# Investigation of Connection Between Deep Learning and Probabilistic Graphical Models

by

Paul Andrew Hager

S.B., Massachusetts Institute of Technology (2016)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2018

Author . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Department of Electrical Engineering and Computer Science
February 1, 2018

Certified by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Devavrat Shah
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Christopher Terman
Chairman, Master of Engineering Thesis Committee

# Investigation of Connection Between Deep Learning and Probabilistic Graphical Models

by

Paul Andrew Hager

Submitted to the Department of Electrical Engineering and Computer Science
on February 1, 2018, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

## Abstract

The field of machine learning (ML) has benefitted greatly from its relationship with the field of classical statistics. In support of that continued expansion, the following proposes an alternative perspective at the link between these fields. The link focuses on probabilistic graphical models in the context of reinforcement learning. Viewing certain algorithms as reinforcement learning gives one an ability to map ML concepts to statistics problems. Training a multi-layer nonlinear perceptron algorithm is equivalent to structure learning problems in probabilistic graphical models (PGMs). The technique of boosting weak rules into an ensemble is weighted sampling. Finally regularizing neural networks using the dropout technique is conditioning on certain observations in PGMs.

Thesis Supervisor: Devavrat Shah
Title: Professor of Electrical Engineering and Computer Science

## Acknowledgments

First, I would like to thank my advisor, Professor Devavrat Shah, for his continued guidance and wisdom. Second, the staff of the Massachusetts Institute of Technology, the Electrical Engineering and Computer Science Department, and especially Dr. Katrina LaCurts. Many of them have been instrumental during the time this work was composed. Third, Iris Fung and Benjamin Bloomberg must be thanked for helping me see this thesis through to an end. Fourth, this work would not be possible without the support of my family and group of friends.

# Contents

# Chapter 1

# Learning

## 1.1 Motivation

Machine learning is enjoying a renaissance of interest. It is more important than ever to draw parallels between the fields of machine learning and statistics, as connections between disparate fields often lead to exciting discoveries.

Despite this fact, one must first clarify boundaries between the fields. One difference between the two fields that has been suggested is their purposes. Statistics is the study of the collection, analysis, interpretation, presentation, and organization of data. Therefore, to extend statistics is to better any of those facets. One extension could develop a method which allows an easier interpretation of information.

Machine learning is the study and construction of algorithms that make predictions on data. At first glance these two fields seem related but distinct. However this idea is incorrect. The two are dependent fields and are intimately connected.

When one can interpret data perfectly one understands the mechanism or system which creates it. In turn, if one can predict how a system evolves one can functionally understand it.

## 1.2 Problems in Interpretation

A statistical study often proceeds according to the scientific method, many cycles of subsets of the following procedures:

- Observing phenomena
- Recording measurements
- Forming hypotheses
- Inferring predictions
- Compare predictions and measurements

In statistics, the phenomena studied are not limited to any particular scientific discipline or field. Proposed hypotheses are models by which the observations, sometimes called variables or predictors, could have been generated. The simplest relationship often used between the predictors is an affine transformation.

A more complete treatise on standard statistical methods can be found in [1].

## 1.3 Problems in Prediction

Many subfields of machine learning (ML) are studying how to make structured predictions given an input. The simplest of these tasks is binary classification, where given an input one makes a decision between two targets. Binary classification can be extended to its more general form multinomial classification, where the target set's cardinality is only constrained to be finite. When the target set's cardinality extends beyond the finite, we rename the task, regression. However, it is important to notice the intent is still to make predictions, just onto a different range of values.

A longer, more complete treatment of a number of problems and topics in ML can be found in [3].

## 1.4 Enviornments

The environment for a reinforcement learning (RL) task can be described by $B' = (S, O, A, P, Q, R)$. $S$ is a multi-dimensional set of states, which may be have a finite or countably infinite support. $O$ be a multi-dimensional observation set, which again may have a finite or countably infinite support Notably, it could also be a subset of $S$. Given a state, $s$, $\mathbb{A}(s)$ is a space of available actions, again this can an arbitrary set but for the sake of convenience let, $A = \cup_s \mathbb{A}(s)$. $P$ represents the probability distribution of transitioning between states. $Q$ represents the probability distribution of outputting and observation. $R$ is a function, $R : (S \times A \times S) \to \mathbb{R}$, which maps a previous state, a new state, and an action to a real number. One with background in decision theory can recognize the similarity to a Markov Decision Process (MDP). For a brief introduction to MDPs, succinctly introduced in [2].

For the remainder of the work, we consider a reduced space of processes where the following conditions are assumed.

- Given a state, $s_i$, and action, $a$, transitions between states are deterministic, thus $P_{s_i,a,s_j} = 1$ and $P_{s_i,b,s_k} = 0 \quad \forall (b, s_k) \neq (a, s_j)$.
- Given a state, $s_i$, and observation, $o$ are deterministic, thus $Q_{s_i,o_j} = 1$ and $Q_{s_i,o_k} = 0 \quad \forall s_k \neq s_j$.

Under these assumptions, we can fully characterize the RL environment as $B = (S, O, A, R)$.

Furthermore, the learning and prediction tasks outlined in 1.3 are other special cases of the framework, $B'$. For example, classification can be described as an environment where the input is the state $s_i$ and the binary classes are the actions available to the agent at every state. When explicit targets are unavailable, as is the case after selecting an action, the system must wait and proceed until it receives feedback. One could then think of actions of ordered collections of predictions.

In the game theory field, one could think of this as the state of the game, or the current node of the game tree. For notational convenience, we will consider that some latent space, $S$, represents the true state. The information available to us is available to the player / agent is only some subset of that $S$ or some $f(S) = O$. This $O$ is available at each time when a decision is demanded of the agent. Often these are called information sets in the game theory literature, one could also consider this as a just some observations of the underlying state. A simple structure to make this concept clear is a hidden Markov model where:

1. a state $S_i$ is conditionally independent of all states, $S_0, S_1, \cdots S_{i-2}$ given $S_{i-1}$.
2. a state $S_i$ is conditionally independent of all states, $S_{i+2}, S_{i+3}, \cdots$ given $S_{i+1}$.
3. an observation $O_i$ are conditionally independent of all $S_j \,|\, j \neq i$ given $S_i$.

For some intuition about the meaning of conditionally independent used here see section 2.2.4.

One may consider a general idea of this task where states may be a function of all previous states, observations and actions. Thus the true states and observations take the forms,

$$S_i(S_{i-1,\cdots,0}, O_{i-1,\cdots,0}, a_{i-1,\cdots,0})$$
$$O_i(S_{i,\cdots,0}, O_{i-1,\cdots,0}, a_{i-1,\cdots,0}).$$

We can summarize this information in a history, $h(i, G, A)$, of an agent interacting with an environment up to point $i$ where the graph $G$ describes the conditional relationships of all states and observations and $A$ are an agents' sequence of actions. One can now see a history $h$ partially satisfies an RL environment, $B$ with only the function $R$ removed. An appropriately general $R$ remains an open question, however please grant for all further discussions $R$ will be specified.

# Chapter 2

# Graphical Models

This chapter provides a brief introduction to graphs. It extends further into using graphs as representations of joint probability distributions. Those who have previously studied graphs should proceed to 2.2.4.

## 2.1 Graph Theory

For the following discussion, let a graph, $G$, consist of two objects $(V, E)$. $V$ is a set of nodes, one can think of it as an arbitrary collection of items. The letter V is used because nodes are commonly referred to as vertices in the graph theory community and the terms will be used interchangeably.

### 2.1.1 Directed Graphs

A more principled introduction to graph theory can be found in [5]. Despite it being taught motivated by a different set of problems. This text refers to *digraphs*, the following discussion will refer to these objects as *directed graphs*. $E$ is a set of pairs of nodes, $e = (v_i, v_j)$ where $v_i, v_j \in V$. These pairs may therefore represent arbitrary connections between nodes.

Note, the above definitions do not prevent self loops, $e_{i,i}$, however for much of the following discussion graphs with self-loops will not be considered.

### 2.1.2 Undirected Graphs

An undirected graph differs from those defined in 2.1.1 by one property. An edge, $e$ in an undirected graph does not communicate an ordering onto its incident nodes. Therefore, an undirected graph's edge set, $E$, contains edges, $e = \{v_i, v_j\}$, where $v_i, v_j \in V$.

### 2.1.3 Definitions

A *walk* between nodes $v_i$ and $v_j$ is a sequence of edges, $(e_1, e_2, \cdots, e_{n-1})$, such that $e_k = (v_{k-1}, v_k)$ and

$$e_{k+1} = (v_k, v_{k+1}) \,\forall\, k \,|\, 1 \leq k \leq n-1$$

$$e_1 = (v_i, \bullet), e_{n-1} = (\bullet, v_j).$$

The first condition is an edge in the walk must be directed towards a node from which the next edge is pointed away. The second condition is the first edge must point away from the initial node and the last edge must point towards the final node. There are special walks where the first and last edges are incident to the same node; these are called *cycles*.

Often it can be informative to restrict one's thinking to walks which do not include an edge multiple times. This idea is formalized and often called a path in $G$ between $v_i$ and $v_j$. Paths can be thought of as walks with distinct edges. A *path* exists between nodes $v_i$ and $v_j$ if there is a walk between these nodes, $v_i, v_j$ and

$$e_k \neq e_l \;|\; \forall\; k, l \;|\; 1 \leq k, l \leq n \;|\; k \neq l$$

## 2.2 Probabilistic Graphical Models

The main idea of this section is to detail how graphical structures are used to describe joint probability distributions.

A node represents a random variable and an edge denotes a relationship, potentially a dependency, between two variables. Note, a lack of an edge will not mean independence between two variables. The two non-adjacent variables will for some set of observed variables be independent. This idea allows us to encode a family of distributions on $n$ variables, for which every probability distribution is a member, in a complete undirected graph of $n$ nodes. This representation is not necessarily efficient as others but from its formulation below you can see it captures all possible relationships. Such a graph that represents a probability distribution will be referred to as a probabilistic graphical model (PGM). The flexibility is possible because an edge does not imply a dependency simply that a dependency is possible.

### 2.2.1 Undirected PGMs

Although an undirected PGM was introduced in the beginning of this section, it is worth examining the exact structure proposed. Consider again the graph on $n$ nodes. The family of joint probability distributions described are any that adhere to the following form:

$$p\left(x_1, x_2, \cdots, x_n\right) = \prod_{i=1}^{n} p(x_i | \mathbb{N}(x_i))$$

where $\mathbb{N}(v_i)$ be the set of adjacent nodes to a node, $v_i$. Note, the $\mathbb{N}(x_i) = V_{/x_i}$ because the graph is fully connected. Therefore one could interpret this family as the set of distributions where every variable is possibly dependent on every variable. This family is universal in that every joint probability distribution on $n$ variables is a case of this distribution with some subset of the variables not being directly dependent. However, this is not a terribly interesting representation of every distribution because, as will be shown, finding the marginal distribution of any one variable requires summing over all other settings for every other variable 2.2.5. Thus, one seeks to find the minimally connected graph which describes the distribution in concern and does not contain any additional dependence relationships.

**Factorization**

There is a theorem relating to the family of probability distributions represented by undirected PGMs, called the *Hammersley-Clifford* Theorem. The meaning of it is that a distribution must be able to factor according to its maximal cliques. Where *maximal clique* is a clique such that addition of another member would eliminate the clique property of the subset of nodes.

### 2.2.2 Directed PGMs

This search for efficient leads one to examine using a directed graph to describe a probability distribution, especially those with the additional condition of being acyclic. For this discussion, the parents of node $x_i$, will be $\pi(x_i)$. Where a parent of $x_i$ is any node with a directed edge leading to the node $x_i$. The probability distribution families described by these graphs are those that satisfy the following form.

$$p\left(x_1, x_2, \cdots, x_n\right) = \prod_{i=1}^{n} p(x_i|\pi(x_i))$$

It is a worthy exercise to construct a directed PGM where the family of distributions is different than the family represented by an undirected PGM with edges between the same nodes.

### 2.2.3 Factor Graphs

The final structure of PGMs introduced will be factor graphs. Factor graphs are efficient in representing joint distributions where clusters of variables all must satisfy a joint condition. Specifically, the family described is

$$p\left(x_1, x_2, \cdots, x_n\right) = \prod_{i=1}^{k} f(\mathbb{C}_i)$$

### 2.2.4 Conditional Independence

When speaking about conditional independence in undirected PGM, one should assume some subset of nodes, $V_c$, have been observed and their nodes and incident edges have been removed from the graph. The variables represented in a connected component, $V_a$, are conditionally independent from any subset, $V_b$, disconnected in the new graph. The concept of graph separation extends to the directed PGM except in a particular case. This case is often referred to as a *V structure* or an *explaining away* of nodes. These names refer to when a child node of multiple parents is observed, the former parents are not conditionally independent as they would be were it an undirected PGM.

### 2.2.5 Marginalization

Finding a marginal distribution for a variable is equivalent to reducing the PGM to only the relevant information about that variable. There are reduction algorithms one can use to squash the uncertainty encoded in different variables into their neighbors of a graph. Thus one can see the appeal and efficiency of using graphs with the minimal number of dependencies possible to describe their relationships.

## 2.3 Estimation Problems

Besides representing known distributions using PGMs, one may also want to observe a phenomenon and create a new model. For such a situation, one must face the following problems according to the information already collected about the phenomenon. They are ordered according to most prior information to least.

1. Given the factorization structure, learn the parameters of the factors.

2. Given the relevant variables and their alphabets, learn the factorization structure (and, in turn, the parameters of the factors).

3. Given the variables, learn their alphabets, and, in turn, their (factorized) joint distribution.

4. Learn the number of variables in the system, their alphabets, and their joint distribution.

These problems of estimation were proposed in [8].

My central claim is deep learning algorithms are attempting to address all four problems simultaneously. Specifically one could view the training of a DNN as a graph selection problem where an edge is only present if the weights after training are greater than some threshold.

# Chapter 3

# Deep Neural Networks

This is a pointer to areas for further exploration by the reader.

## 3.1 Vast Subfield

Deep Neural Networks (DNN) are another name for any multi-layer perceptron algorithm. However, the class of algorithms called DNNs is now a vast subfield of machine learning. One can attribute much of the recent popularity of this subfield to [6]. Although, it is not the singular example of landmark work. An example of an effective training and deployment of an agent to interact with a certain environments can be found in [7] or [9].

## 3.2 Backpropagation

An important concept to understand with regard to DNNs are how they are trained. The training technique is varied and a large topic in itself but a common thread through much of the literature is backpropagation. Generally, backpropagation is the practice of changing the model's parameters in some relation to the observed errors in the current model's predictions. For a brief theoretical introduction please see [4].

## 3.3 Further Work

The field continues active research with many focuses including but not limited to:

- How to efficiently and effectively train a model to make predictions in a new environment
- How environments should be represented by an agent or model
- How to efficiently retain information about an agent's history
- How agents should select actions given their history

# Chapter 4

# Connections and Open Questions

Another perspective is deep neural networks (DNN) are hierarchical potentially non-linear mappings of input features to new features. In the spirit of different perspectives I present a few open questions worth studying.

## 4.1 Backpropagation

Imagine a DNN with a single input node a layer of two hidden nodes and a single output node. Is there a way to display a PGM for this algorithm? One can naively imagine it being represented by a table of size $|A|^{|E_1|^{|E_2|}}$, however perhaps there is a more efficient space representation. Is there also a way to describe backpropagation in a procedural manner relating to this PGM? Ostensibly, the information gathered from a single training point is simply updating the tables to reflect the new information, however inefficient. Perhaps one must build a hierarchical data structure to capture and succinctly update the larger structure.

## 4.2 Boosting

One could alternatively think of this model training technique, boosting, as increasing the weights, edges, of the trees, among a polytree, which correctly meet the targets. However if more efficient one could equally decrease the strength of non-helpful computation graphs.

## 4.3 Dropout Regularization

A common technique in the deep learning subfield is to dropout weights during training. Dropout randomly selects a reduced space of models to be selected. Could the addition of dropout be shown to be equivalent to a randomly quasi-mean field approximation in PGMs?

## 4.4 Marginalization and Attention

It is also worth mentioning PGMs 'attention' is explicitly encoded in the edges present in a graph. When one marginalizes a variable in a PGM, using any algorithm, forward-backward, message passing, or loopy belief propagation (LBP), one encodes all the information of the joint distribution into a single distribution 2.2.5. After marginalizing each variable, it can be easily distinguished which variables are closely related and affect each other.

As a tool for determining relationships in DNNs, the subfield now attempts to maximize the output of a single node and show the values of the input and internal nodes leading up to that node which maximize this output. Could one cast this as a form of marginalizing the node?

## 4.5 Summary

Statistics holds a bright future for all those willing to delve into further study as well as a number of interesting problems to learn more.

# Bibliography

[1] Jennifer Hill Andrew Gelman. *Data Analysis Using Regression and Multilevel/Hierarchical Models.* Cambridge University Press, 2007.

[2] Richard Bellman. A markovian decision process. *Indiana Univ. Math. J.*, 6(4):679–684, 1957.

[3] C Bishop. *Pattern Recognition and Machine Learning.* Springer, New York, 2 edition, 2006. Information Science and Statistics.

[4] Robert Hecht-Nielsen. *Neural Networks for Perception*, chapter 3, pages 65–93. Academic Press, 1992.

[5] R. M. Wilson J. H. van Lint. *A Course in Combinatorics.* Cambridge University Press, 2001.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02 2015.

[8] Devavrat Shah. 6.438 *Algorithms for Inference.* `https://ocw.mit.edu`. License: Creative Commons BY-NC-SA., Fall 2014.

[9] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 01 2016.