

Joint Representation image-text with Multimodal Deep Boltzmann Machine

Achari Berrada Youssef
MVA, ENS Paris Saclay

youssef.achari-berrada@polytechnique.edu

Keywords : BOLTZMANN MACHINES, UNSUPERVISED LEARNING, MULTIMODAL LEARNING, NEURAL NETWORKS, DEEP LEARNING

Abstract

Deep Boltzmann Machine is described for learning a generative model of data that consists of multiple and diverse input modalities. The model can be used to extract a unified representation that fuses modalities together. This project aims to use a multimodal Deep Boltzmann Machine architecture in order to learn a good generative model for the joint space of image and text input that can be used for information retrieval from both unimodal and multimodal queries.

1. Introduction

1.1. Context

In this project, we want to train multimodal Deep Boltzmann Machine in order to do joint representation between image and text. The project is inspired from the work of Nitish Srivastava in [4].

1.2. The goal of the project

In real life, we can find the data in a multimodal way. Image can come with annotations. The goal is to learn a joint distribution over images and text $P(v_{img}.v_{txt}; \theta)$. And then by drawing samples from $P(v_{img}|v_{txt}; \theta)$ and $P(v_{txt}|v_{img}; \theta)$ we can fill-in the missing data, thereby doing image retrieval and image annotation respectively.

1.3. Motivation

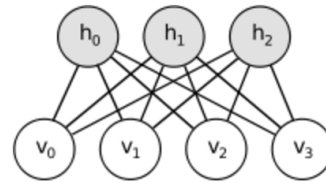
The intuition behind our model is that each data modality has very different statistical properties which make it difficult for a single-layer model to directly find correlations in features across modalities. In our model, this difference is bridged by putting layers of hidden units between the modalities.

1.4. Structure of the report

As the project is a joint project between the Probabilistic Graphical Model course by Prof. Francis Bach and the Object Recognition & Computer Vision course by Prof. Josef Sivic, the report starts first by describing the model and then presents the results of the experiments. In more details, in section 2, we describe the Building Blocks of the architecture we used in our model and an explanation of the training phases of the RBM. We use the Block-Gibbs sampling method for training the Deep Boltzmann Machine as explained in section 3. Then in section 4, we will describe the data used in our model. And in section 5, we will describe the architecture of the model. In section 6, we will show the tools used to do the experiments. Finally in the section 7 we will describe the experiments performed.

2. Building Blocks

A Restricted Boltzmann Machine is an undirected graph, with two layers. A first layer of visible units and a second one for hidden units. It's a special case of a Boltzmann Machine.



2.1. RBMs

Restricted Boltzmann Machines (RBMs) have been used effectively in modeling distributions over binary-valued data.

$$E(v, h; \theta) = - \sum_{i=1}^D \sum_{j=1}^F W_{i,j} v_i h_j - \sum_{i=1}^D b_i v_i - \sum_{j=1}^F a_j h_j \quad (1)$$

where $\theta = \{\mathbf{W}, \mathbf{a}, \mathbf{b}\}$ are the model parameters. The joint distribution over the visible and hidden units is defined by :

$$P(v, h; \theta) = \frac{1}{\mathcal{Z}(\theta)} \exp(E(v, h; \theta)), \quad (2)$$

$$\mathcal{Z}(\theta) = \sum_v \sum_h E(v, h; \theta)$$

From equation (1) and (2), we can factorize the conditional distributions over h and v as:

$$P(h|v; \theta) = \prod_{j=1}^F p(h_j|v), \quad p(h_j = 1|v) = g\left(\sum_{i=1}^D W_{i,j} v_i + a_j\right)$$

$$P(v|h; \theta) = \prod_{i=1}^D p(v_i|h), \quad p(v_i = 1|h) = g\left(\sum_{j=1}^F W_{i,j} h_j + b_i\right)$$

and g is the sigmoid function defined by $g(x) = \frac{1}{1 + \exp(-x)}$ is the logistic function.

2.2. Gaussian-Bernoulli RBM

When modeling a visible real-valued units, we use Gaussian RBMs which is an RBM with stochastic real valued visible variables $\mathbf{v} \in \mathbb{R}^D$, and stochastic binary hidden variables $h \in \{0, 1\}^F$. The energy of the state $\{h, v\}$ is then defined as :

$$E(v, h; \theta) = \sum_{i=1}^D \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{i=1}^D \sum_{j=1}^F \frac{v_i}{\sigma_i} W_{i,j} h_j - \sum_{j=1}^F a_j h_j \quad (3)$$

where $\theta = \{\mathbf{a}, \mathbf{b}, \mathbf{W}, \sigma\}$ are the model parameters. Similar to the binary RBM, the conditional distribution factorize as:

$$P(h|v; \theta) = \prod_{j=1}^F p(h_j|v), \text{ with}$$

$$p(h_j = 1|v) = g\left(\sum_{i=1}^D W_{i,j} \frac{v_i}{\sigma_i} + a_j\right)$$

$$P(v|h; \theta) = \prod_{i=1}^D p(v_i|h), \text{ with}$$

$$v_i|h \sim \mathcal{N}\left(b_i + \sigma_i \sum_{j=1}^F W_{i,j} h_j, \sigma_i^2\right)$$

2.3. Replicated Softmax Model

We can modify an RBM in order to model word count vectors. In the replicated softmax model, the energy of the state is defined as:

$$E(v, h) = - \sum_{i=1}^M \sum_{j=1}^F \sum_{k=1}^K W_{i,j}^k v_i^k h_j - \sum_{i=1}^M \sum_{k=1}^K b_i^k v_i^k - \sum_{j=1}^F a_j h_j \quad (4)$$

$$P(h_j|v) = g\left(M a_j + \sum_{k=1}^K \sum_{i=1}^M v_i^k W_{i,j}^k\right)$$

$$P(v_i^k = 1|h; \theta) = \frac{\exp(b_i^k + \sum_{j=1}^F h_j W_{i,j}^k)}{\sum_{q=1}^K \exp(b_i^q + \sum_{j=1}^F h_j W_{i,j}^q)}$$

2.4. Deep Boltzmann Machine

A Deep Boltzmann Machine (DBM) is a concatenation of RBMs. It contains a set of visible units \mathbf{v} and a sequence of layers of hidden binary units $h^{(l)} \in \{0, 1\}^{F_l}$ with $l \in \{1, \dots, L\}$. It was introduced by Salakhutdinov et al. in [5] in order to model complex features. More layers we have, more clear the representation is, and more harder is to train the model.

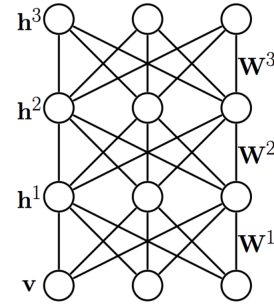


Figure 1: Deep Boltzmann Machine

In our experiment, we will train an image-specific two-layer DBM that uses a Gaussian model to model the distribution over real-valued image features. Then we train another text-specific two-layer DBM that uses a Replicated Softmax model to model the distribution over the word count vectors.

2.5. Multimodal Deep Boltzmann Machine

When we train both specific Deep Boltzmann Machine, we stack them with a high level joint layer and form a Multimodal Deep Boltzmann Machine.

3. Training Deep Boltzmann Machine

In this section, we will show how do we train an RBM using CD-k method, and which consists in the simultaneous sampling of sets ('blocks') of nodes. This methode is described in the thesis [1].

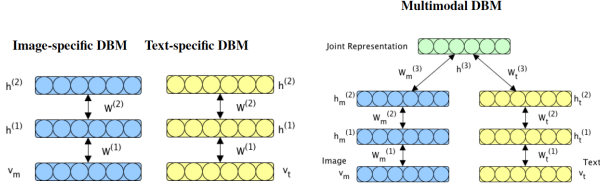


Figure 2: Architecture of the Multimodal Deep Boltzmann Machine (Figure from [4])

3.1. Unsupervised pretraining

First, we do an unsupervised pretraining on each of the specific Deep Boltzmann Machine in order to initialize the weights of the multimodal Deep Boltzmann Machine. To do so, we follow these steps:

1. **Stochastic Bottom-up pass:** In the first step, we adjust the top-down weights to be good at reconstructing the feature activities in the layer below.
2. **Gibbs Sampling:** Then we do a few iterations of sampling in the top level RBM. This is to adjust the weights in the top-level RBM.
3. **Stochastic top-down pass:** We adjust the bottom-up weights to be good at reconstructing the feature activities in the layer above.

3.2. Block-Gibbs sampling

Block gibbs MCMC is possible thanks to the special structure of the RBM. As we have for an RBM the following equation:

$$P(h|v; \theta) = \prod_{j=1}^F p(h_j|v) \quad , \quad P(v|h; \theta) = \prod_{i=1}^D p(v_i|h)$$

All hidden variables are sampled in a single iteration. Then all observed variables are sampled in a single iteration. The training does this operation k times. Usually $k = 1$. In Contrastive Divergence (CD) method, we initialize MCMC chain from data points.

4. Data

4.1. Image data

The data consists of 1 million images retrieved from the Flickr website along with user assigned tags. There are 975K images that are unlabelled and used for pretraining. And 25K labelled images, among which 10K are used for training, 5K for validation and 10K for testing.

Images are represented by a 3,857-dimensional features, that were extracted by concatenating Pyramid Histogram of Words (PHOW) features, Gist, and MPEG-7 descriptors.

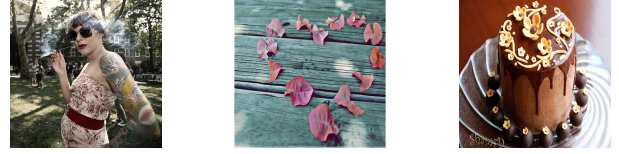


Figure 3: Set of 3 images present in the dataset.

Table 1: Distribution of the MIR Flickr Dataset

Unlabelled	Labelled		
Pretraining	Training	Validation	Testing
975K	10K	5K	10K

4.2. Text data

Espagna, mountain, bird, travel..

The text input is vector of 2,000 most frequent tags. Each image is associated with a vector that indicates its associated tags. There is however images that do not have tags. Over the labelled dataset of images, 4,551 images are without tags. In section 7.4, we infer to those images tags by sampling from the joint hidden layer of the Multi-DBM.

5. Architecture Details

5.1. Pipeline

The image pathway consists of a Gaussian RBM with 3,857 linear visible units and 1024 hidden units. And is followed by 1024 hidden units. The text pathway consists of a Replicated Softmax Model with 2000 visible units and 1024 hidden units followed by another 1024 hidden units. The joint layer contains 2048 hidden units. All hidden units are binary.

5.2. Inferring task

Once we learned the weights of the Multinodal Deep Boltzmann Machine, we can perform different tasks by sampling from the joint hidden layer given the input. We can infer images from text thereby doing image retrieval, or infer words from image thereby doing image annotation. We can also perform unimodal queries like image search from image or text retrieval from text. Each of these two last tasks could be performed using only unimodal Deep Boltzmann Machine. However, in the paper [4], it is shown that unimodal queries in a multimodal DBM perform better than in the unimodal DBM. The Mean Average Precision improves for unimodal queries from **0.469** to **0.531** in the image search task.

It is also shown that **pretraining** with the unlabelled dataset improves a lot the performance of classification. It improves the MAP of the classification from **0.526** to **0.585**.

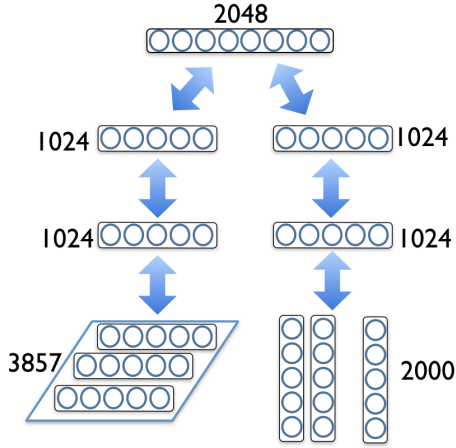


Figure 4: Architecture of the Multimodal Deep Boltzmann Machine. From [4] presentation.

6. Tools

6.1. Code

The results obtained in the paper [4] was done using python. We wanted to reproduce the results obtained in the paper with a code done in Tensorflow. So we built a code on the top of an available code for deep learning that you can find in this github repository¹. I then implemented a specific text Deep Boltzmann Machine with two layers and a specific image Deep Boltzmann Machine with two layers. Then I joined the two models with a joint binary hidden layer in order to build the multimodal deep boltzmann machine.

6.2. Structure of the code

- **config.py** a python configuration file that store the directory of the data and the directory to store the learned model.
- **rbm.py** a python class that implements an RBM. It supports the binary and gaussian type of RBMs. We added to it another type, in order to include the Replicated Softmax Model of RBMs.
- **dbm.py** a python class that implements a Deep Boltzmann Machine. It is implemented as stacked RBMs.
- **run_flickr_text_dbm.ipynb** is the python script that trains the text specific Deep Boltzmann Machine. This script outputs the **text_dbm_layer_i.W.npy** and **text_dbm_layer_i.b.npy** learned weights and bias of the text DBM, with $i \in \{1, 2\}$ as we use two layers.

- **run_flickr_image_dbm.ipynb** is the python script that train the image specific Deep Boltzmann Machine. This script outputs the **image_dbm_layer_i.W.npy** and **image_dbm_layer_i.b.npy** learned weights and bias of the image DBM, with $i \in \{1, 2\}$.
- **run_flickr_multimodal_dbm.ipynb** is the python script that implements the multimodal Deep Boltzmann Machine using the weights learned from the two precedent scripts.

You can find these scripts in the Amazon Web Service account

6.3. Data

We use in our experiments the data provided by Nitish Srivastava in [4] as described in section 4.

6.4. Platform

The code was deployed on the amazon web service platform. First, we used the **p2.xlarge** instance with 1 GPU, 4 CPU and 61 GiB of RAM. But it was not sufficient enough. So we moved it to the **g2.8xlarge** instance, with 4 GPUs, 32 CPUs and 60 GiB of RAM.

The first is accessible from

<http://35.167.62.134:8888/>

And the second is accessible from

<http://35.167.218.163:8888/>

Both have `project` as password.

In the first, I do experiment with the code described in section 6.1. The running code is available in **Deep-Learning-Tensorflow/command_line** directory.

In the second, I have tested the code of Nitish in order to perform his results. The code is available in the **deepnet/** directory of the notebook.

¹<https://github.com/blackecho/Deep-Learning-TensorFlow>

7. Experiments

To reproduce the results described in the paper [4], the code we write could only reproduce results in text retrieval from text using unimodal Deep Boltzmann Machine. So I used the code of Nitish available in his website. The training of the Multimodal Deep Boltzmann Machine take about one entire day in the **g2.8xlarge** amazon web service instance. And the cost is estimated at **642.65\$** in AWS.

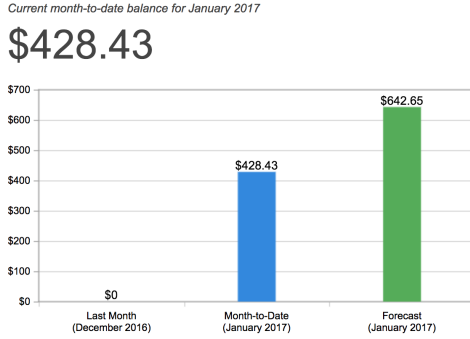


Figure 5: Cost of training a Multimodal Deep Boltzmann Machine an entire day.

7.1. Training and Classification task

We measure the performance of the Multimodal Deep Boltzmann Machine on the classification task on the labelled dataset of images. We measure the Mean Average Precision and the precision@50 for each layer of the Multi-DBM.

Layer	MAP	Prec@50
image_input	0.417 \pm 0.002	0.661 \pm 0.011
image_hidden1	0.476 \pm 0.003	0.755 \pm 0.004
image_hidden2	0.475 \pm 0.003	0.753 \pm 0.008
joint_hidden	0.623 \pm 0.003	0.881 \pm 0.005
text_hidden2	0.512 \pm 0.005	0.832 \pm 0.007
text_hidden1	0.507 \pm 0.003	0.830 \pm 0.008
text_input	0.464 \pm 0.005	0.788 \pm 0.008

We find in our experiment that the MAP of the classification task that Multimodal Deep Boltzmann Machine is **0.623**, which is higher than in the original paper [4] that achieves **0.609** with the same model. It is probably due to the **CUDA** version. In the paper [4], the Multi-DBM was trained on a **CUDA-5.0** version. In our experiment, we use the **CUDA-7.5** available in the **Bitfusion Ubuntu 14 Tensorflow AMI on the AWS account**. We can compare this value with other multimodal classification models, like the semi supervised learning multimodal classification that is performing on average **0.623**, see [3]. Other model reaches less efficient MAP, like the multimodal Deep Belief Network that achieves a MAP of **0.599** [4].

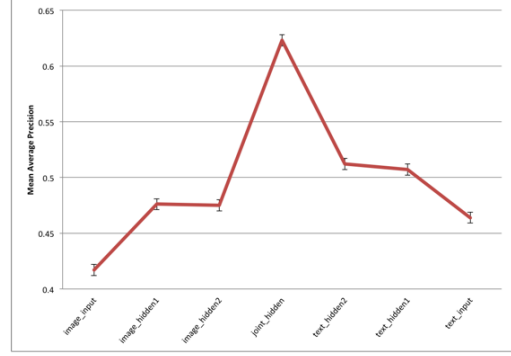





Figure 6: MAP of classification task using representation from different layers of Multimodal DBMs.

7.2. Image annotation

We do the experiment of generating tags from given images.

Table 2: Tags generated from images

Image	Given Tags	Generated Tags
	nikon, nature, macro, flower, water, red, animal, closeup, insect, detail	macro, flower, olympus, 2009, purple, fleur, zuiko, iris, e510
	nature, water, abigfave, landscape, reflection, mirror	sky, ciel, landscape, paysage, soleil, olympus, cloud, france, hdr, montagne
	night, france, europe, paris, lumix, traffic, nuit	france, paris, lumix, panasonic, francia, rue, french, nuit, ville

The generated tags contain information that we do not really need to know in image annotation, like the date or the type of the camera used. But this is due to the bias on the most frequent tags on the image we use in our model. In the MIR Flickr data set, the 10 most frequent tags are:

7.3. Image retrieval

We can do the opposite task, which is generating images from a given tags.

7.4. Inferring missing annotation

In this experiment, we infer tags to images that have no annotations on them. In the labelled dataset, there are 4,551 images without any annotations. By sampling from the joint hidden layer, we can infer to those images tags that have

Table 3: Top 10 most frequent tags in the MIR Flickr image Data Set

1	Nikon	45950
2	Canon	43308
3	Nature	40284
4	2008	39955
5	sky	33331
6	blue	31930
7	macro	30290
8	bw	30290
9	flower	29564
10	water	28683

Table 4: Images retrieved from text













Tags	Images Retrieved		
france, paris			
sky, sunset, landscape, mountain, cloud			
flower, canon, japan			

Table 5: Inferring tags on images without annotations

Images	Generated tags
	explore, interesting, germany, olympus, glass, england, berlin
	colourartaward, olympus, moon, germany, interesting, shadow, blueribbonwinner, sweden, city, dark, europe, abstract, summer
	moon, flickr, cold, rainbow, photography, digital, england, stars, space, dark, castle, summer, sunrise


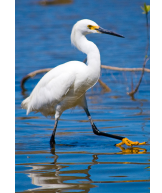
high probability. Here are some sampled annotations for those images:

In this task, Multimodal Deep Boltzmann Machine perform poorly because of their high bias that exists in the inferring task. We notice that even if the figure is already learned by the modal, the inferring task is not efficient in recognizing the learned object. Like for example, recognizing a man that it has first seen

7.5. Errors

In the annotation task, sometimes it outputs false tags in particular if the image is not correctly annotated or their are missing information about it. Here are some examples:

Table 6: False tags generated on images

Image	Given Tags	Generated Tags
	d80	sea, beach, ocean, picnic, coast, mar, island, pacific, waves, surf
	california, birds, sandiego	obama, election, politics, president, hope, change

8. Conclusion

Boltzmann Machines introduced by Geoffrey Hinton in the 90s were invented first in order to do representation of complexe features. They can be efficient in in learning features like the MNIST data [2]. But they are not efficient in learning complexe features like objects in images. This is due mainly on the computational cost of the training procedure of them. Instead we use Convolutional Neural Networks in order to perform these tasks.

The MIR Flickr dataset suffer from high bias present in the training dataset, which leads to false annotations.

Due to limited budget and the computation cost of the Multimodal Deep Boltzmann Machine, I couldn't do more experiments than I did in order to compare different models or try on different image-text datasets like the MSCOCO or Pascal VOC 07 datasets.

References

- [1] I. C. Fischer, A. Training restricted boltzmann machines: an introduction. *Pattern Recognition*, 47(1):25–39, 2014.
- [2] S. O. Geoffrey E. Hinton and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation* 18, 2006.
- [3] J. V. Matthieu Guillaumin and C. Schmid. Multimodal semi-supervised learning for image classification. *CVPR*, 2010.
- [4] R. S. Nitish Srivastava. Multimodal learning with deep boltzmann machines. *NIPS*, 2012.
- [5] G. H. Ruslan Salakhutdinov. Deep boltzmann machines. *JMLR*, 2009.