues) can never become negative.

It is possible to get this working but it seems that the convergence is really slow. The primary idea is to check if updated covariance matrix has positive diagonal and is otherwise good looking (non complex numbers, NaNs, Infs etc) and always drop the learning rate by 50% if it seems that the update would make covariance matrix "unhealthy". But this is not robust or a good solution as it "filters out" updates that would turn matrix into bad one. Additionally, the update rule seem to be really SLOW here. (`gbrbm-variance-model-v1`)

The direct variance update don't work, we instead try to fix the situation with the change of variable $e^{-z_i} = 1/\sigma_i^2$ (as recommended below), leading into the equation:

$$F(v) = \tfrac{1}{2}(v-a)^T\Sigma^{-1}(v-a) + ((v+p+1)/2)\log(|\Sigma|) + \tfrac{1}{2}\mathrm{tr}(\Sigma^{-1}) - \textstyle\sum_i \log(1 + e^{(W^T\Sigma^{-0.5}v + b)_i})$$

$$= \tfrac{1}{2}\textstyle\sum_i (1 + (v_i - a_i)^2)e^{-z_i} + ((v+p+1)/2)\sum_i z_i - \sum_j \log\left(1 + \exp\left(\sum_k w_{kj} v_k e^{-z_k/2} + b_j\right)\right)$$

$$\frac{\partial F}{\partial z_i} = -\tfrac{1}{2}\left(1 + (v_i - a_i)^2\right)e^{-z_i} + ((v+p+1)/2) - \textstyle\sum_j \mathrm{sigmoid}(W^T\Sigma^{-0.5}v + b)_j\left(-\tfrac{1}{2}w_{ij}v_i e^{-z_i/2}\right)$$

$$\frac{\partial F}{\partial z_i} = -\tfrac{1}{2}e^{-z_i}\left(1 + (v_i - a_i)^2\right) + ((2\dim(v)+1)/2) + \tfrac{1}{2}e^{-z_i/2}v_i\textstyle\sum_j w_{ij}\,\mathrm{sigmoid}(W^T\Sigma^{-0.5}v + b)_j$$

Wishart prior's degrees of freedom is chosen to be low so we choose the minimum $v = p$ so that mean of inverse wishart prior/regularizer so degrees of freedom is minimized and variance is maximized (maximum unceretainty).

This works somewhat better but **still** leads into NaNs and other problems when the larger number of hidden nodes are used. (This may be because of small covariance matrixes leading into eqs near Infinities.)

ALTERNATIVE GAUSSIAN MODEL

Because their basic approach to introduce variances into model seem to cause problems I now try another slightly modified model introduced by Aalto university researchers (Cho et al. 2011).

Now the energy of the system is defined to be

$$E_{GB}(v,h)$$

$$= \tfrac{1}{2}\alpha\textstyle\sum_i (v_i - a_i)^2/\sigma_i^2 - \sum_{i,j}\frac{v_i}{\sigma_i^2}w_{ij}h_j - \sum_i b_i h_i$$

$$= \tfrac{1}{2}(v-a)(\alpha D^{-2})(v-a) - v^T D^{-2} W h - b^T h$$

$$= \textstyle\sum_i \frac{1}{2}v_i^2\,\alpha/\sigma_i^2 + \frac{1}{2}a_i^2\,\alpha/\sigma_i^2 - \sum_i (a_i + \sum_j w_{ij}h_j/\alpha)v_i\,\alpha/\sigma_i^2 - \sum_i b_i h_i$$

$$= \tfrac{1}{2}\textstyle\sum_i (v_i - (a_i + \sum_j w_{ij}h_j/\alpha))^2\alpha/\sigma_i^2 + \frac{1}{2}\sum_i a_i^2\alpha/\sigma_i^2 - \frac{1}{2}\sum_i (a_i + \sum_j w_{ij}h_j/\alpha)^2\,\alpha/\sigma_i^2 - \sum_i b_i h_i$$

$$= \tfrac{1}{2}(v - (a + Wh/\alpha))^T(\alpha\, D^{-2})(v - (a + Wh/\alpha)) + \tfrac{1}{2}a^T(\alpha\, D^{-2})a - \tfrac{1}{2}\|a + Wh/\alpha\|^2_{(D^2/\alpha)} - b^T h$$

After this it is again quite straightforward to calculate the conditional probabilities given hidden or visible neurons.

$$P(v|h) = \frac{e^{-\frac{1}{2}\sum_i (v_i - (a_i + \sum_j w_{ij} h_j/\alpha))^2 \alpha/\sigma_i^2}}{\int e^{-\frac{1}{2}\sum_i (v_i - (a_i + \sum_j w_{ij} h_j/\alpha))^2 \alpha/\sigma_i^2 + \dots} dv} \propto e^{-\frac{1}{2}\sum_i (v_i - (a_i + \sum_j w_{ij} h_j/\alpha))^2 \alpha/\sigma_i^2}$$

$$\sim \prod_i N(a_i + \sum_j w_{ij} h_j/\alpha, \sigma_i^2/\alpha)$$

$$\boldsymbol{P(v|h) \sim \text{Normal}(a + Wh/\alpha, D^2/\alpha),\ D^2 = \text{diag}[\sigma_1^2 ... \sigma_D^2]}$$

$$P(h_j|v) = \frac{e^{\sum_{i,j} \frac{v_i}{\sigma_i^2} w_{ij} h_j + \sum_j b_j h_j}}{\sum_{h_j} e^{\sum_{i,j} \frac{v_i}{\sigma_i^2} w_{ij} h_j + \sum_j b_j h_j}} = \frac{e^{(\sum_i \frac{v_i}{\sigma_i^2} w_{ij} + b_j) h_j}}{\sum_{h_j} e^{(\sum_i \frac{v_i}{\sigma_i^2} w_{ij} + b_j) h_j}} = \text{sigmoid}(\sum_i \frac{v_i}{\sigma_i^2} w_{ij} + b_j)$$

$$\boldsymbol{P(h|v) = \text{sigmoid}(b + v^T D^{-2} W)}$$

Notice here that this time the mean is not related to variances which can be useful.

After calculating the conditional probabilities, we again calculate free energy of the energy function (and then calculate its partial derivates).

$$F(v) = -\log \sum_h e^{-E_{GB}(v,h)} = -\log \sum_h e^{-\frac{1}{2}\alpha (v-a)^T D^{-2} (v-a) + v^T D^{-2} W h + b^T h}$$

$$= \frac{1}{2}\alpha (v-a)^T D^{-2} (v-a) - \sum_i \log \sum_{h_i} e^{(W^T D^{-2} v + b)_i h_i}$$

$$= \frac{1}{2}\alpha (v-a)^T D^{-2} (v-a) - \sum_i \log (1 + e^{(W^T D^{-2} v + b)_i})$$

$$= \frac{1}{2}\sum_i \alpha \frac{(v_i - a_i)^2}{\sigma_i^2} - \sum_i \log (1 + \exp (\sum_j w_{ji} v_j/\sigma_j^2 + b_i))$$

From this formulation we can calculate

$$\frac{\partial F}{\partial a} = -\alpha D^{-2} (a - v)$$

$$\frac{\partial F}{\partial b_i} = -\frac{e^{(W^T D^{-2} v + b)_i}}{1 + e^{(W^T D^{-2} v + b)_i}} = -\text{sigmoid}(W^T D^{-2} v + b)_i$$

$$\frac{\partial F}{\partial b} = -\text{sigmoid}(W^T D^{-2} v + b) = -h$$

$$\frac{\partial F}{\partial w_{ij}} = -\frac{e^{(W^T D^{-2} v + b)_i}}{1 + e^{(W^T D^{-2} v + b)_i}} (v_j/\sigma_j^2) = -\frac{v_j}{\sigma_j^2} \text{sigmoid}(W^T D^{-2} v + b)_i = -D^{-2} v\, h^T$$

But the crucial derivate is the derivate of σ_i terms, we alter energy function by the change of terms: $1/\sigma_i^2 = e^{-z_i}$ leading into formula:

$$F(v) = \frac{1}{2}\sum_i \alpha (v_i - a_i)^2 e^{-z_i} - \sum_j \log (1 + \exp (\sum_i w_{ij} v_i e^{-z_i} + b_j))$$

And then derivate with respect to z_i:

$$\frac{\partial F}{\partial z_i} = -\frac{1}{2}\alpha (v_i - a_i)^2 e^{-z_i} + \sum_j \frac{\exp (\sum_i w_{ij} v_i e^{-z_i} + b_j)}{1 + \exp (\sum_i w_{ij} v_i e^{-z_i} + b_j)} w_{ij} v_i e^{-z_i}$$

$$= -e^{-z_i} [\frac{1}{2}\alpha (v_i - a_i)^2 - v_i \sum_j w_{ij} \text{sigmoid}(W^T D^{-2} v + b)_j] = e^{-z_i} S(a, b, D)$$

But because variance is difficult parameter to optimize for, we want to calculate its Hessian matrix and use second order derivation

This is the exactly same formula as the one given in the paper of Aalto uni people (*Improved Learning of Gaussian-Bernoulli Restricted Boltzmann Machines. ICANN 2011.*)

Important: The analysis of results seem to imply to following result. Aalto university people model performance with data having multiple modes (circle data etc) have **bad performance** if gradient descent method is used and variance should be learned - cannot learn the variance (Parallel Tempering MCMC may be possible to learn the model variance correctly). However, it has **BETTER** performance if correct variance is known and gradient descent method is used.

1. This seem to imply to following approach: first optimize "original" GB-RBM so that you can learn the variances (correctly)

2. After learning the variances switch to "Aalto-university" GB-RBM model which can give better reconstruction error. With correct variances known it is now possible to learn multiple modes of the distribution easily.

3. Alternatively try to use more sophisticated methods.

Parallel Tempering Annihilated Importance Sampling (AIS)

Parallel tempering is a powerful approach that was used by Aalto university people in 2011 (also see *On the Quantitative Analysis of Deep Belief Networks. Salakhutdinov.* about AIS). I will try to use same method to generate intermediate distributions from which to estimate partition function Z which is needed to estimate exact probabilities.

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

We generate series of temperatures $\beta = [0...1]$ and define intermediate models as (*Cho et al. 2011*):

$$\boldsymbol{W}^{(\beta)} = \beta\,\boldsymbol{W}\ ,\ a_i = \beta\,a_i + (1-\beta)\,m_i$$

$$\boldsymbol{b}^{(\beta)} = \beta\,\boldsymbol{b},\ \sigma_i^{(\beta)} = \sqrt{\beta\,\sigma_i^2 + (1-\beta)\,s_i^2}$$

So with $\beta = 0$ we simply sample from unscaled normal probability distribution $v_i \sim \mathcal{N}(m_i,\, s_i^2)$ and then define **stochastic** transition operation as

$$T_\beta(v', v) = p_\beta(v'|h)\,p_\beta(h|v)$$

Because we know (**proof?**) that iterative Gibbs sampling $T(v', v)$ from distribution will lead to distribution $p(v)$ given any distribution $p(v)$ for any v (assuming all $p(v) > 0$) then the transition from probability distribution $p(v)$ of values will lead to limiting distribution of $p(v)$ of values?? (Convergence point of markov chain).

After this we use AIS to estimate Z-ratios (which mean is used as an estimate) using un-scaled distributions by using samples:

$$\frac{Z_{\beta_K}}{Z_{\beta_0}} = \frac{p'_{\beta_1}(\boldsymbol{v}_1)}{p'_{\beta_0}(\boldsymbol{v}_1)}\frac{p'_{\beta_2}(\boldsymbol{v}_2)}{p'_{\beta_1}(\boldsymbol{v}_2)}\cdots\frac{p'_{\beta_K}(\boldsymbol{v}_K)}{p'_{\beta_{k-1}}(\boldsymbol{v}_K)}\,.$$

(NOTE: is $\boldsymbol{v}_K$ generated this way also distributed as $p_K(\boldsymbol{v})$?)

But because we know $p_{\beta_0}(\boldsymbol{v})$ to be exactly normal it is trival to sample from and its normalizing constant Z can be computed directly.

$$Z_{\beta_0} = (2\,\pi)^{D/2}\,\sigma_1...\sigma_D$$

What is still needed to estimate Z is the calculation of un-scaled probability distribution values $p'(\boldsymbol{v})$ given parameters of GB-RBM. This needs summation over all values of $\boldsymbol{h}$:

$$P'(\boldsymbol{v}) = e^{-\frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{v}-\boldsymbol{a})} \sum_{\boldsymbol{h}} e^{\boldsymbol{v}^T\boldsymbol{\Sigma}^{-0.5}\boldsymbol{W}h + \boldsymbol{b}^T\boldsymbol{h}}$$

For $\boldsymbol{h}$ this simplifies further into summation:

$$S = \sum_{\boldsymbol{h}} e^{\boldsymbol{\alpha}^T\boldsymbol{h}} = \sum_{\boldsymbol{h}} \prod_i e^{\alpha_i h_i}$$

The next comes the tricky/smart step, because $\boldsymbol{h}$ takes only values 0 and 1 this summation can be simplified without going through all the $2^{\dim(\boldsymbol{h})}$ possible values by using property of multiplication $(\alpha+\beta)(\varsigma+\delta)$.... "generates"/"goes through" all possible 2^D states **in linear time!**

$$S = \sum_{\boldsymbol{h}} \prod_i e^{\alpha_i h_i} = \prod_i \left(e^{\alpha_i(h_i=0)} + e^{\alpha_i(h_i=1)}\right) = \prod_i (1 + e^{\alpha_i})$$

Therefore, the unscaled log probability is:

$$\log(P'(\boldsymbol{v})) = -\frac{1}{2}(\boldsymbol{v}-\boldsymbol{a})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{v}-\boldsymbol{a}) + \sum_i \log\left(1 + e^{\alpha_i(\boldsymbol{v},\boldsymbol{W},\boldsymbol{\Sigma},\boldsymbol{b})}\right),$$
$$\boldsymbol{\alpha} = \boldsymbol{W}^T\boldsymbol{\Sigma}^{-0.5}\,\boldsymbol{v} + \boldsymbol{b}.$$

About numerical stability: in practice, the exponent of alpha can become too large. If e^{α_i} becomes floating point infinity and destroys numerical accuracy, we use approximation

$$\log(1 + e^{\alpha}) \approx \log(e^{\alpha}) = \alpha$$

which is good approaximation if e^{α} is near infinity (even at floating point accuracy).

Building Neural Network from Stacked RBM

After the problem of learning GB-RBM and BB-RBM has been solved it is naturally interesting to try to use it to construct *feedforward neural network*. This is rather straight forward as the probability functions always has sigmoidal probability function (even at the top layer) and we can directly insert weights and biases from the RBM. The only problem is that our RBM is a stochastic/probabilistic meaning that we will now make significant error when approximating hidden neurons to have continuos values.

TODO

Hamiltonian Monte Carlo sampler based approach

Because the direct optimization method doesn't seem to work very well. It seems that an interesting approach could be try to use Monte Carlo sampling as seen in many papers as the given probability model fits into HMC approach. For hamiltonian we will use function

$$H(\boldsymbol{q},\boldsymbol{p}) = E_T(\boldsymbol{q}) + K(\boldsymbol{q}),\ K(\boldsymbol{p}) = \boldsymbol{p}^T\boldsymbol{M}^{-1}\boldsymbol{p}/2$$

And we want to use HMC to sample from distribution:

$$p(\boldsymbol{\theta}|\boldsymbol{v}) = \frac{1}{Z(\boldsymbol{v})}\exp(-E_T(\boldsymbol{\theta}|\boldsymbol{v})) \propto p(\boldsymbol{v}|\boldsymbol{\theta})\,p(\boldsymbol{\theta})$$

But in practice we have only know the data likelihood $p(\boldsymbol{v}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})}e^{-F(\boldsymbol{v}|\boldsymbol{\theta})}$ term from the previous sections. This will lead to the following eqs (for the gradient):

$$E_T(\boldsymbol{\theta}|\boldsymbol{v}) = -\log\left(\frac{Z(\boldsymbol{v})}{Z_T(\boldsymbol{\theta})}\right) + F_T(\boldsymbol{v}|\boldsymbol{\theta}) - \log\left(\frac{p(\boldsymbol{\theta})}{p(\boldsymbol{v})}\right)$$

$$\nabla_{\boldsymbol{\theta}}E_T(\boldsymbol{\theta}|\boldsymbol{v}) = \nabla_{\boldsymbol{\theta}}F_T(\boldsymbol{v}|\boldsymbol{\theta}) - E_{\boldsymbol{v}}[\nabla_{\boldsymbol{\theta}}F_T(\boldsymbol{v}|\boldsymbol{\theta})] - \nabla_{\boldsymbol{\theta}}\log(p(\boldsymbol{\theta}))$$

We can decide for the flat priors meaning that $p(\boldsymbol{\theta}) \propto 1$ and the last term will be zero. This means that we can use directly gradient computed for the $-\log(P(\boldsymbol{v}))$. But HMC also calculates difference between energy functions

$$E_T(\boldsymbol{\theta}_{n+1}|\boldsymbol{v}) - E_T(\boldsymbol{\theta}_n|\boldsymbol{v}) = F_T(\boldsymbol{v}|\boldsymbol{\theta}_{n+1}) - F_T(\boldsymbol{v}|\boldsymbol{\theta}_n) + \log\left(\frac{Z_T(\boldsymbol{\theta}_{n+1})}{Z_T(\boldsymbol{\theta}_n)}\right) - \log\left(\frac{p(\boldsymbol{\theta}_{n+1})}{p(\boldsymbol{\theta}_n)}\right)$$

The last term can be assumed to be zero but the problem is ratio of partition functions. We know that ratio of partition functions will be close to 1 if the change between parameters is not too large so it can be initially approximated to be zero but in practice this seem to lead to too small variances causing problems. But we can approximate by using the previous results (by using first order derivate approximation from $\boldsymbol{\theta}_{n+1}$ to $\boldsymbol{\theta}_n$ and from $\boldsymbol{\theta}_n$ to $\boldsymbol{\theta}_{n+1}$ and calculating the mean):

$$\log\left(\frac{Z_T(\boldsymbol{\theta}_{n+1})}{Z_T(\boldsymbol{\theta}_n)}\right) \approx -\frac{1}{2}(\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n)^T \left(E_{\boldsymbol{v}}[\nabla_{\boldsymbol{\theta}} F_T(\boldsymbol{v}|\boldsymbol{\theta}_{n+1})] + E_{\boldsymbol{v}}[\nabla_{\boldsymbol{\theta}} F_T(\boldsymbol{v}|\boldsymbol{\theta}_n)]\right)$$

This approximation does not seem to work well in practice and cannot be used.

You can calculate gradient quite easily but you cannot calculate difference between energy functions as it requires estimating the ratio of partition functions. For this case I will try to use *importance sampling* (as the distributions should be close to each other and we want only the ratio and not the actual Z_T-values. Theory:

$$\frac{Z_T(\boldsymbol{\theta}_{n+1})}{Z_T(\boldsymbol{\theta}_n)} = \int \frac{p'_T(\boldsymbol{v}|\boldsymbol{\theta}_{n+1})}{Z_T(\boldsymbol{\theta}_n)} d\boldsymbol{v} = \int \frac{p'_T(\boldsymbol{v}|\boldsymbol{\theta}_{n+1})}{p'_n(\boldsymbol{v}|\boldsymbol{\theta}_n)} p(\boldsymbol{v}|\boldsymbol{\theta}_n) d\boldsymbol{v} = E_{\boldsymbol{\theta}_n}\left[\frac{p'_T(\boldsymbol{v}|\boldsymbol{\theta}_{n+1})}{p'_n(\boldsymbol{v}|\boldsymbol{\theta}_n)}\right]$$

So we need to sample $\boldsymbol{v}$:s from the distribution using AIS and "parallel tempering" approach described by Cho et. al. in 2011. This seem to work rather well after which we calculate ratios of unscaled distributions $p'(\boldsymbol{v}|\boldsymbol{\theta}) = e^{-F(\boldsymbol{v}|\boldsymbol{\theta})}$ and calculate the mean value:

$$E_{\boldsymbol{v}|\boldsymbol{\theta}_n}\left[\frac{p'_T(\boldsymbol{v}|\boldsymbol{\theta}_{n+1})}{p'_n(\boldsymbol{v}|\boldsymbol{\theta}_n)}\right] = E_{\boldsymbol{v}|\boldsymbol{\theta}_n}\left[e^{F(\boldsymbol{v}|\boldsymbol{\theta}_n) - F(\boldsymbol{v}|\boldsymbol{\theta}_{n+1})}\right].$$

This is easy to parallelize and seem to work with enough accuracy, **in low dimensions** (<256). TODO: Probably a good way to estimate $Z_T(\boldsymbol{\theta}_n)$ would be to use variational methods (variatioanl bayes but currently it is too complicated for me).

Latent variable model

In general, the RBM method works by first defining probability function $p(\boldsymbol{v},\ \boldsymbol{h}|\boldsymbol{\theta})$ and then maximizing visible states (visible data) probability

$$\max_{\boldsymbol{\theta}} p(\boldsymbol{v}|\boldsymbol{\theta}) = \int\ p(\boldsymbol{v}, \boldsymbol{h}|\boldsymbol{\theta})\, d\boldsymbol{h}$$

But instead of calculating maximum likelihood solution, one could also calculate bayesian samples and estimate distribution or sample from

$$p(\boldsymbol{\theta}|\boldsymbol{v}) \propto p(\boldsymbol{v}|\boldsymbol{\theta})\, p(\boldsymbol{\theta})$$

But sampling from this distribution can be difficult but one can maybe use MCMC sampling which is only interested in relative probabilities

$$r = \frac{p(\boldsymbol{\theta}_{n+1}|\boldsymbol{v})\, p(\boldsymbol{\theta})}{p(\boldsymbol{\theta}_n|\boldsymbol{v})\, p(\boldsymbol{\theta})} = \frac{\sum_{\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}|\boldsymbol{\theta}_{n+1})}{\sum_{\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}|\boldsymbol{\theta}_n)}$$

If we furthermore assume that $Z(\boldsymbol{\theta}_{n+1}) \approx Z(\boldsymbol{\theta}_n)$ then we get a formula:

$$r(v) \approx \frac{\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h}|\boldsymbol{\theta}_{n+1})}}{\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h}|\boldsymbol{\theta}_n)}} \approx$$

After which hidden states are sampled or calculated using a formula

$$p(\boldsymbol{h}|\boldsymbol{v}) = \int\ p(\boldsymbol{h}|\boldsymbol{\theta}, \boldsymbol{v})\, p(\boldsymbol{\theta}|\boldsymbol{v})\, d\boldsymbol{\theta} \approx \frac{1}{N}\sum_i p(\boldsymbol{h}|\boldsymbol{\theta}_i, \boldsymbol{v})$$

Now for multivariate probability distribution $\boldsymbol{x} = [\boldsymbol{v}, \boldsymbol{h}]^T$, $p(\boldsymbol{x}) \sim N(\boldsymbol{\mu_x}, \boldsymbol{\Sigma_x})$ it is trivially easy to compute $p(\boldsymbol{v}|\boldsymbol{\mu_v},\ \boldsymbol{\Sigma_v})$ as the solution is yet another normal distribution. But now because marginal distribution do not have dependency to $\boldsymbol{\mu_h}$ or $\boldsymbol{\Sigma_h}$ or other variables given known variable, it is not possible to calculate relatioship between $\boldsymbol{h}$ and $\mathbf{v}$.

If one does not want to use bayesian approach when maximizing probability of visible states given parameters, another approach could be still use gradient descent but now giving the gradient to L-BFGS optimizer method (conjugate gradient) method which should then be able to do higher quality maximization of visible states probability by using only gradient information.

Solving variance model with bayesian prior distribution

We add inverse scaled chi-squared distribution as prior/regularizer distribution for σ_i^2 terms in order to keep variance as low as possible

$E_{GB}(v,h) + \text{ScaledInv}\chi^2(\boldsymbol{\sigma}^2)$

$= \frac{1}{2}\sum_i (v_i - a_i)^2/\sigma_i^2 - \sum_{i,j} \frac{v_i}{\sigma_i^2} w_{ij} h_j - \sum_i b_i h_i + \frac{1}{2}(\alpha - 1)\sum_i (v_i - a_i)^2/\sigma_i^2$

$= \frac{1}{2}\alpha\,(v - (a + Wh))^T D^{-2}(v - (a + Wh)) + \frac{1}{2}a^T D^{-2} a - \frac{1}{2}\|a + Wh\|^2 - b^T h$

After this It is possible to calculate free energy derivate of σ_i^2.

$F(v) = -\log \sum_h e^{-E_{GB}(v,h)} = -\log \sum_h e^{-\frac{1}{2}(v-a)^T D^{-2}(v-a) - \alpha\frac{1}{2}s^T D s + v^T D^{-2} W h + b^T h}$

$= \frac{1}{2}(v-a)^T D^{-2}(v-a) + \frac{1}{2}s^T D^2 s - \sum_i \log \sum_{h_i} e^{(W^T D^{-2} v + b)_i h_i}$

$= \frac{1}{2}(v-a)^T D^{-2}(v-a) + \frac{1}{2}s^T D^{-2} s - \sum_i \log\,(1 + e^{(W^T D^{-2} v + b)_i})$

$= \frac{1}{2}\sum_i \frac{(v_i - a_i)^2}{\sigma_i^2} + \frac{1}{2}\sum_i s_i^2 \sigma_i^2 - \sum_i \log\,(1 + \exp\,(\sum_j w_{ji} v_j/\sigma_j^2 + b_i))$

And the derivate with respect to z_i is:

$\frac{\partial F}{\partial z_i} = -e^{-z_i}[\frac{1}{2}(v_i - a_i)^2 - v_i \sum_j w_{ij} \text{sigmoid}(W^T D^{-2} v + b)_j] + \frac{1}{2}s_i^2 e^{z_i}$

And s_i is a sample square statistic $s_i^2 = \sum_{j=1}^N (v_i(j) - \bar{v_i}(j))^2$ where j:s are indexes for individual observations from the data. Now from the data it is easy to calculate s_i^2 but from the model this is a different thing. If we approximate $s_i^2 = \alpha\,(v_i - a_i)^2$ we get regularized gradients.

Multiple visible elements

In real world applications, it is often interesting to learn distribution $p(\boldsymbol{y}|\boldsymbol{x})$ and in RBM context this means we have multiple different visible elements $\boldsymbol{v} = [\boldsymbol{v}_1, \boldsymbol{v}_2]$ and we need to be able to sample from $p(\boldsymbol{v}_2|\boldsymbol{v}_1)$ in order to predict value of $\boldsymbol{v}_2$ given elements $\boldsymbol{v}_1$. In this model, the learning of RBM elements can be done normally. However, in reconstruction phase we need to sample from $p(\boldsymbol{h}|\boldsymbol{v}_1)$ and then from $p(\boldsymbol{v}_2|\boldsymbol{h})$ in order to approximate sample from integral

$$p(\boldsymbol{v}_2|\boldsymbol{v}_1) = \int\ p(\boldsymbol{v}_2|\boldsymbol{h})\,p(\boldsymbol{h}|\boldsymbol{v}_1)\,d\boldsymbol{h}$$

In GB-RBM we need that $p(\boldsymbol{v}|\boldsymbol{h})$ has a normal distribution and calculating conditional normal distribution $p(\boldsymbol{v}_1|\boldsymbol{h})$ and $p(\boldsymbol{v}_2|\boldsymbol{h})$ is "easy":

$$p(\boldsymbol{v}|\boldsymbol{h}) \sim \text{Normal}\big(\boldsymbol{v}\big|\boldsymbol{\Sigma}^{1/2}\boldsymbol{W}\boldsymbol{h} + \boldsymbol{a}, \boldsymbol{\Sigma}\big), \boldsymbol{\Sigma} = \text{diag}([\sigma_1 ... \sigma_D])$$

$$p(\boldsymbol{v}_2|\boldsymbol{v}_1 = \boldsymbol{x}, \boldsymbol{h}) \sim \text{Normal}\big(\boldsymbol{v}_2|\boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{21}\,\boldsymbol{\Sigma}_{11}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_1), \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\,\boldsymbol{\Sigma}_{11}^{-1}\,\boldsymbol{\Sigma}_{12}\big)$$

But because $\boldsymbol{\Sigma}$ is diagonal conditional distribution does not directly depend on values of $\boldsymbol{v}_1$.

$$p(\boldsymbol{v}_1|\boldsymbol{h}) \sim \text{Normal}(\boldsymbol{v}_1|\boldsymbol{\mu}_1(\boldsymbol{h}), \boldsymbol{\Sigma}_{11})$$
$$p(\boldsymbol{v}_2|\boldsymbol{h}) \sim \text{Normal}(\boldsymbol{v}_2|\boldsymbol{\mu}_2(\boldsymbol{h}), \boldsymbol{\Sigma}_{22})$$

What is left is to calculate $p(\boldsymbol{h}|\boldsymbol{v}_1)$ to generate hidden variables from input values $\boldsymbol{v}_1 = \boldsymbol{x}$, which is more difficult.

$$p(\boldsymbol{h}|\boldsymbol{v}) = \text{sigmoid}((\boldsymbol{\Sigma}^{-0.5}\,\boldsymbol{v})^T\boldsymbol{W} + \boldsymbol{b}\,)$$

By generating $\boldsymbol{v}_2$:s randomly we can generate candidate(s) $\boldsymbol{h}_0$ which can be then improved and we can use bayes rule to write

$$p(\boldsymbol{h}|\boldsymbol{v}_1) \propto p(\boldsymbol{v}_1|\boldsymbol{h})\, p(\boldsymbol{h})$$

And if we assume each configuration is is equally likely in our model (without any data hidden states should have equal probability of being either 0 or 1 $\Rightarrow$ so the prior should maybe still have some effect - it should force values of $\boldsymbol{h}$ to be either 0 or 1 thought the values should be equally likely), then the prior $p(\boldsymbol{h})$ is flat constant. This makes sampling from $p(\boldsymbol{h}|\boldsymbol{v}_1)$ using HMC rather easy. Our $U(\boldsymbol{h})$ terms are ($\boldsymbol{\Gamma}_1$-matrix is simple used to select $\boldsymbol{\mu}_1$ from $\boldsymbol{\mu}$):

$$U(\boldsymbol{h}) = \tfrac{1}{2}(\boldsymbol{v}_1 - \boldsymbol{\mu}_1(\boldsymbol{h}))^T\boldsymbol{\Sigma}_{11}^{-1}(\boldsymbol{v}_1 - \boldsymbol{\mu}_1(\boldsymbol{h})),\ \boldsymbol{\mu}_1(\boldsymbol{h}) = \boldsymbol{\Gamma}_1\big(\boldsymbol{\Sigma}^{1/2}\,\boldsymbol{W}\boldsymbol{h} + \boldsymbol{a}\big)$$
$$\nabla_{\boldsymbol{h}}U(\boldsymbol{h}) = (\boldsymbol{\mu}_1(\boldsymbol{h}) - \boldsymbol{v}_1)^T\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Gamma}_1\,\boldsymbol{\Sigma}^{1/2}\,\boldsymbol{W}$$

Here we have assumed $\boldsymbol{h}$ can take continuous values between 0 and 1 in practice, the sampling should maybe happen in continuous $\boldsymbol{h}$-space but the exact samples generated should be discretized. Alternatives:

- keep $\boldsymbol{h}$ values discretized to 0, 1 values always. Now there is problem that with too small epsilon the sampler stays in a same state and cannot never try to change state
- keep values of $\boldsymbol{h}$ continuous internally and only discretize to 0, 1 states when emitting a sample
- use continuous $\boldsymbol{h}$ values, everywhere, this is not well-defined as everything in our theory assumes samples must be discretized and other sources recommend against completely continuous variables..

Another approach would generate candidate $\boldsymbol{h}_0$ states and then do gradient descent until convergence to the best state and discretize $\boldsymbol{h}$. After discretization calculate gradient descent again until convegence and discretize again $\boldsymbol{h}$. Stop until there is no changes in $\boldsymbol{h}$. Repeat N times. This will be maximum data likelihood estimate of $p(\boldsymbol{h}|\boldsymbol{v}_1)$. After this sample $\boldsymbol{p}(\boldsymbol{v}_1|\boldsymbol{h})$.

Brute force (modular) approach

Assume we can divide $\boldsymbol{h}$ into parts $[\boldsymbol{h}_1 ... \boldsymbol{h}_L]$ and go through all the combinations separatedly. If we have perfect parallel machine this happens in $O\big(2^{\max_i \dim(\boldsymbol{h}_i)}\big)$ time or in $O\big(2^{\dim(\boldsymbol{h})/K}\big)$ on average if we can divide problem perfectly into K parts. We can choose, for example, $\dim(\boldsymbol{h}_i) = 10$ and the computations scale now linearly (although the results are not equally good).

General multiple elements (modular) approach

In practice, the memory requirements $O(\dim(\boldsymbol{v}) \times \dim(\boldsymbol{h}))$ and computation of huge RBM's grow too quickly if we use perfect model and use full matrix $\boldsymbol{W}$. Because of this we consider modular approach (also suggested by neuroscience research) where vectors are divided into modules/subvectors. This leads into following energy equations:

$$E_{BB}(\boldsymbol{v}_1 ... \boldsymbol{v}_K, \boldsymbol{h}_1 ... \boldsymbol{h}_L) = -\boldsymbol{a}^T\boldsymbol{v} - \textstyle\sum_{k,l} \boldsymbol{v}_k^T\boldsymbol{W}_{kl}\,\boldsymbol{h}_l - \boldsymbol{b}^T\boldsymbol{h}$$
$$E_{GB}(\boldsymbol{v}_1 ... \boldsymbol{v}_K, \boldsymbol{h}_1 ... \boldsymbol{h}_L) = \tfrac{1}{2}\|\boldsymbol{v} - \boldsymbol{a}\|^2 - \textstyle\sum_{k,l} \boldsymbol{v}_k^T\boldsymbol{W}_{kl}\,\boldsymbol{h}_l - \boldsymbol{b}^T\boldsymbol{h}$$

This reduces amount of memory required during computations but doesn't lead into other improvements. But we want that $p(\boldsymbol{h}) = p(\boldsymbol{h}_1)p(\boldsymbol{h}_2)...p(\boldsymbol{h}_L)$ in order to be able to brute force through $\boldsymbol{h}$ when looking for optimum because now we can look for optimum for each $\boldsymbol{h}_i$ vector separatedly.