Les Cahiers du GERAD ISSN: 0711–2440

Pattern-based stochastic simulation using Restricted Boltzmann Machine

Y. Mu, F. Ferrie,
R. Dimitrakopoulos

G–2016–107

November 2016

Pattern-based stochastic simulation using Restricted Boltzmann Machine

Yanyan Mu [a]

Frank Ferrie [a]

Roussos Dimitrakopoulos [a]

[a] *GERAD & COSMO Stochastic Mine Planning Laboratory, Department of Mining and Materials Engineering, McGill University, FDA Building, 3450 University Street, Montreal, Quebec, H3A 2A7, Canada*

yanyanmu@cim.mcgill.ca
ferrie@cim.mcgill.ca
roussos.dimitrakopoulos@mcgill.ca

November 2016

Les Cahiers du GERAD

G–2016–107

Abstract: For pattern-based simulation methods such as SIMPAT, filtersim, wavesim, ect, patterns are stored by scanning a training image with a sliding template. Dimensional reduction and pattern clustering are then preformed to generate a set of cluster prototypes. During simulation, the conditional data is matched to the best cluster by subsequent similarity measurements and a pattern is randomly sampled from that class. In this paper, a multiple-point, pattern-based, stochastic simulation algorithm using Restricted Boltzmann Machine (RBMSim) is proposed. In the learning phase of RBMSim, the clusters are learned by a contractive divergence learning algorithm which aggregates the two steps of dimensional reduction and pattern clustering. In simulation phase of RBMSim, the two steps of classification and sampling are also aggregated with an efficient Gibbs sampling algorithm. The contributions of this paper are: the learning and simulation are embedded within a mathematically well-defined probabilistic model which represents the entire configuration space; and the learning and simulation phases share the same Markov chain, Monte Carlo based sampling algorithm which explores the entire configuration space efficiently.

Keywords: Pattern based simulation, generative model, Restricted Boltzmann Machine, Gibbs sampling

1 Introduction

Training images are widely used in spatial stochastic simulation methods based on multiple points (Wu et al., 2008) or high-order statistics (Mustapha and Dimitrakopoulous, 2010) as they act as an explicit prior of the geological structures.

Pattern-based simulation algorithms extract multiple-point statistics from the training pattern database to build a probabilistic model to mathematically represent the geological structure over the multiple-points as a joint distribution over the configuration space. Given hard data, realizations can be generated from the model by sampling these joint distributions. For example, given a pattern database $v\{img\}$ which contains patches taken from training images, the multiple-point statistics can be defined as the joint distribution $p(v\{img\}; \emptyset)$ which is learned from $v\{img\}$, where $\emptyset$ are the parameters of the probabilistic model. Given hard data $v\{hard\}$, the missing pixels in the current simulating pattern can be drawn by sampling from the conditional distribution $p(v\{missin\ g\}|v\{hard\}; \emptyset)$. In the extreme case, where no hard data exists in the pattern, simulation could be generated by drawing samples from the joint distribution $p(v\{img\}; \emptyset)$ unconditionally.

The early development of pattern-based simulation is based on searching. These methods create a search tree that predicts the missing value based on input hard data. In these methods, the conditional pdf $p(v\{missin\ g\}|v\{hard\})$ is modeled directly. The Snesim algorithm (Strebelle, 2002; Strebelle and Cavelius, 2014) builds a search tree to store all the patterns from the training pattern database. The algorithm is improved by implementing a compact search tree (Zhang et al., 2012) or a list (Straubhaar et al., 2011) to increase the search efficiency and memory utilization. When the exact reproduction doesn't exist in the pattern database, some points in the hard data are discarded.

Another early development in the field of multiple point simulation is based on similarity measure, such as SIMPAT (Arpat and Caers, 2005). This algorithm uses a pre-defined proximity transform distance function to find the best pattern within the training database given the conditioning data event. Unlike SNESIM, exact replicates are not required and conditioning data is never ignored, but the search will span the entire configuration space. An improved version of this distance-based method is presented in Arpat and Caers, 2007 which increases the search efficiency by limiting the search space. The pattern configuration space is reduced to a lower dimension then partitioned into several clusters.

The state-of-the-art pattern-based simulation methods are feature-based, such as FilterSim (Wu et al., 2008) and WaveSim (Chatterjee et al., 2012). The two algorithms are very similar but use different sub-algorithms at each step. They both map the pattern configuration space to a lower dimensional feature space by filtering the patterns with pre-defined features or wavelets and then partition this subspace into several clusters. The conditioning data event is then classified to the best matched cluster by some similarity measure (distance functions) between the conditioning data event and the cluster prototypes. Finally, a pattern is randomly sampled from that class.

These feature-based methods have the following disadvantages: the pre-defined feature (filters or wavelets) extractors may not yield the maximum entropy coding solution of the training data and the joint distribution over the configuration space of the multiple points is a centroid based "multi-modal" distribution. As a result, the sampling during the simulation is bounded within the single best match class rather than the entire configuration space which reduces the stochasticity of the simulation. Future more, the algorithm suffers from a lack of mathematical formulation. As long as the pattern database is given and a random path is defined, WaveSim runs the following steps sequentially: 1) dimensional reduction by wavelet decomposition; 2) clustering the Wavelet coefficients by K-means; 3) classification of hard data by minimum distance; 4) drawing of a realization by sampling the conditional distribution in the selected cluster. The first two steps are implemented once whereas the last two steps are repeated for each position along the random path. Each step in the algorithm implements different sub-algorithms independent from each other. The first step is working in the pattern configuration space and the second step is working in the wavelet coefficient space. Pattern-based stochastic simulation using Restricted Boltzmann Machine (RBMSim) proposed in this paper simultaneously incorporates these steps in one model, which work in the same configuration space and share the same sampling algorithm. This allows for consistency between learning and simulating and a computationally efficient framework.

Restricted Boltzmann Machine (RBM), is a mathematically well-defined probabilistic graphical model applied to model the joint distribution $p(v\{img\}, \emptyset)$ over the configuration space by learning directly from training pattern database. The parameter $\emptyset$ in RBM contains a weight matrix and two bias vectors. Previous graphical models (Colin, 2004) use Markov Random Fields (MRF) and Gibbs sampling, however, their MRF don't include hidden layers which limit the performance of the simulation.

Similar to the WaveSim algorithm, the RBMSim contains two phases: the learning phase and the generating phase. The learning phase is applied once for each simulation task; the generating phase is repeated for each position along the random path. In the learning phase, the features are learned from the training pattern database; this ensures that these learned features are the most-likely descriptors for reconstructing the training patterns. Also, the peaks of $p(v\{img\}, \emptyset)$ correspond to the center of clusters, so explicit pattern clustering is not necessary. The learning result $p(v\{img\}, \emptyset)$ is a continuous distribution in the pattern configuration space rather than an isolated centroid based "multi-modal" distribution (e.g. the prototypes in WaveSim). In the generating phase, the realizations can be generated efficiently with or without the conditioning data. The Gibbs sampling will 'push' the input conditioning data along the learned probability $p(v\{img\}, \emptyset)$ to the more probable configuration. Finally, the sampling process has the possibility to explore the entire configuration space rather than sampling within a single cluster ensuring better stochasticity. RBMSim integrates multiple steps of stochastic simulation jointly to one unified algorithm. The steps of this algorithm are as follows: 1) learn the features from the geological structures; 2) model the patterns by the learned features; 3) reduce the dimensionality of the patterns using a non-linear mapping from Cartesian space to hidden space; 4) apply pattern classifications to the modeled patterns. RBMSim borrows concepts from both HOSIM (Mustapha and Dimitrakopoulous, 2010) and WAVESIM in that it learns a continuous distribution in the pattern configuration space and also simulates multiple points simultaneously.

This paper is organized as follows. Section 2 briefly reviews RBM learning algorithms. Section 3.1 describes RBM as a density model with the mathematical formulation of the RBM learning algorithm in Section 3.2. Section 3.3 defines the link between the learning and simulating algorithms. The learned features, conditional and unconditional simulations are shown in Section 4. The sensitivity of the pattern size is presented in Section 5. Section 6 compares the mathematical formulation of RBM with the conventional algorithm of multiple point simulation followed by conclusions and future work.

2 Short review of RBM learning algorithms

Restricted Boltzmann Machines (RBM) are undirected graphical models. Like all undirected models, the log-likelihood gradient of a RBM contains two terms: expectation with respect to the data distribution (positive phase) and expectation with respect to the model distribution (negative phase). The second term is intractable; the following algorithms apply different approximations.

RBM learning algorithms are based on variations of alternating Gibbs sampling. The most popular one is the Contrastive Divergence (CD) algorithm (Hinton, 2002). To do the learning, a Gibbs chain is initialized from a sample of the training data and alternating the Gibbs chain over several iterations, the last sample from the last iteration of the chain is used to approximate the intractable model expectation. The Persistent Contrastive Divergence (PCD) algorithm (Tijmen, 2008) also uses Gibbs sampling to approximate the model expectation. However, PCD doesn't reset to a training data after updating the weights, the state of the chain 'persists' from the previous iteration. By simply running a few iterations of the chain, these changes would be sufficient to track the samples from the 'updated' model distribution. The Fast Persistent Contrastive Divergence (FPCD) algorithm (Tieleman and Hinton, 2009) aims to achieve faster mixing of the Gibbs chain. It maintains a separate set of parameters that work with a higher learning rate for sampling only. This learning simply pushes the chain out of the local mode. Another popular method to train RBM is the Parallel Tempering (PT) algorithm (Desjardins et al., 2010) which uses an advanced technique; Replica Monte Carlo. It runs several chains in parallel under different temperatures rather than a single Gibbs chain. As the temperature increases, the Gibbs distribution becomes smoother and the mixing of corresponding chain becomes simple and direct.

In general, PT (Desjardins et al., 2010; Brakel et al., 2012) yields a better approximation with less bias but requires a heavier computational cost than CD (Hinton, 2002).

3 RBM: Model and learning algorithm

Notation and terminology	
v_i	The activation of visible units.
h_j	The activation of the hidden units.
b_i, b_j	The bias terms associated with the visible and hidden units respectively.
w_{ij}	The weights of connections between the visible and hidden units.
$E(v,h)$	The joint energy of the network for the configuration of visible and hidden units.
$p(v,h)$	The joint distribution of the network for the configuration of visible and hidden units.
$p(v)$, $p(h)$	The marginalized distribution of visible/hidden units respectively.
ε	Learning rate of gradient descent.

3.1 RBM: mathematical formulation of the model

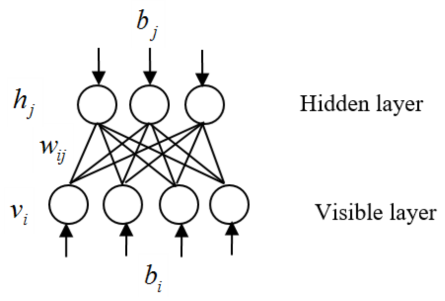

Figure 1. Restricted Boltzmann Machine.

As shown in Figure 1, a Restricted Boltzmann Machine (RBM) is a generative model with a bipartite structure. RBM is analogous to a Markov Random Field (MRF), it has two layers, a visible layer and a hidden layer. The units between the two layers are fully connected, but different from MRF that RBM has no lateral connections within each layer. RBM is also analogous to regular feed-forward neural network with one layer of binary stochastic hidden units and tied weights. The parameters of an RBM consist of weight matrix w_{ij} associated with the connection between hidden unit h_j and visible unit v_i, as well as two bias weight vectors, b_i for the visible units and b_j for the hidden units respectively.

$$E(v,h) = -\sum_{i,j} w_{ij} v_i h_j - \sum_i b_i v_i - \sum_j b_j h_j \tag{1}$$

$$p(v,h) = \frac{e^{-E(v,h)}}{Z} \tag{2}$$

$$Z = \sum_{v,h} e^{-E(v,h)} \tag{3}$$

An RBM is also an energy-based model. The energy over the configuration over all visible and hidden units is defined by (1); the probability distribution which is associated with the energy is defined by (2); and the normalizing factor Z is the sum over all configurations and called the partition function (3).

$$\begin{aligned} p(v) &= \sum_h p(v,h) = \sum_h \frac{e^{-E(v,h)}}{Z} = \frac{1}{Z}\sum_h e^{\sum_{i,j} w_{ij} v_i h_j} \\ &= \frac{1}{Z}\sum_{h_1}\sum_{h_2}\cdots\sum_{h_n}\prod_j e^{\sum_i w_{ij} v_i} \\ &= \frac{1}{Z}\sum_{h_1} e^{\sum_i w_{i1} v_i}\sum_{h_2} e^{\sum_i w_{i2} v_i}\cdots\sum_{h_n} e^{\sum_i w_{in} v_i} \\ &= \frac{1}{Z}\prod_j (1+e^{\sum_i w_{ij} v_i}) \end{aligned} \tag{4}$$

Similarly,

$$p(h) = \sum_v p(v,h) = \frac{1}{Z}\prod_i (1+e^{\sum_j w_{ij}h_j}) \tag{5}$$

The marginalized distribution of visible units (input layer) is factored by (4) and (5), to simplify the equation, temporally ignoring the two bias vectors b_i and b_j. Given the absence of lateral connections, an RBM is a "product of expert" model (Hinton, 2002), the joint distribution of all visible units is combined with each individual visible unit multiplicatively.

$$\begin{aligned} p(h \mid v) &= \frac{p(v,h)}{\sum_h p(v,h)} = \frac{\frac{1}{Z} e^{\sum_{i,j} w_{ij}v_i h_j}}{\frac{1}{Z}\sum_h e^{\sum_{i,j} w_{ij}v_i h_j}} \\ &= \frac{\prod_j e^{\sum_i w_{ij}v_i h_j}}{\prod_j \sum_{h_j} e^{\sum_i w_{ij}v_i h_j}} = \prod_j \frac{e^{\sum_i w_{ij}v_i h_j}}{\sum_{h_j} e^{\sum_i w_{ij}v_i h_j}} \qquad (6) \\ &= \prod_j p(h_j \mid v) \end{aligned}$$

(7)

The conditional distribution of hidden units (hidden layers) given the visible units (input layer) is factored by (6) and (7). Due to the absence of lateral connections, hidden units are independent given the state of the visible units and vice versa.

$$p(v_i \mid h) = Sigmoid(b_i + \sum_j h_j w_{ij}) \tag{8}$$

$$p(h_j \mid v) = Sigmoid(b_j + \sum_i v_i w_{ij}) \tag{9}$$

Activation probability for each individual unit is as shown in function (8) & (9). This activation probability (conditional probability of a single unit) is also analogous to a regular feed-forward neural network (Caers and Ma, 2002).

3.2 RBM: The learning algorithm

Learning an RBM is an unsupervised learning task which models the data distribution by tuning the related parameters. The typical approach is to find the parameters by "maximum likelihood estimation". The log-likelihood gradient of an RBM with respect to the weights is given by (10):

$$\begin{aligned} \frac{\partial \log p(v)}{\partial w_{ij}} &= \frac{\partial}{\partial w_{ij}}(\log \sum_h e^{-E(v,h)}) - \frac{\partial}{\partial w_{ij}}(\log \sum_{v,h} e^{-E(v,h)}) \\ &= \frac{1}{\sum_h e^{-E(v,h)}} \sum_h e^{-E(v,h)} \frac{\partial(-E(v,h))}{\partial w_{ij}} - \frac{1}{\sum_{v,h} e^{-E(v,h)}} \sum_{v,h} e^{-E(v,h)} \frac{\partial(-E(v,h))}{\partial w_{ij}} \\ &= \sum_h p(h \mid v) v_i h_j - \sum_{v,h} p(v,h) v_i h_j = < v_i h_j >_{p(h|v)} - < v_i h_j >_{p(v,h)} \end{aligned} \tag{10}$$

Angle brackets: $< \cdot >_{p(h|v)}$ mean expectation with respect to the data distribution and $< \cdot >_{p(h|v)}$ means expectation with respect to model distribution.

The log-likelihood gradient of an RBM with respect to the bias terms (11) and (12):

$$\frac{\partial \log p(v)}{\partial b_i} = < v_i >_{p(h|v)} - < v_i >_{p(v,h)} \tag{11}$$

$$\frac{\partial \log p(v)}{\partial bj} = < h_j >_{p(h|v)} - < h_j >_{p(v,h)} \tag{12}$$

The first term in (10) is the expected value under the joint distribution (model distribution); the second term in (10) is the expected values under the conditional distribution of hidden units given the training sample feed-into the visible units (data distribution). Both expected values are intractable because they are exponential in the number of units. However, the expected values can be approximated by sampling over the corresponding distributions with Markov Chain Monte Carlo (13).

$$\Delta w_{ij} = \frac{\partial \log P(v)}{\partial w_{ij}} = \varepsilon(< v_i h_j >_{data} - < v_i h_j >_{\mathrm{mod}\,el}) \tag{13}$$

Alternating Block Gibbs Sampling: As shown in function (8) and (9), the units in the visible and hidden layer are both conditionally independent. In this way, all units in one layer can be sampled simultaneously. Thus, the sampling of an RBM involves two alternating steps: sampling the hidden units based on visible units (8) and sampling the visible units based on the hidden units (9). The outline of the learning phase of the algorithm can be summarized below.

The Maximum Likelihood (ML) learning of RBM:
1. Get one training sample from the training dataset.
2. Feed this training sample into the visible units; update all of the hidden units in parallel.
3. Compute $(v_i h_j)_{data}$ by taking the outer product of *v* and *h*.
4. Update all of the visible units in parallel to get a “reconstruction”.
5. Update all of the hidden units again.
6. Repeat 4 and 5 many times (run the chain for several steps).
7. Compute $(v_i h_j)_{\mathrm{mod}\,el}$ by taking the outer product of *v* and *h*.
8. Go back to step 1 using a different training sample.
9. Compute average $< v_i h_j >_{data}$ and $< v_i h_j >_{\mathrm{mod}\,el}$ over all training samples, then update parameters.

Contractive divergence learning of RBM:
By alternating Block Gibbs sampling, the Markov chain could be run “infinite” steps until it reaches its stationary distribution and compute $(v_i h_j)_{\mathrm{mod}\,el}$. Instead of running many steps, Contrastive Divergence (CD) learning takes only one step (CD-1) to speed up.

It’s the same as ML if you removed step 6 and only executed (4) and (5) one time.

3.3 Stochastic simulation using RBM

Given the training image and hard data, learning is required only once. After learning, multiple simulations can be generated by sampling the learned distribution.

3.3.1 Learning joint distribution from training patch database

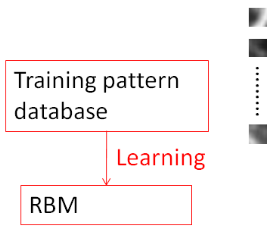

Figure 2. Learn a general appearance model: Learn the model distribution of RBM as the prior distribution of p(v) given the patterns extracted from training image.

As shown as Figure 2, this phase involves two steps:

1. Training samples extraction: Scan the training images using pre-defined patch size and store the patches in a database as training samples.
2. RBM Learning: Learn the model parameters b_i b_j and w_{ij} by contractive divergence learning.

3.3.2 Generating simulations from RBM

Prior to generating simulations, a patch-wise random path is defined that covers the conditioning data image.

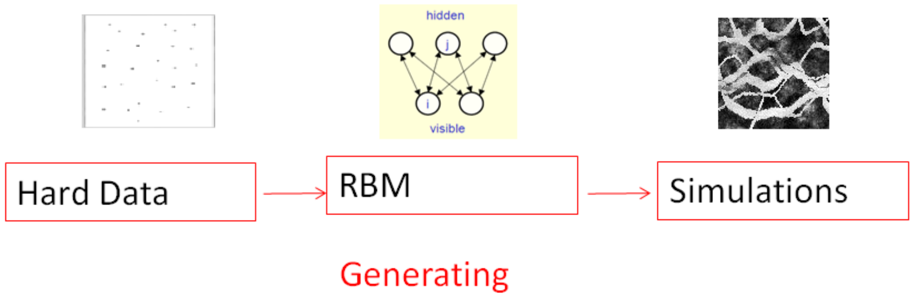

Figure 3. Generating simulation given conditioning data.

As shown in Figure 3, the generating phase involves three repeated steps for each position in the random path:

1. Feed the current conditional data patch (hard data) into the visible units of the trained RBM. The remaining visible units with unknown pixels and all hidden units are randomly initialized.
2. By running several steps of Gibbs sampling, the chain reaches a certain local energy minimum. The values of the visible units are the realization for the current local patch. The visible units where the hard data or simulated values (overlap-patches) are clamped are never updated (frozen).
3. Move to the next location alone the random path, repeat steps 1 and 2. The pattern of the current location could be slightly (less than 25%) overlapped with the pattern of the previous location.

For unconditional simulation, all visible units are associated with unknown pixels. All the visible and all hidden units are randomly initialized. Using Gibbs sampling, RBMSim is able to generate realizations over the entire configuration space.

4 Case study

4.1 Unconditional simulation

The Standford V Reservoir Data Set is used as the training set for unconditional simulation. It contains 30 vertical slices each 100x100 pixels with intensity varying between 0 and 1. One slice of the three-dimensional reservoir data is randomly selected and used for training. The training patch size is 16x16 yielding a total of 7225 training patches.

The dominant parameter of RBM is the weight matrix w_{ij}. It contains the weights on connections between 256 visible units and 500 hidden units. Taking the connections from a single hidden unit (one export) to 256 visible units and reshaping it to represent as 2D feature for visualization. These 500 features are the basis of the simulation operation.

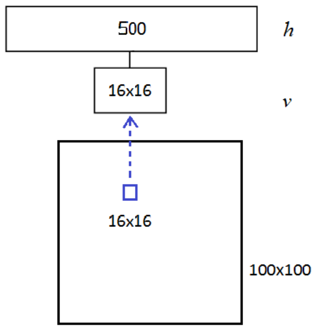

Figure 4. RBM with 500 hidden units and 256 visible units.

Training image is shown in Figure 5a. The channels in the training image are unique but share similar structures.

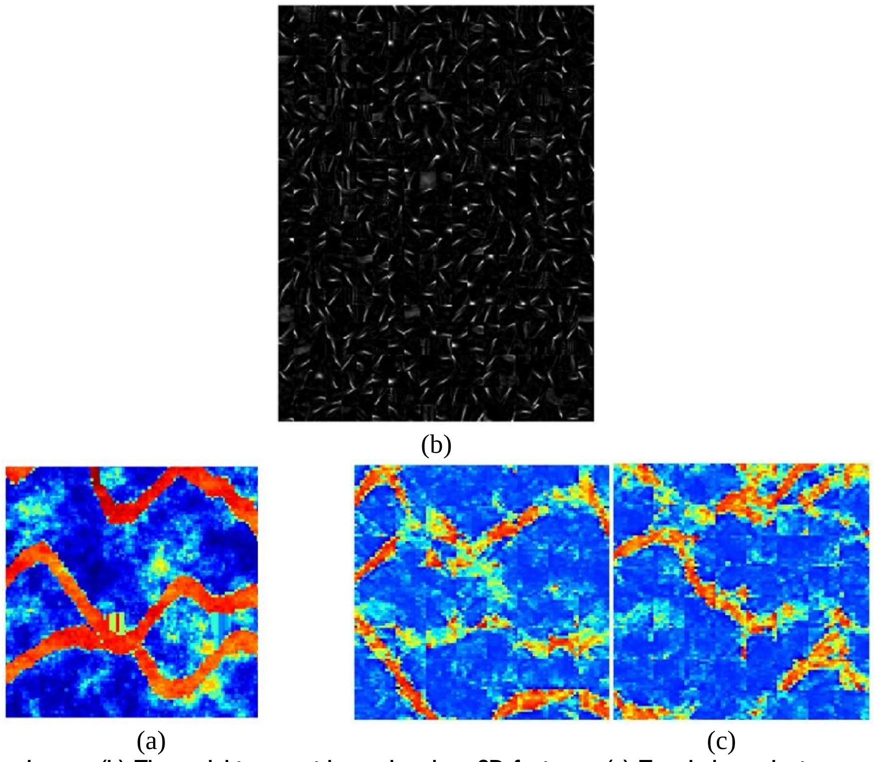

Figure 5. (a) Training image. (b) The weight w_{ij} matrix rendered as 2D features. (c) Two independent unconditional simulations, 100x100 pixels each.

The weights of RBM after training are as shown in Figure 5b in the form of 2D bases. From the figure, it's clear that there are three types of bases: ridgelet, circular, and grating bases. These learned features are similar to those learned from natural images, and they appear to be consistent with other feature learning algorithms (Olshausen et al., 2009). Specifically, the ridgelets are orientated differently, the circular bases are at different locations; and the gratings are of different frequencies.

Two independent, unconditional simulations are shown in Figure 5c. The simulated channels both share similarities with the structures in the training image but are significantly different from one another. The simulated channels are primarily orientated in the horizontal and diagonal direction and the size and shape are analogous to the training image.

The intensity histograms among the training and simulated images are compared in Figure 6, which also reassures that the simulations are consistent with the training data.

4.2 Conditional simulation

The same Standford V Reservoir data set is used as the training set for conditional simulation using an RBM with identical structure to the previous unconditional simulation.

As shown in Figure 8a, another slice of the three-dimensional reservoir data is used for training. The training results are shown in Figure 7 in the form of 2D basis. From the figure, it can be seen that the learned bases are comparable to the learned bases in the unconditional simulation case in that the same three types of bases have emerged.

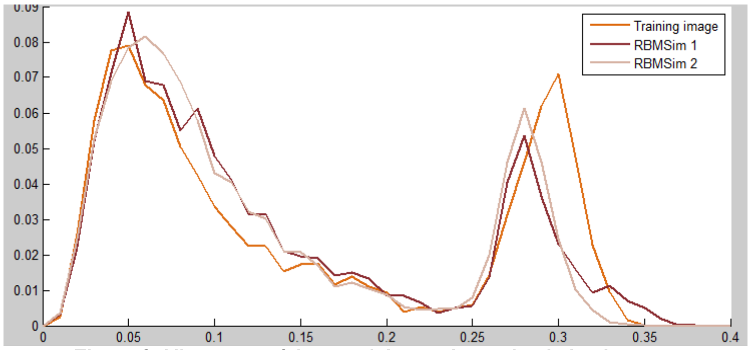

Figure 6. Histogram of three training and two simulation images.

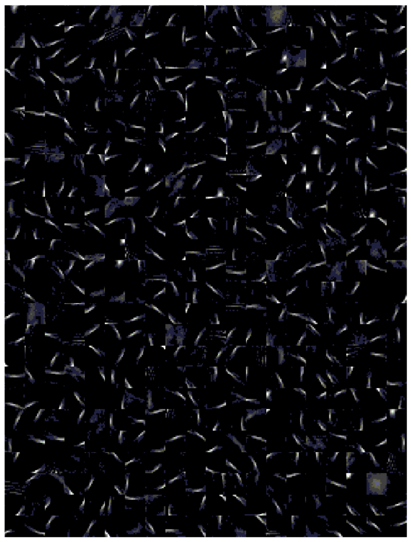

Figure 7. The weight matrix rendered as 2D features.

The conditioning data is taken from the training image shown in Figure 8b, the RBMSim results at different sub-sampling inputs are shown in Figure 8c. For comparison, the WaveSim results using the same simulation conditions are shown in Figure 8d. 3 different conditioning scenarios were explored, where 10% 5% and 2% of the training image data was retained as hard data". While the input hard data is sub-sampled at densities of 10% and 5%, as shown the first two rows in Figure 8 c&d, both algorithms tend to yield reasonable simulation results. However, even in those cases, RBMSim yields a better localization with lower noise.

The most notable discrepancy is sumlation quality between the two simulation methods becomes apparent at the data density of 2%. RBMSim is still able to capture much of the structure visible in Row 3, although there is some degradation in the fine structure. In contrast, most of the structure information is lost in the WaveSim results.

5 Sensitivity analysis

RBM is sensitive to two architecture parameters, the number of visible and hidden units. The number of visible units corresponds to the pattern size. Generally speaking, a larger pattern size tends to better maintain the main geological structure in the simulations; however, the detailed local textures are smoothed out. A smaller pattern size will better reproduce the texture, but at the expense of poor continuity. The number of hidden units largely depends on the number of visible units. The more hidden units, the finer distribution the model can represent. However, the possibility of over fitting is increases, especially when simulating complex geological structures when training data is limited. The ideal configuration of visible and hidden units is far from solved; future research will aim to address this problem.

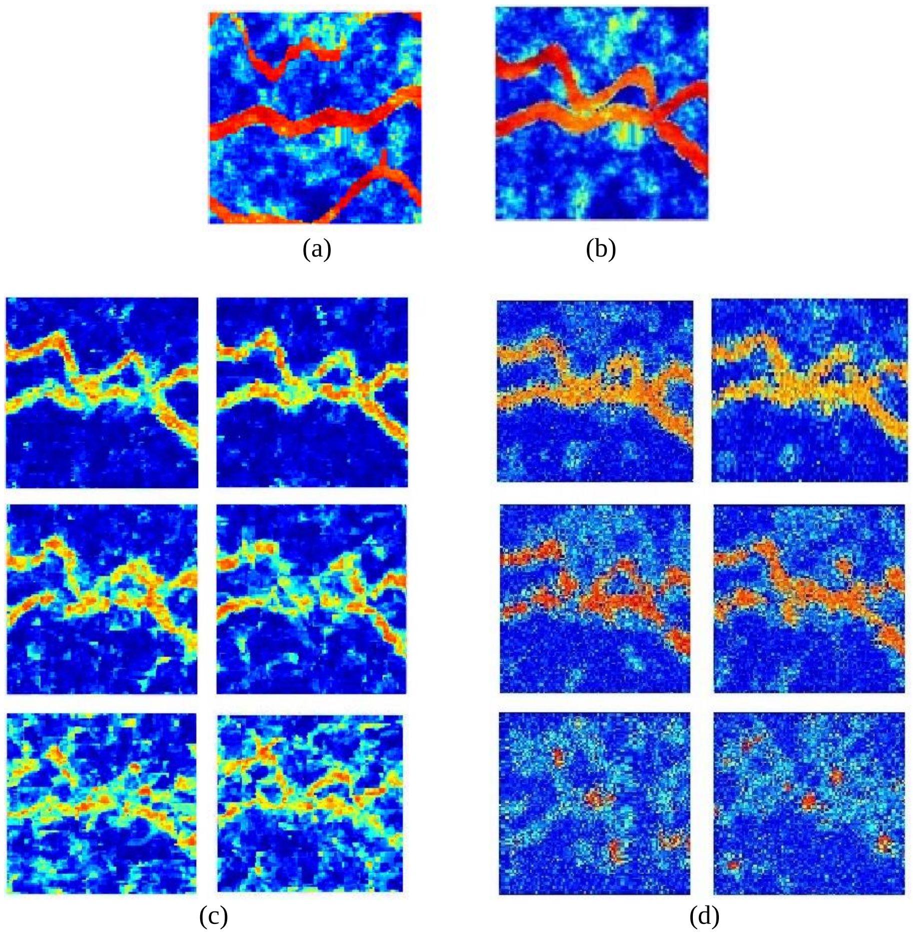

Figure 8 .(a) Training image for conditional simulation. (b) Testing image from which the hard data is sampled. (c)&(d) simulation results: From top to bottom, each row contains two independent conditional simulation on hard data sub-sampled at densities of 10%, 5% and 2% respectively. RBMSim on (c) and WaveSim on (d).

A comparison of RBMSim with different number of visible units is shown as in Figure 9. Three RBMs with 64/200, 256/500 and 1024/2000 visible/hidden units respectively are trained by the same training image and simulations are generated with the same hard data. The training image is the same as conditional simulation in 4.2 which is shown in Figure 7a. The test image is shown in Figure 9a. The hard data is sub-sampled at densities of 20%. The simulation result of RBMSim with 64/200, 256/500 and 1024/2000 visible/hidden units are shown in Figure 9b,c,d. The simulation results using the 8x8 pattern size and 200 hidden units (64/200) are quite poor; the overall structural continuity is lost, and many pattern boundary artifacts are observed. When the pattern size is 16 x 16 with 500 hidden units, the simulation is well balanced between the local texture and overall structure. The cumulant maps between 9a and 9c are very similar. Using a pattern size of 32x32 and 2000 hidden units reproduces the main structure well, but the detailed texture information is smoothed out. The cumulant map shows that the simulation has stronger local correlation than the testing image due to this smoothing effect.

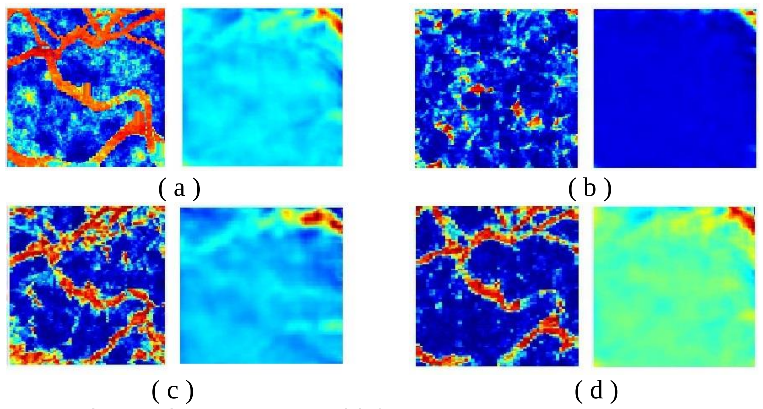

Figure 9. Simulations and their x-y (L shape) cumulant maps. (a) Original testing image and its cumulant map. (b) Simulation by RBM has 64 visible and 200 hidden units (8x8 pattern size) and its cumulant map. (c) Simulation by RBM has 256 visible and 500 hidden units (16x16 pattern size) and its cumulant map. (d) Simulation by RBM has 1024 visible and 2000 hidden units (32x32 pattern size) and its cumulant map.

6 Discussion

We will discuss the algorithms step by step and compare the difference between RBMSim and other pattern-based simulation.

The algorithm for pattern-based (multiple points) simulation is usually executed in the following way:

1) Pattern Extraction
2) Dimensional Reduction of Patterns
3) Pattern Clustering
4) Define a random path
5) Classification for the current patch
6) Random sampling in the class
7)

The first three steps aim to learn the joint distribution and the last three generate the sequential simulation. Each of these steps is detailed below:

Pattern extraction: Scan the training image with a patch (of fixed size) over all spatial locations and store them in a pattern database.
Dimensional reduction: In the configuration space of the training patterns, which is a large multidimensional space, locate the sub-space where the samples of training patches are densely located.
Define a random path: Define a path that visits the entire simulation grid.
Pattern clustering: Partition that subspace from the dimensional reduction into several clusters. The cluster centre is the prototype of each class.
Classification for the current patch: For each patch containing hard data in the random path, allocate to a cluster using some similarity measure measurement with the class prototypes;
Random sampling in the class: Sample that cluster for a pattern and freeze the realization to the grid. This introduces the stochasticity in many simulation algorithms.

Each of these steps in different simulation algorithm, (e.g. SimPat/FilterSim/WaveSim) use different sub-algorithms to achieve the desired goal. RBM organises the process in a coherent way and shares the same simple sampling algorithm. For comparison, the RBMSim framework will now be detailed in the context of the traditional pattern-based approach.

6.1 Algorithm of learning joint distribution

The first three steps of the conventional work flow are combined into to the learning phase of the RBSim framework. The first step consists of simply gathering the training data, which is not formally part of the learning algorithm; the main algorithm involves the two remaining steps: dimensional reduction and pattern clustering. RBM aggregates the two steps together and returns a feature-based representation.

Generally speaking, the most popular linear dimensionality reduction technique is principle component analysis (PCA). While using linear units rather than sigmoid units, RBM would behave quite similar to PCA. After learning, the bases weight vectors in the weight matrix will span the same subspace as the first k component as in PCA. Here the stochastic binary (sigmoid) units are used so RBM can be conceived as a non-linear variant of PCA. RBM has been used for dimensionality reduction of multiple datasets (Hinton and Salakhutdinov, 2006). The bases are the key for Dimensional Reduction; WaveSim is very sensitive to the selected wavelet basis. The advantage of learning the features is that it guarantees that the learned features are the best bases for representing the training patterns. Clustering is an approach for partitioning a large space to a number of sub-regions. The K-means clustering algorithm partitions the space with linear decision boundaries (lines, planes or hyper-planes etc.). RBM models the input configuration space with an energy surface, after learning, there would be finite number of local energy minima. Those local energy minima act as the "prototype" of each cluster. In contrast to K-means, which partition the space explicitly, RBM partition the space implicitly. On the energy surface of RBM, the local minima are bounded by the surrounding ridges. These ridges are the non-linear decision boundaries between local minima.

There are several advantages to this "soft" partitioning. First, the tedious section of the hyper parameter, k, is avoided altogether. Secondly, it increases the stochasticity of the simulation in that it's possible to generate samples at different, but nearby, local energy minim on the energy surface. This flexibility will increase the variation of the simulation, especially when there is strong ambiguity.

6.2 Algorithm of stochastic simulation

The final three steps of the conventional framework are simultaneously combined in the generation phase of RBSim. Similar to the learning phase, where RBMSim does not have an explicit partitioning (clustering), and the generation phase of RBMSim does not have an explicit classification process (similarity measure). RBMSim aggregates the two steps in simulation (classification and sampling) with single sampling algorithm. The patterns are reproduced directly by sampling from the model distribution. RBM can sample over the entire configuration space. While the subspace of the conditional data has several local minima, RBMSim is capable of generating samples from each of them rather than just generating samples from one class.

The Gibbs chain is able to jump from one local energy minimum to a nearby local energy minimum, which allows for the exploration of multiple clusters within the configuration space without re-initialization. In contrast, WaveSim would first find a single best-match class then samples only using the conditional cumulative distribution function of that particular class rather than the entire configuration space.

7 Conclusions

In this paper, a learning based probabilistic graph model for multiple points simulation is proposed (RBMSim). In RBMSim, the pattern database is mathematically formalized by an energy surface (a distribution) over the configuration space of RBM; searching for the best matched pattern, given conditional data, is mathematically formalized using Gibbs sampling. The algorithm is verified by performing conditional and unconditional two-dimensional simulation of naturally occurring channels (or however they're formally described). The algorithm reproduces the continuity of the channels while maintaining the underlying similarity to the training data. A theoretical discussion and experimental comparison with WaveSim is also explored.

The main weakness of the proposed algorithm is that the learning phase takes a very long time especially when the training pattern database is large. As the learning is done, for each incoming conditioning data, the simulation is almost real time.

RBMSim suffer from the limitation as the other pattern based simulation methods. It's a training image driven algorithm rather than data driven. Therefore, while there are conflicts between training and hard data, the simulation results compromise the statistic of both.

References

Arpat, B., & Caers, J. (2007). Conditional simulation with patterns. Mathematical Geology, 39, 177–203.

Arpat, G. B., & Caers, J. (2005). A multiple-scale pattern-based approach to sequential simulation. Geostatistics Banff 2004, 255–264.

Brakel, P., Dieleman, S., & Schrauwen, B. (2012). Training Restricted Boltzmann Machines with Multi-tempering: Harnessing Parallelization. Artificial Neural Networks and Machine Learning, 92–99.

Caers, J., & Ma, X. (2002). Modeling Conditional Distributions of Facies from Seismic Using Neural Nets. Mathematical Geology, 34, 143–167.

Chatterjee, S., Dimitrakopoulos, R., & Mustapha, H. (2012). Dimensional Reduction of Pattern-Based Simulation Using Wavelet Analysis. Mathematical Geosciences, 44, 343–374.

Colin, D. (2004). Higher Order Models using Entropy, Markov Random Fields and Sequential Simulation. Geostatistic Banff 2004, 215–224.

Desjardins, G., Courville, A. C., Bengio, Y., Vincent, P., & Delalleau, O. (2010). Tempered Markov chain Monte Carlo for training of restricted Boltzmann machines, 145–152.

Hinton, G. (2002). Training products of experts by minimizing contrastive divergence. Neural computation, 14, 1771–1800.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. Science, 313, 504–507.

Mustapha, H., & Dimitrakopoulous, R. (2010). High-order Stochastic Simulation of Complex Spatically Distributed Natural Phenomena. Math Geosci, 42, 457–485.

Olshausen, B., Cadieu, C., & DK, W. (2009). Learning real and complex overcomplete representations from the statistics of natural images. SPIE Proceedings, 7446.

Straubhaar, J., Renard, P., Mariethoz, G., Froidevaux, R., & Besson, O. (2011). An improved parallel multiple-point algorithm using a list approach. Mathematical Geosciences, 43, 305–328.

Strebelle, S. (2002). Conditional simulation of complex geological structures using multiple-point statistics. Mathematical Geology, 34, 1–21.

Strebelle, S., & Cavelius, C. (2014). Solving Speed and Memory Issues in Multiple-Point Statistics Simulation Program SNESIM. Mathematical Geosciences, 46, 171–186.

Tieleman, T., & Hinton, G. (2009). Using fast weights to improve persistent contrastive divergence. Proceedings of the 26th Annual International Conference on Machine Learning, 1033–1040.

Tijmen, T. (2008). Training restricted Boltzmann machines using approximations to the likelihood gradient. Proceedings of the 25th international conference on Machine learning, 1064–1071.

Wu, J., Zhang, T., & Journel, A. (2008). Fast FILTERSIM Simulation with Score-based Distance. Mathematical Geosciences, 40, 773–788.

Zhang, T., Pedersen, S. I., Knudby, C., & McCormick, D. (2012). Memory-efficient categorical multi-point statistics algorithms based on compact search trees. Mathematical Geosciences, 44, 863–879.