1 Introduction

Stacked de-noising autoencoders are a type of deep belief network that can discover arbitrarily deeply nested hierarchical structure in data, loosely based on a hypothetical organization of mammalian neocortex. Geoffrey Hinton et al. have shown that adding another layer to the stack of autoencoders always increases the upper bound on prediction accuracy in machine perception tasks. An autoencoder is a neural network that learns the identity function for given data in a space. Either using fewer hidden nodes than input nodes, or enforcing sparsity causes the network to learn a compressed representation of the input space. Autoencoders are a type of content addressable memory. Autoencoders can be stacked to improve their prediction accuracy. Once a single level auto-encoder is trained, it's hidden node activation levels are used as the input space for another autoencoder. With the addition of the crucial de-noising modification, these stacks of autoencoders become capable of modeling very complex data, and show comparable performance to other deep belief networks (DBN).

2 Reason

Stacked de-noising auto-encoders, and other deep belief networks show much promise in machine perception, they are relatively new, and have a wide range of applicability to AI problems. They may not be an optimal allocation of computational resources in all cases but they show surprisingly human qualities. I want to apply them to planning, because they are powerful and could perform well. Some planning tasks require a sustainable outlook, and many traditional planning algorithms are too greedy. Additionally, in games which end, and for games with an infinite payoff for the winner, and in games in which you can change the rules, a strategic balance between short term and long term solutions must be found. You know how a sustainable strategy can beat a greedy one, but alternatively, a player who uses an exploitative short term tactic can beat a player who sticks to sustainable strategies, because he may win, or change the rules of the game before the time comes to suffer the consequences. An agent who can chain together short periods of exploitation punctuated by changes to the game's rules can achieve an impossibly high rate of growth to an agent who simply simply acts sustainably within the current rules. This type of play would require generating nested plans, and in order to navigate this vasty space, the levels of planning would need to be segmented into say, high level strategy, and low level tactics. A deep belief network planning by "optimistic dreaming", as I describe below, would structure it's model of the possible actions in the game in just this way. Just to be clear, it would not be possible to literally "change the rules" but sub-systems within the DBN may be operating in a sub-game invented by the DBN with more restricted rules than the super-game the whole system is playing.

3 Method

The task of planning in AI, or strategy building in game theory is a difficult one. The usual way to do it is to search and rank possible futures within computational limits. Planning with a generative deep belief net is simpler. All the work is done during perception, and generating a plan takes about the same amount of time no matter how complex the problem. Generating likely sequences

of percepts, or "dreaming" as DBN researchers often call it, is the inverse of perception. All that is required to make a plan is to generate a likely sequence of events using a network that has preferentially remembers sequences of that led to desirable outcomes. This is what I am calling "planning by optimistic dreaming." I will test a planning system that uses a stack of de-noising auto-encoders in this way by allowing it to play a game where there are appropriate risks and rewards and a balance between short and long term solutions is optimal. My current choice is the board game "Puerto Rico" which has themes of sustainability and growth. I plan to encode the rules of Puerto Rico or whatever game I eventually decide to use, into my existing open source networked, turn-based game arbitration software, which I previously used in a Mancala AI competition. I will be building off of the deeplearning.net suite of Python tools to implement the stacked de-noising encoder, and running the system on my personal GPU cluster, if I can get that to work. I will be branching off from the term project I worked on for Will Landecker's advanced machine learning class, so I have some code and results already. I expect the planning task to be computationally simpler than the typical machine perception tasks to which this algorithm is usually applied, so I shouldn't run up against any hard physical limits. I'll be spending about half of my time on writing, and half on coding. In my opinion, publishing freely available, well-documented, running code on the Internet, provides the highest value to the machine learning and systems science communities.

References

[1] Lamport, L., *LaTeX : A Documentation Preparation System User's Guide and Reference Manual*, Addison-Wesley Pub Co., 2nd edition, August 1994.