UGP Report - CS395A

Shubham Jain
Roll No. : 13683

Supervisor: **Dr. Vinay Namboodiri**

Department of Computer Science
Indian Institute of Technology Kanpur

November 8, 2015

1 Introduction

The student was introduced to the topics in Machine Learning and Deep Learning for working in the future. The goal was to extend the Deep Lambertian Networks by using other Reflectance Model like Phong's reflection model that the supervisor intend to carry out in the coming months.

2 Duration and structure

Initial reading began in the summer of 2015. More serious work was done during the first semester, 2015-2016. The student met with his supervisor once in a week for at least an hour to discuss the progress and work.
Since the topics was very wide, their level of difficulty compared to the introductory course/topics the students was previously exposed to, the reading was not very well oraganized and hence various sources was explored

3 Reading List

Note: The material used to study various topics are cited after the topic. They were used as a reference.

- Neural Networks [4, 3, 9] : Perceptrons, Back Propagation, Convolution Neural Networks, Recurrent Neural Networks, Momentum Method.
- Machine learning [3, 2, 8]: Regularization, Bayesian statistics, Support Vector Machines, Principal Component analysis, Fitting probability models.
- Deep Learning and Computer Vision [3, 8, 5]: Feed forward Deep Networks and its optimization techniques, Feature Learning, Regularization from bayesian perspective, Sparse Representations, Optimizations algorithms like Nestrov Momentum, BFGS etc., Skin detection
- Deep Lambertian Networks [1, 6, 7]: Gaussian Restricted Boltzmann Machines, Markov Chain Monte Carlo Techniques, Gibbs Sampling, Contrastive Divergence
- Torch [10, 11]

4 Topics and Background

We describe some of the topics that are were studied during the course very briefly

4.1 Feed Forward Deep Networks

They are also known as *multilayer perceptrons* (MLPs). They are functions which are formed by compositions of many other parametric functions. Each sub-function is called a layer and the output of these functions are referred as feature. The connection between units do not form a directed cycle that is to say that information travels in a single direction. In these networks information moves through hidden layers to the output layer. The functions can be choosen from a varied range from simpler functions like linear to complex functions like Tanh function and so on. Training these networks requires a good deal of optimization techniques and traning procedures (like Backpropagation) as well as regularization techiniques so that the model can fit on general data. It can be shown that Multilayer Perceptron can represent any function but learning that can be very hard.

4.2 Regularization

While training a machine, we need that the algorithm preforms well not on just the training data but also on the new input data. These techniques are called regularization and are based on the trade off between bias for reduced variance. The regularized cost function J', with an initial cost J will look like

$$J'(\theta; X, y) = J(\theta; X, y) + \alpha\Omega(\theta) \tag{1}$$

where $\Omega(\theta)$ is the norm penalty and α is the hyperparameter that weighs the contribution of the term. Some of the regularization that are useful in most of the applications are:

- L^2 **Parameter Regularization:** $\Omega(\theta) = \frac{\|\theta\|^2}{2}$.
  Also called as ridge regression and is the most frequently used regularizer. Minimizing this error is functionally equivalent to maximizing the log of posterior distribution where we have a Gaussian distribution as a prior.
- L^1 **Parameter Regularization:** $\Omega(\theta) = \|\theta\|$.
  In comparision to L^2 regularization, L^1 regularization results in a sparse solution (there exists free parameters of the model with zero as an optimal value. Minimizing this error is functionally equivalent to maximizing the log of posterior distribution where we have a Laplace distribution as a prior.

Other type of regularizations like adding noise at the weights, at the input, early stopping etc. are also important when the required funtion is needed to have a certain property.

4.3 Optimization Techniques

The most difficult optimization problem is traning the neural networks. For a given input x, target y and some loss function L(x,y) we want to minimize the Emprical

risk which is defined as:

$$E_{x,y\sim\hat{p}(x,y)}[L(f(x;\theta),y] = \frac{1}{m}\sum_{i=1}^{m} L(f(x^{(i)};\theta),y^{(i)}) \tag{2}$$

This is known to be prone to overfitting.
Several challenges are faced in optimization like ill conditioning, saddle points that are to be taken into consideration. The techniques that are found to be work in general are Gradient descent, Momentum techniques, adaptive learning rates etc but they are first order techiniques. At times approximate second order techniques like BFGS and Conjugate methods are favoured.

4.4 Convolutional neural network

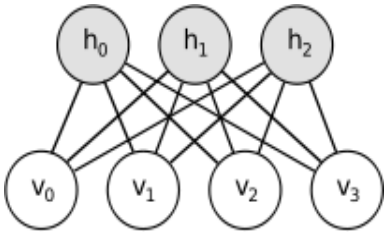
They are a type of feed forward artificial neural network that are used to process data that has a known grid like structure and hance they are widely used for image recognition. "Convolutional networks are simply neural networks that use convolution in place ofgeneral matrix multiplication in at least one of their layers."[1] The **convolution** operation for two function f and g is defined as:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(a)g(t-a)da \tag{3}$$

Convolutuion is advantageous because of: *sparse interactions, parameter sharing and equivariant representations.* Often pooling is used which replaces the output at a certain location with a summary of nearby outputs. Pooling provides an important property if invarinace with respect to local transition which is very beneficial for image processing.

4.5 Restricted Boltzmann machines

They are **energy based** model with hidden units but without visible-visible and hidden-hidden connections. A graphical depiction of an RBM is shown below.



The energy function E(v,h) of an RBM is defined as:

$$E(v,h) = -b'v - c'h - h'Wv \tag{4}$$

[1] Yoshua Bengio and Ian J. Goodfellow and Aaron Courville, *Deep Learning*, 2015

where W represents the weights connecting hidden and visible units and b, c are the offsets of the visible and hidden layers respectively.
This leads to the free energy formula:

$$\mathcal{F}(v) = -b'v - \sum_i \log \sum_{h_i} e^{h_i(c_i + W_i v)}. \tag{5}$$

Because of the structure of RBMs, hidden units are conditionally independent given the other. We can arrive at these equrations by some manipulations:

$$p(h|v) = \prod_i p(h_i|v) \tag{6}$$

$$p(v|h) = \prod_j p(v_j|h). \tag{7}$$

We require binary units, hence the probability of a particular unit on is can be found out as:

$$P(h_i = 1|v) = sigm(c_i + W_i v) \tag{8}$$

$$P(v_j = 1|h) = sigm(b_j + W_j' h) \tag{9}$$

We can now find the update equations by find the log-likelihood gradients.
For Deep Lambertian Networks we require Gaussian RBM which are described next

4.6 Gaussian Restricted Boltzmann machine (GRBMs)

They use Gaussian units in the visible layer of RBMs.“ GRBMs can be viewed as a mixture of diagonal Gaussians with shared parameters, where the number of mixture components is exponential in the number of hidden nodes“[2]. With visible nodes v $\in$ $\mathbb{R}^{\mathbb{N}}\approx$ and hidden nodes h $\in 0, 1^{N_h}$, the energy of the joint configuration is given by:

$$E_{GRBM}(v, h) = \frac{1}{2} \sum_i \frac{(v_i - b_i)^2}{{\sigma_i}^2} \tag{10}$$

The conditional distributions needed for inference and generation are given by:

$$P(h_i = 1|v) = \frac{1}{1 + exp(-\sum_i W_{ij} v_i - c_j)} \tag{11}$$

$$P(v_j = 1|h) = \mathcal{N}(v_i|\mu_i, {\sigma_i}^2) \tag{12}$$

where $\mu_i = b_i + {\sigma_i}^2 \sum_j W_{ij} h_j$. Binary RBMs are often stacked on top of additional layers of a GRBM to form a Deep Belief Net (DBN)[12]. “Inference in a DBN is approximate but efficient, where the probability of the higher layer states is a function of the lower layer states.“[2]

[2]Tang, Yichuan, Ruslan Salakhutdinov, and Geoffrey Hinton. *Deep lambertian networks.* arXiv preprint arXiv:1206.6445 (2012).

4.7 Markov Chain Monte Carlo Techniques

4.7.1 Markov random field (MRF)

They are a set of random variables that follow the "Markov property if the joint probability distribution p fulfills the (global) Markov property w.r.t. the graph:
For all disjunct subsets A, B, $S \subset V$, where all nodes in A and B are separated by S the variables $(X_a)_{a\in A}$ and $(X_b)_{b\in B}$ are conditional independent given $(X_S)_{s\in S}$, i.e. for all $x \in \lambda^{\|V\|}$ it holds $p((x_a)_{a\in A}|(x_t)_{t\in S\cup B}) = p((x_a)_{a\in A}|(x_t)_{t\in S})$."[3]

In RBM traning Markov chains play a crucial role by presenting method that all to draw complex probability distributions. We will now define Markov Chain:
"A markov chain is a random process that undergoes transitions from one state to another on a state space. It must possess a property that is usually characterized as "memorylessness": the probability distribution of the next state depends only on the current state and not on the sequence of events that preceded it"[4]
Mathematically this can be viewed as a set of rv's X = $X^(k)|k \in \mathbb{N}$ for which takes value in a set Ω and $\forall k/geq0$ and $\forall j, i, i_0, ..., i_{k-1} \in \Omega$ it holds:

$$\begin{aligned} {p_{ij}}^{(k)} &= P\Big(X^{(k+1)} = j|X^{(k)} = i, X^{(k-1)} = i_{k-1}, ..., X^{(0)} = i_0\Big) \\ &= P\Big(X^{(k+1)} = j|X^{(k)} = i\Big) \end{aligned} \tag{13}$$

Starting from the probability distribution μ^0 (of X^0), the distribution of μ^k (of X^k) is given by $\mu^{(k)T} = \mu^{(0)T}P^k$.

A distribution A is said to be a *stationary distribution* if $A^T = A^TP$. So, if Markov chain reaches a stationary distribution then allsubsequent states will be distributed accordingly. Markov chains which have a unique stationary distribution are especially relevant. Markov vhain Monte Carlo methods use these convergence methods to produce samples of certain probability dist. by setting a Markov chain that converge to desired distributions. Gibbs Sampling is such a MCMC method and will be described in the following section.

4.8 Gibbs Sampling

Gibbs Sampling belongs to the class of Metropolis-Hastings algorithms. It is used to generate a sequence of observations approximated from a multivarirate probablity distribution from which sampling direct is hard. The idea is to update the variables based on its conditional distribution given the others state. Gibs sampling is descrived in detail now:
"We consider an MRF X = $(X_1, ..., X_N)$ w.r.t. a graph G = (V,E), where V = 1,...,N for the sake of clearness of notation. The random variables X_i, i $\in$ V take values in a finite set Λ and $\pi(x) = \frac{1}{Z}e^{-E(x)}$ is the joint probability distribution of X. Furthermore, if we assume that the MRF changes its state during time, we can

[3] Tang, Yichuan, Ruslan Salakhutdinov, and Geoffrey Hinton. *Deep lambertian networks.* arXiv preprint arXiv:1206.6445 (2012).
[4] Taken from Wikipedia.org

consider X = $X^{(k)}$—k ∈ $\mathbb{N}_{\nvDash}$ as a Markov chain taking values in $\Omega = \Lambda^N$ where $X^{(k)} = ({X_1}^{(k)}, ..., {X_N}^{(k)})$ describes the state of the MRF at time k 0. At each transition step we now pick a random variable X_i, i ∈ V with a probability q(i) given by a strictly positive probability distribution q on V and sample a new value for X_i based on its conditional probability distribution given the state $(x_v)_{v \in V_1}$ of all other variables $(X_v)_{v \in V_1}$, i.e. based on $\pi(X_i|(x_v)_{v \in V_1} = \pi(X_i|(x_w)_{w \in \mathcal{N}_i}$ Therefore, the transition probability p_{xy} for two states x, y of the MRF X with x ≠ y is given by:

$$\begin{cases} q(i)\pi(y_i|(x_v)_{v \in V \setminus i}) & if \exists i \in V \text{ so that } \forall v \in V \text{ with } v \neq i : x_v = y_v \\ 0 & \text{else} \end{cases}$$

And the probability, that the state of the MRF x stays the same, is given by:

$$p_{xx} = \sum_{i \in V} q(i)\pi(x_i|(x_v)_{v \in V \setminus i}) \tag{14}$$

It is easy to see that the joint distribution π of the MRF is the stationary distribution of the Markov chain defined by these transition probabilities "[5]
It can be show that X_i can take every possible state in a single transition step and the every state of the MRF can reach any other in finite number of steps and the Markov chain in irreducible. IN practice the rv's are usually choosen in a predefined order. It can be shown that the convergence rate of Gibbs sampler is bounded.

4.9 Contrastive Divergence

It is a learning method which can obtain sufficient estimates for model training after running the Markov chain for a few number of steps. The main idea of contrastive divergence learning (CD-k) is that it doesn't wait for the chain top converge. Samples are obatined after running k-steps of Gibbs sampling and in practice k=1 has been working very well. Also, the chain in initialized with a training example which essentially means that we expect the distribution to be already close of final distribution. Each step t consists of sampling h from p(h—v^t) and sampling v and p(v—h^t). The gradient w.r.t. θ of the log-likelihood for one training pattern $v^{(0)}$ is then approximated by

$$CD_k(\theta, v^{(0)}) = -\sum_h p(h|v^{(0)}) \frac{\partial E(v(0), h)}{\partial \theta} + \sum_h p(h|v^{(k)}) \frac{\partial E(v(k), h)}{\partial \theta} \tag{15}$$

4.10 Lambertian Reflectance

The linear Lambertian model is

$$I(\vec{x}\,) = a(\vec{x}\,)\vec{n}\,(\vec{x}\,) \cdot \vec{s} \tag{16}$$

where image is represented by $I(\vec{x}\,)$, albedo by $a(\vec{x}\,)$, surface normal by $\vec{n}\,(\vec{x}\,)$, light source $\vec{s}$. The illumination strength has the magnitude ($|\vec{s}|$ and $\vec{s}/|\vec{s}|$ represents the light source direction. The brightness of a Lambertian surface to an observer is same regardless of the observer's angle of view.

[5]Fischer, Asja, and Christian Igel. "An introduction to restricted Boltzmann machines." Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications (2012): 14-36.

5 Current Work

After getting a good understanding of these underlying concept we are working on first understanding their usage in the paper "Deep Lambertian Network"[1] and then implementing the techniques used in the paper to get our own result on the model. We want to make the learning less supervised as possible.

6 Future Research Work

We will be continuing the project in December and January and where we will implement Deep Lambertian Networks on the dataset available and then changing the reflectance model to Phong's reflection model and analyzing and comparing the results obtained by the two of them. The target would be to achieve some publishable material for the top conferences by Feburary. After completion of this project, the student aim to work on deeper topics in Comupter Vision and Machine Learning in the next semesters.

References

[1] Tang, Yichuan, Ruslan Salakhutdinov, and Geoffrey Hinton. *Deep lambertian networks.* arXiv preprint arXiv:1206.6445 (2012).

[2] Andrew Ng, *Machine Learning*, Coursera, 2013

[3] Yoshua Bengio and Ian J. Goodfellow and Aaron Courville, *Deep Learning*, 2015

[4] Geoffrey Hinton, *Neural Networks for Machine Learning*, Coursera, 2012

[5] CILVR Lab, NYU. *Material for the Deep Learning Course*. 2012

[6] Fischer, Asja, and Christian Igel. "An introduction to restricted Boltzmann machines." Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications (2012): 14-36.

[7] Restricted Boltzmann Machines (RBM) DeepLearning 0.1

[8] Prince, Simon J. D. Computer Vision. New York: Cambridge University Press, 2012.

[9] Unsupervised Feature Learning and Deep Learning Tutorial, *Convolutional Neural Network*

[10] "www.torch.ch" *Torch*, Scientific computing for LuaJIT.

[11] "https://github.com/torch/torch7" *Torch 7*

[12] Hinton, G. E., Osindero, S., and Teh, Y. W. A fast learning algorithm for deep belief nets. Neural Computation, 18 (7):15271554, 2006.