Deep Belief Networks

1. Introduction

Belief networks are probabilistic model that represents a set of variables and their conditional dependencies with a directed acyclic graph. Deep belief networks combine a probabilistic model with the power of deep neural networks, creating a model that can we used for both supervised and unsupervised learning.

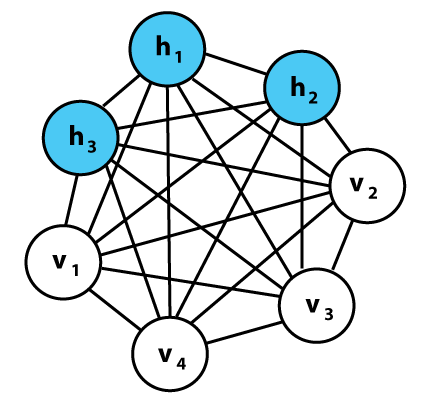
This model arose as a response to the limitations of back-propagation. Back-propagation algorithms require labeled data and a lot of data is unlabeled. Using local computation does not guarantee finding the global minima and can get struck in a local minima. In today’s networks that use back-propagation leads to the vanishing and exploding gradients, where gradients become either too small or too big and the network cannot correctly update its weights in order to arrive at a minima.

A deep belief network (DBN) is an autoencoder, meaning that it encodes the structure of the data. It is composed of stacked restricted Boltzmann machines, which we will discuss in the following section. Given an unlabeled dataset, a DBN can probabilistically reconstruct its inputs, serving the purpose of feature detection. A useful application is feature extraction, that can reduce the dimension of the data and help use infer similarity within the data. For the task of supervised learning, we can use the features extracted to train a simpler classifier then we will normally use for a given task.

1. Boltzmann Machines

Boltzmann machines are generative stochastic artificial neural networks that are able to learn an internal representation of a set of data. They are based on the Boltzmann distribution, which, in physics, is a probability distribution that gives the probability that a system can be observed in a certain state in terms of it’s energy and temperature. They were invented in 1985 by Geoffrey Hinton, then a Professor at Carnegie Mellon University, and Terry Sejnowski, then a Professor at Johns Hopkins University.

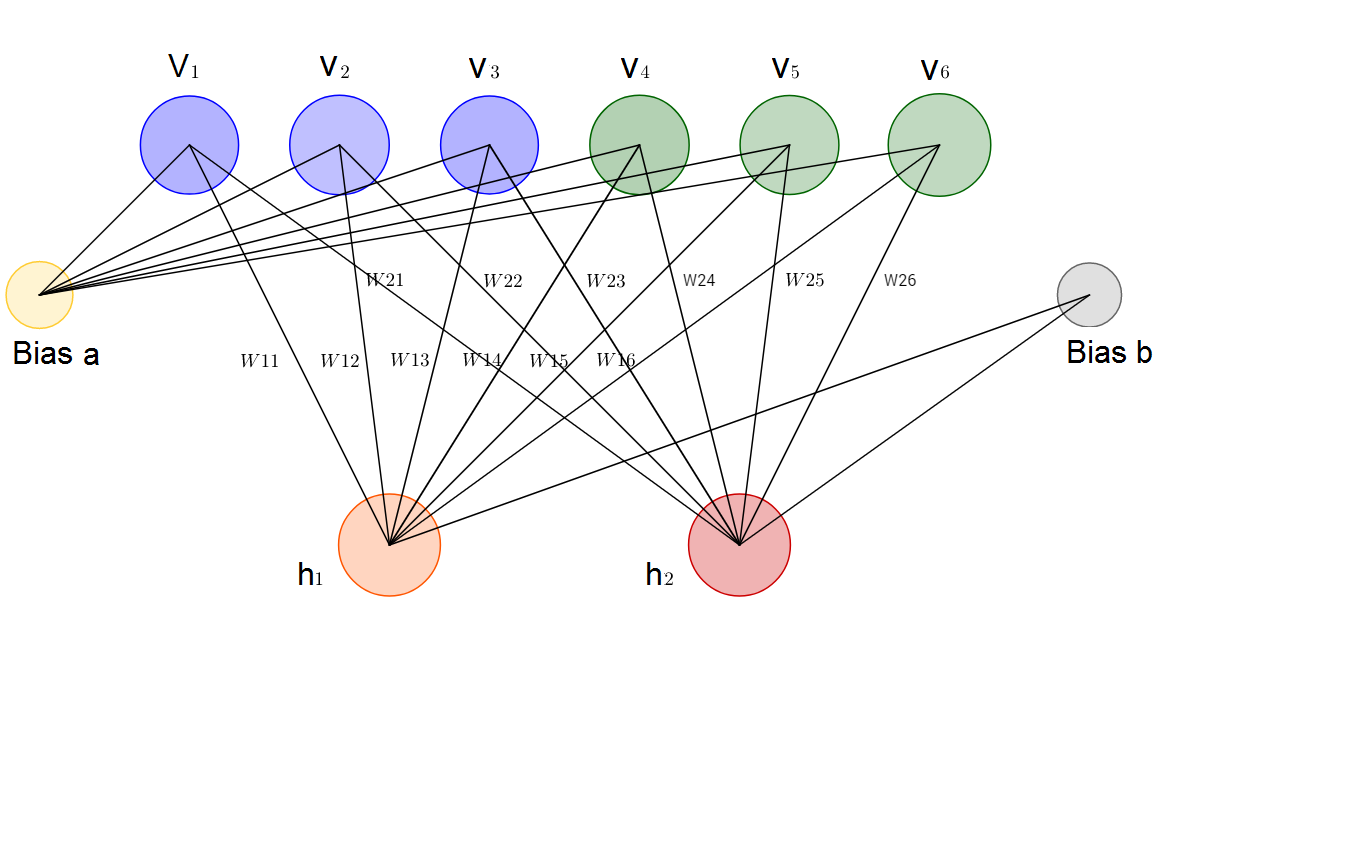
Let’s dive deeper into the definition. A generative model is a model that can produce an abstraction of the data set and can even generate new data instances, as opposed to discriminative models, that distinguishes between different kinds of data instances. A stochastic network introduces random variations into the network for the purpose of escaping local minima. Here is the architecture:



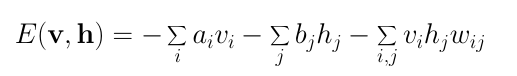
As we can observe, the network has two types of nodes: hidden and visible. The input is given to the visible layer and the network tries to reconstruct it. All the nodes are connected, even the visible one, unlike in many classic networks. This allows them the share information among themselves. Each neuron has a binary value, and the chance of it firing depends on the other neurons in the network. We will discuss its inner workings in the context of the next section.

1. Restricted Boltzmann Machines (RBM)

The restriction imposed to a Boltzmann Machine is that nodes in the same layer (visible or hidden) are not connected. Thus, a restricted Boltzmann machine is fully connected bipartite graph, where the two partitions are called visible and hidden layers. This restriction allows for easier computation and a larger network. This is the architecture:

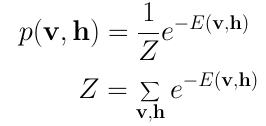


RBMs are energy-based models, similar to Hopfield networks. In some deep learning architectures, the idea of energy Is used as a measure of the model’s quality. The goal is to encode dependencies between variables and it is reached by associating a scalar energy to each configuration of values. A high energy means bad compatibility. An energy-based model tries to minimize the energy function, which in the case of RBMs is:

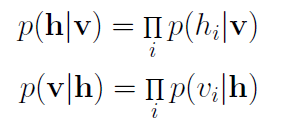


We can notice that the value of the energy function depends on the configuration of visible states, hidden states, weights and biases. In training of an RBM the goal is to find parameters for the input values such that the energy is minimized.

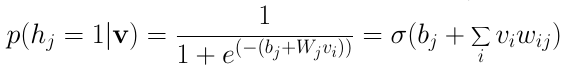
RBMs assigns probabilities instead for discrete values. At each point in time the RBM is in a certain state given by the values of the neurons in the visible(v) and hidden(h) layers. The probability that a certain state of v and h can be observed is given by the following joint distribution:

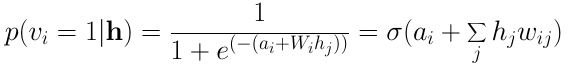


Because it is difficult to calculate the joint probability due number of possible combinations of v and h in the partition function Z. We can instead calculate the conditional probabilities of h given v and the conditional probabilities of v given h:



Each neuron in a RBM has binary state. We can compute the probability of a neuron in the hidden or visible layer to be activated:



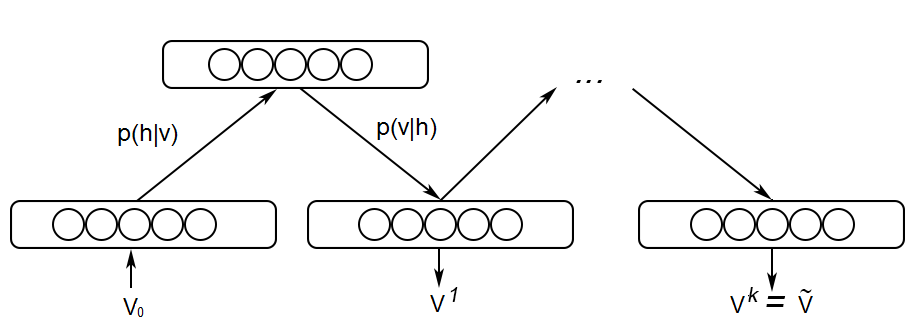


The training of a RBM uses a stochastic gradient descent. It includes two main steps: Gibbs sampling and contrastive divergence.

Gibbs sampling is a Markov chain of a joint of N random variables S=(S_1, ... , S_N) is done through a sequence of N sampling sub-steps. For RBMs, since the visible and hidden are conditionally independent, we can perform block Gibbs sampling. Given this, the visible layer can be sampled simultaneously given fixed value of the hidden layer and the hidden layer can be sampled simultaneously given fixed value of the visible layer, as follows:

h^{(n+1)} &\sim sigm(W'v^{(n)} + c) \\
v^{(n+1)} &\sim sigm(W h^{(n+1)} + b),

This can be illustrated graphically:



As t tends to infinity the samples are guaranteed to be accurate samples of p(v, h). Since we cannot sample to infinity it is sufficient to sample k times.

Contrastive divergence is the algorithm updated the weight matrix. V(0) and V(k) are used to calculate the activation probabilities for the hidden values H(0) and H(k). The update matrix is given by the difference between the outer products of those probabilities:

Image for post

Using the update matrix the new weights can be calculated with gradient **ascent,**given by:

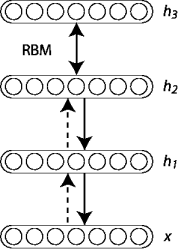
Image for post

1. Deep Belief Networks (DBN) Architecture

A DBN is structurally the same as an MLP but is trained by considering stacking RBMs. DBN are graphical models that learn a representation of the training data. They model the joint distribution between observed vector x and the hidden layers h as follows:

P(x, h^1, \ldots, h^{\ell}) = \left(\prod_{k=0}^{\ell-2} P(h^k|h^{k+1})\right) P(h^{\ell-1},h^{\ell})

where x=h^0, P(h^{k-1} | h^k) is condition distribution for the visible units conditioned on the hidden units of the RBM at the k level and P(h^{\ell-1}, h^{\ell}) is the visible-hidden joint distribution in the top level RBM. We can observe this in the following figure:



The DBN follows a greedy layer-wise unsupervised training described in the paper [10]. The algorithm is as follows:

1. Train the first RBM that models the input data as its visible layer.

2. Use that first layer to obtain a representation of the input that will be used as data for the second layer. Two common solutions exist. This representation can be chosen as being the mean activations p(h^{(1)}=1|h^{(0)}) or samples of p(h^{(1)}|h^{(0)}).

3. Train the second layer as an RBM, taking transformed data as training examples.

4. Iterate (2 and 3) for the desired number of layers, each time propagating upward either samples or mean values

5. Fine-tune all the parameters of the deep architecture.

1. Applications and issues

A DBN can be used for both supervised and unsupervised tasks.

Deep belief networks can be used in image recognition. A picture would be the input, and the category the output. This technology has broad applications, ranging from relatively simple tasks like photo organization to critical functions like medical diagnoses. For example, smart microspores that can perform image recognition could be used to classify pathogens. This would alleviate the reliance on rare specialists during serious epidemics, reducing the response time.

Video recognition also uses deep belief networks. Video recognition works similarly to vision, in that it finds meaning in the video data. For example, it can identify an object or a gesture of a person. It can be used in many different fields such as home automation, security and healthcare.

Motion capture data involves tracking the movement of objects or people and also uses deep belief networks. Motion capture is tricky because a machine can quickly lose track of, for example, a person - if another person that looks similar enters the frame or if something obstructs their view temporarily. Motion capture thus relies not only on what an object or person look like but also on velocity and distance. Motion capture is widely used in video game development and in filmmaking.

An issue with the DBN is that it is a shallow network and does not take advantage of a deep architecture that is possible to train nowadays. It was described in 2006, and, for its time, it achieved state of the art results on problems like MNIST.

1. Data Sets Used
   1. Wine Quality Data Set

Two datasets are included, related to red and white wine samples, from the north of Portugal. The goal is to model wine quality based on physicochemical tests. [16]

The attribute information:

Input variables (based on physicochemical tests):  
1 - fixed acidity  
2 - volatile acidity  
3 - citric acid  
4 - residual sugar  
5 - chlorides  
6 - free sulphur dioxide  
7 - total sulphur dioxide  
8 - density  
9 - pH  
10 - sulphates  
11 - alcohol  
Output variable (based on sensory data):  
12 - quality (score between 0 and 10)

Some samples are:

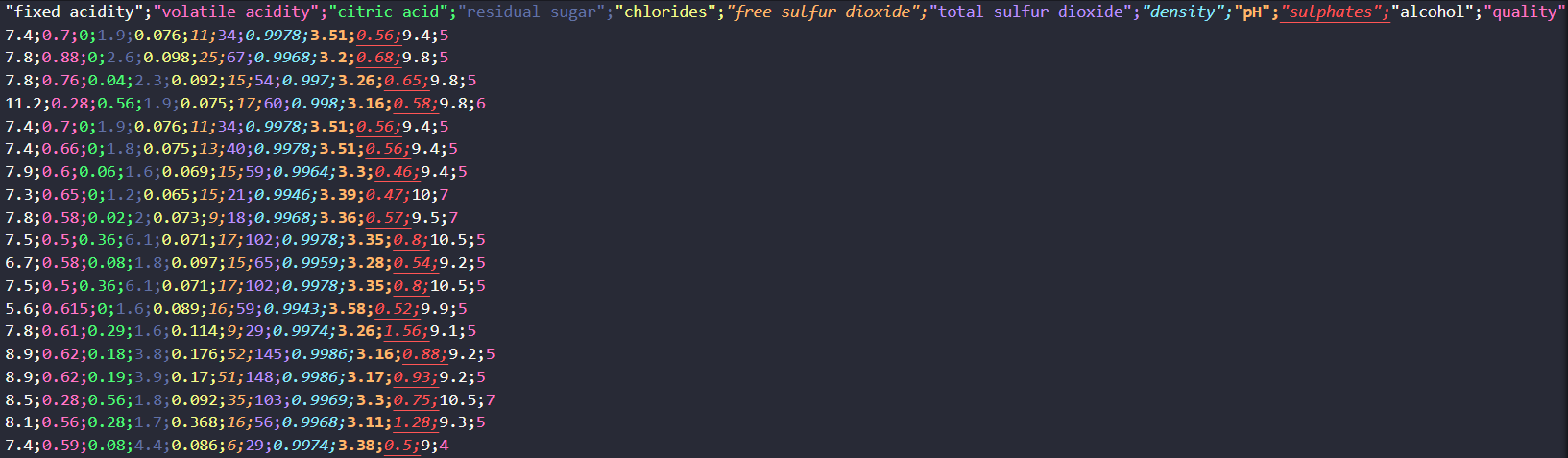


Figure 1. Red Wine Data Set Sample

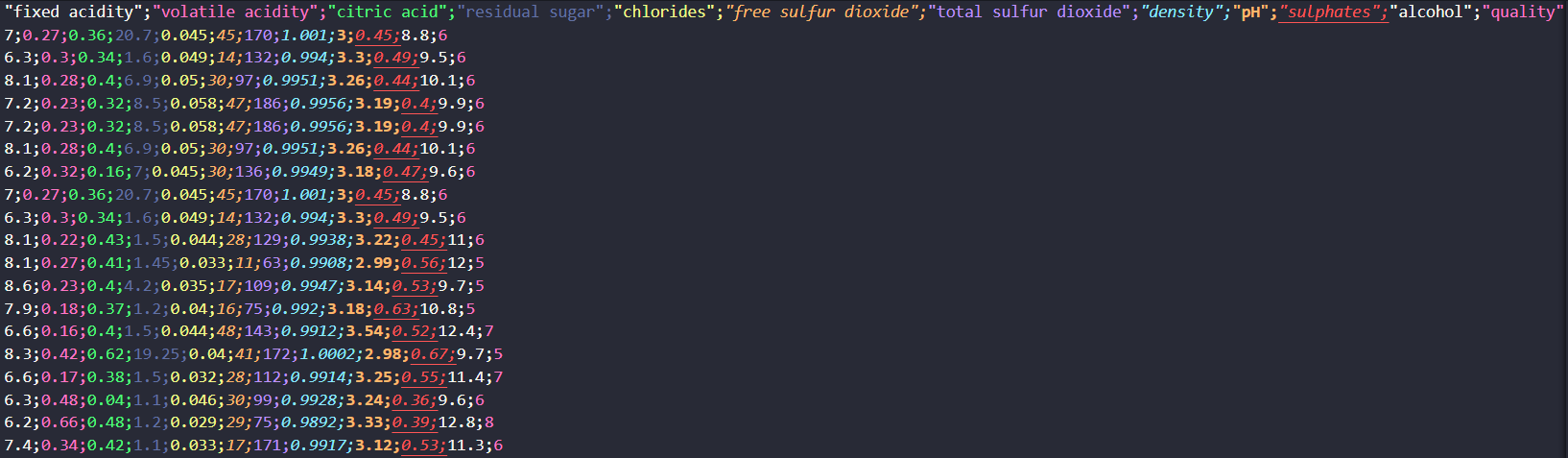


Figure 2. White Wine Data Set Sample

* 1. Bank Marketing Data Set

The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

The attribute information:

Input variables:  
# bank client data:  
1 - age (numeric)  
2 - job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')  
3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)  
4 - education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')  
5 - default: has credit in default? (categorical: 'no', 'yes', 'unknown')  
6 - housing: has housing loan? (categorical: 'no', 'yes', 'unknown')  
7 - loan: has personal loan? (categorical: 'no', 'yes', 'unknown')  
# related with the last contact of the current campaign:  
8 - contact: contact communication type (categorical: 'cellular', 'telephone')  
9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')  
10 - day\_of\_week: last contact day of the week (categorical: 'mon', 'tue', 'wed', 'thu', 'fri')  
11 - duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.  
# other attributes:  
12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)  
13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)  
14 - previous: number of contacts performed before this campaign and for this client (numeric)  
15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')  
# social and economic context attributes  
16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)  
17 - cons.price.idx: consumer price index - monthly indicator (numeric)  
18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)  
19 - euribor3m: euribor 3 month rate - daily indicator (numeric)  
20 - nr.employed: number of employees - quarterly indicator (numeric)  
  
Output variable (desired target):  
21 - y - has the client subscribed a term deposit? (binary: 'yes','no')

Some samples are:



Figure 3. Bank Marketing Data Set

* 1. MNIST Data Set

The dataset contains images of hand-written digits, from 0 to 9. Some samples are:



Figure 4. MNIST Data Set

1. Results
   1. Wine Quality Data Set

We treat this problem like a regression problem. We compare the results with a regressor using a DBN and a Ridge regressor. The r2 scores are, for the red wine: DBN regressor -> 0.3473308034958251 and Ridge regressor -> 0.3293642538047954, and for white wine: DBN regressor -> 0.30096417836718434 and Ridge regressor -> 0.25788728218319523. We can observe that the DBN regressor obtains better results, although not great one. This might mean that there is no strong correlation between the physicochemical properties and its rating.

Some predicted ratings are:

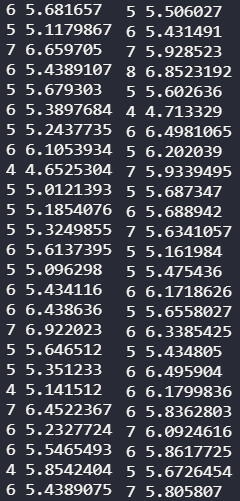


Figure 5. Red wine prediction(left) and white wine prediction(right)

* 1. Bank Marketing Data Set

This is a classification problem. The accuracy obtained 0.9291721291575625. It is interesting to see some examples of false negatives:

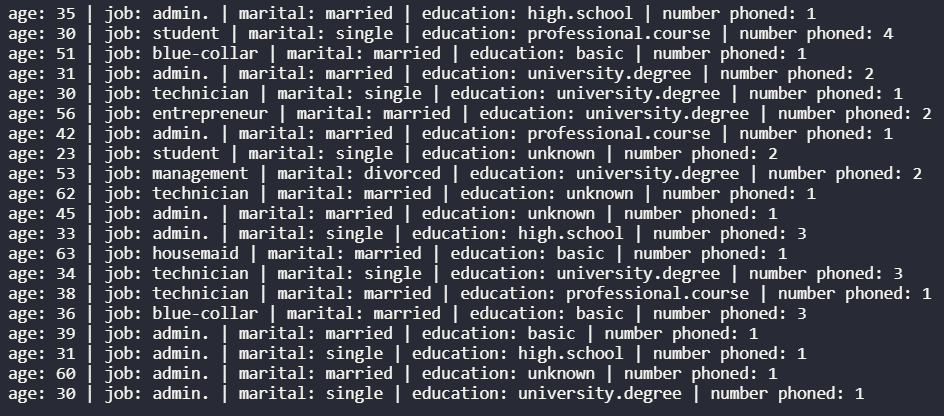


Figure 6. Bank marketing false negative

* 1. MNIST Data Set

Since we obtain some new features after running the DBN, it is interesting to see what are the closest images for the closest vectors to some random image:

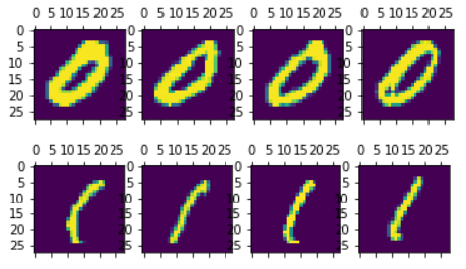


Figure 7. A random image (furthest to the left) and the closest 3 images according to obtained features

Next, we are interested in using the obtained features for training a logistic regression model. The obtained accuracies are 0.9186 for a logistic regression using the raw pixels and 0.9409 for a logistic regression using the DBN features.

Lastly, we are interested in seeing the impact of the DBN pre-training versus no pre-training, on the same number of epochs. The obtained accuracies are 0.9831 with no pre-training and 0.9925 with pre-training.

1. References

[1] <https://en.wikipedia.org/wiki/Bayesian_network>

[2] <https://machinelearningjourney.com/index.php/2020/08/07/vanishing-and-exploding-gradients/>

[3] <https://en.wikipedia.org/wiki/Restricted_Boltzmann_machine>

[4] <http://deeplearning.net/tutorial/rbm.html>

[5] <https://towardsdatascience.com/restricted-boltzmann-machines-simplified-eab1e5878976>

[6] <https://towardsdatascience.com/deep-learning-meets-physics-restricted-boltzmann-machines-part-i-6df5c4918c15>

[7] <https://en.wikipedia.org/wiki/Deep_belief_network>

[8] <http://deeplearning.net/tutorial/DBN.html>

[9] <https://missinglink.ai/guides/neural-network-concepts/deep-belief-networks-work-applications/>

[10] <https://www.cs.toronto.edu/~hinton/absps/fastnc.pdf>

[11] <https://www.cs.toronto.edu/~hinton/nipstutorial/nipstut3.pdf>

[12] <https://arxiv.org/pdf/1807.03953v1.pdf>

[13] <https://www.cs.toronto.edu/~hinton/absps/ruhijournal.pdf>

[14] <https://github.com/albertbup/deep-belief-network>

[15] <https://archive.ics.uci.edu/ml/datasets/Bank+Marketing>

[16] <https://archive.ics.uci.edu/ml/datasets/wine+quality>

[17] <https://www.kaggle.com/oddrationale/mnist-in-csv>