Learning from Games:
Inductive Bias and Bayesian Inference

by

Yue Gao

A thesis submitted in partial fulfillment

of the requirements of the honorary designation to the degree of

Bachelor of Science in Computer Science

University of Wisconsin-Madison

December 17 2008

Advisor: Michael Coen

Abstract

People can learn complex concepts with remarkably little data. The key factor to this incredible learning ability is the “intelligent” choice of an inductive bias that allows humans to make powerful generalizations. As a consequence, in the machine learning community, one of the hardest and most fundamental problems is how to delineate a learner’s inductive learning process, particularly how can human generalize concepts reasonably from only a few positive examples.

I begin this thesis by considering a simple next number game—given a sequence of numbers, come up with the next number. On this simple task, human can eliminate their possible hypothesis space from infinite many to a single one, which reveals the process of how human generalizing from little data and the cognitive bias for solving math problems. This research topic has been an interest to many people already. Brain and cognitive scientists have tried to design a computer program to mimic people playing this game. However, we wanted to solve the problem with a different approach—a more generative framework, which not only fits the training data, but also be able to predict unseen data. Therefore, in order to make computers play the game the way people do, we will need to share our inductive biases or our knowledge space with computers.

We first assumed people know certain concepts in a priori, e.g., prime numbers and addition. Then we constructed a generative framework that given a sequence of numbers as input, creates possible mathematical functions that generated it. When it comes to the implementation phase, we used a probabilistic version of Prolog called Prism to encode the space of functions as if it were a probabilistic linguistic grammar. Subsequently, we used Bayesian networks in Matlab to learn the probabilities from observing people’s preferences. In this thesis, many classic questions related to learning and induction have been asked: Can a concept be learned from nothing? What is human’s cognitive bias? What are the criteria when choosing among multiple models? Solving the next number game is a great example to tackle those problems. In the future, we would like to explore further applications based on our theory or discover.

Contents

Chapter 1

Introduction

1.1 Let's play a game!

Many of us have played a simple game when we were young or at least heard about it. You are given an ordered sequence of numbers and asked to come up with the next coming number. Now let's have some fun and try it! Here is a sequence {1, 2, 3} and what is the next number? Correct, 4. What about sequence {2, 3, 5, 7}? They are all prime numbers and 11 is the next number. What about {101, 1001, 10001}. 100001 is the next number. I am sure you have observed the pattern of the sequence. Patterning finding is something humans are very good at. Maybe, you think solving the next number is a piece of cake. Well, not all sequences are this intuitive, some of them might be quite challenging. Here is another one {9, 10, 12, 14}. Does it look familiar? Not to me at my first glance. The sequence is generated by prime number plus 7 so the next number is 19.

The next number game is probability the simplest game with very intuitive rules. However, the logic behind it is not so trivial at all. For every given sequence, there are infinite possible answers for the next coming number, yet we, humans usually pick a unique one. How did we shrink our solution from infinite possible answers to a single one? Why some of the sequences or number patterns are easier to recognize while others are profound to think of? Do people do math differently, for example, people at different mental development stage or with different education background? Is there a model or framework that can computationally capture the way people play the next number game?

1.2 A Brief Motivation and Goal

Tom Mitchell, the department chair of Carnegie Mellon University has predicted the future of machine learning during an interview. Coincidently, his view also explains the purpose of our project. Mitchell concluded that in the past ten years, machine learning is a rather narrow field because it mainly focused on how to get computer programs to learn some classification from the inputs and outputs. However, when thinking about learning in people, we are not thinking about some processes that run some pre-collected data for a few time, we think of a learning process that involves data collection and analysis for years and decades, which is a cumulative process. Therefore, in the future, Mitchell concluded that the machine learning field will take on more problems that relate to human like learning problems, which is a key essential aspect that we have considered in our project, which makes it unique.

In human cognition, learning is one of the core capacities of human. From a computational point of view, human concept learning is remarkable for the fact that very successful generalization are often produced after experiencing with only a small number of positive examples of a concept [1,2] . Therefore, we want to concentrate on the generalization process that human performs. In addition, we hope to propose a framework for understanding how humans play the next number game and to get computers play it the way people do. With this goal in mind, we developed a theory that tries to explain how the learners generalize the formulation of a sequence of numbers with only a few data examples and still making rational inductive inferences that integrate prior knowledge about plausible hypotheses.

1.3 Roadmap

This section provides a rod map to the rest of the thesis.

Chapter 2 sets the stage for the rest of this thesis by introducing and explaining the basic idea of inductive learning, inductive bias and hypothesis elimination. In addition, we will introduce the Bayesian Occam Razor and provide some background knowledge for probabilistic grammar in the linguistic world.

Chapter 3 demonstrates our approach which creates a framework for the next number game. We will compare previous attempts and existing structures for solving this game. Then, we will emphasis on explanation of our approach, which includes Probabilistic Context Free Grammar, Stochastic logic programming and Miller's magic number and etc.

Chapter 4 is the implementation phase which builds on the framework delivered in Chapter 3. We will explain the reason for using the software Programming in Statistical Modeling (PRISM), how it is being used, and also how we incorporated the Stochastic Logic Programming and Viterbi Algorithm into our implementation. Then we will present the running result and their parse tree will be presented. In addition, preliminary human survey results will be compared with computer’s prediction.

Chapter 5 builds upon the previous two chapters. This chapter focuses on how to train our framework such that the computer’s mind will “think” more similar as human’s. We will be tuning our priors and adjust the system by applying Bayesian network.

Chapter 6 contains a brief summary, the contributions of this thesis and outlines of future possible applications. I would also like to share some of my personal experience and lesions that learned from the process of working on the projects.

Chapter 2

Setting the Background Stage

When we were infants, our brains are like sponges. We babble, within a few years, we can talk in our native language [2,3] At the same time, we learn concepts of language, either abstractly or realistically, we learn “mom”, :dad”, :”cat”, “apple”, “no”…… If we consider every concept as a set of properties, then surprisingly, we never has to see all the properties in a given concept before we learn it [3]. Just like a kid does not have to see all the horses in the world to learn the notion of a “horse”. Sometimes we take it for granted that human somehow owns that powerful and magical ability of generalization of knowledge. Therefore, in this chapter, we will first focus on the background information related to inductive learning and inductive bias. In the second half of the chapter, we will introduce linguist probabilistic grammar which influenced our model selection.

2.1 Inductive Bias and Hypothesis Elimination

In the machine learning community, often, we are interested in concept learning, which can be formulated as a searching problem through a predefined space of potential hypotheses, which fit the training examples the best [4]. However, if no information is given ahead of time, searching through infinites spaces would be an impossible task. Therefore, inductive bias or some predefined knowledge or rules could be the key to make the search faster and more efficient. What exactly is inductive bias? Inductive bias is defined as any basis for choosing one generalization over another, other than strict consistency with the observed training instances [4]. Before we get to the details about inductive bias, let’s first demonstrate how the next number game relates to the above issue. Here is a given sequence {1,2,4}. There are many possible answers for the next coming number, such as:

6, n=primeNimber -1, Sequence 1, 2, 4, 6, 10, 12, 16,……..

7, n= $i(i+1)/2+1$, Sequence 1, 2, 4, 7, 11, 16, 22, 29, …

8, $n = 2^i$ Sequence 1, 2, 4, 8, 16, 32, 64, …

9, *n* = partial sums of Catalan numbers Sequence: 1, 2, 4, 9, 23, 65, 197, …

All the above answers are valid and it is rather difficult to distinct which one is a better choice. So how do people make a preference? It is due to our innate preference that helps us in making choice between choices or acting as a decision making criteria. Let's look at another issue. Here is a data set with *x* varies from 0 to π/2 in increments of 0.1. In machine learning, fitting the known data to certain model is a common task, so what functions can be fitted into this data set?

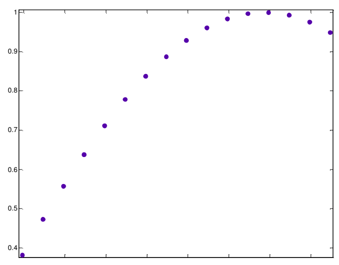


Figure 1 x varies from 0 to $\frac{\pi}{2}$ with increment of 0.1. The dataset is fitted to three different models shown in figure 2,3 and 4.

A dataset is provided displayed in figure 1. The dataset will be fitted to three different models which are shown in figure 2,3,and 4. Figure 2 is the model with function $y=\sin(x)$. Figure 3's function is $y=-0.07x^3-0.2x^2+1.1x-0.03$ and figure 4's function is $y= e^{(1.7-x)^2}+0.3\, e^{2(.8-x)^2}$. These three models explained the data set perfectly well with r^2=1, but which one is the correct model for the dataset?

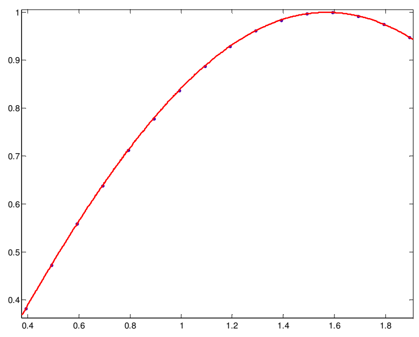


Figure 2 : Function $y=\sin(x)$, the models fits the dataset shown in figure 1 with r^2=1.

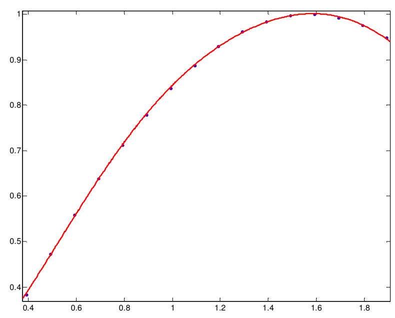


Figure 3 Function: $y=-0.07x^3-0.2x^2+1.1x-0.03$, the models fits the dataset shown in figure 1 with r^2=1.

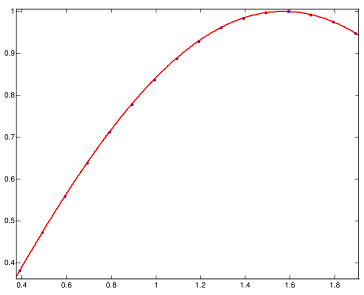


Figure 4 function $y = e^{(1.7-x)^2} + 0.3\, e^{2(.8-x)^2}$ the models fits the dataset shown in figure 1 with r^2=1.

Now let us reveal more of the dataset, *x* varies from 0 to 4π in increments of 0.1 rather than 0 to $\frac{\pi}{2}$. Suppose figure 5 is the real dataset, based on the extension of the data set shown in figure 1. Then the best model that fits would be $y = e^{(1.7-x)^2} + 0.3\, e^{2(.8-x)^2}$. However, what if figure 6 is the real data set? Then function y=sin(x) would be a better model. The above example demonstrate two issues: 1), for any finite data set, there are infinite many possible solutions to it., 2) in order to have a preference to choose a model over others, how to set up and describe the inductive bias computationally is the most difficult part.

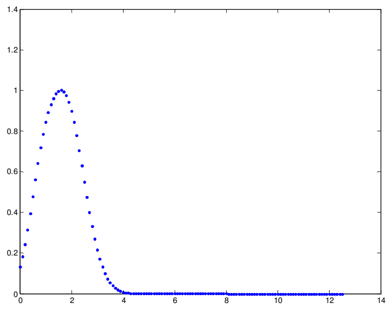


Figure 5, *x* varies from 0 to 4π in increments of 0.1

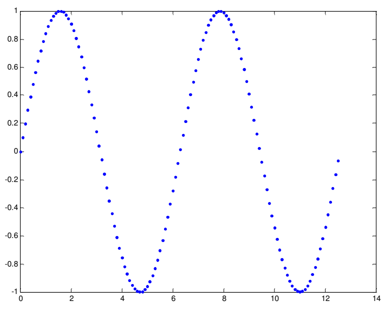


Figure6 *x* varies from 0 to 4π in increments of 0.1

In addition., the above issue is a reflection of the "No free lunch" theorems—learning is theoretically impossible without knowing something in advance about what you are learning [6]. Wolpert and Macredy presented a formal analysis in the "No Free Lunch" theorem which contributes toward such an understanding by addressing questions relating to solving black-box optimization problems and the connection between algorithms and cost functions [6]. Their theory confirmed our previous assumption that in order to make computer systems think in the human mind, it needs to inherit inductive bias from us.

Choosing models for finite data set which is a basic questions of inductive bias. Another problem related to inductive bias is: how to choose a learner's hypothesis space so that it is large enough to contain a solution to the problem being learnt, yet small enough to ensure reliable generalization from reasonably-sized training sets. Unusually, such bias is supplied by hand through the skill and insights of experts[4,7]. Often the hardest problem in any machine learning task is the initial choice of hypothesis [4]. Once a suitable bias has been found, the actual learning task is often straightforward. Many of the existing methods usually require the help of human expert in the domain knowledge, for example in the probabilistic linguistic field, in order to collect data for Bayesian network/s training, professional linguistics are involved to mark the grammar parse for the Wall Street Journal. Even though this method is quite a success so far, it is limited by the cost of it and the reliability. In addition, Jonathan Baxter stated that the best way to bias the learner is to supply it with an containing just a single optimal hypothesis, but finding such a hypothesis is precisely the original learning problem, therefore, he and other researchers have proposed varies frameworks to model human inductive bias learning [4].

As we stated earlier that in the next number game framework, the human inductive bias is crucial for creating a system that mimics the human play. In addition, according to the no free lunch theorem, there is no channel we can accomplish such a task without prior knowledge about the humans. However, how much information is sufficient and what structure to use in order to represent the inductive bias are the questions need to be answered.

2.2 Bayesian Occam Razor and Bayesian factor

In the previous section we concentrated on the intelligent choice of human mind and the importance of inductive bias and its difficulties in modeling it. In this section, we will concentrate on how to formulate inductive bias computationally during the model selection process. Besides the traditional approaches which emphasize on the power of statistical learning, the idea of Occam's Razor —"Entities should not be multiplied beyond necessity"— has been widely adopted as a key principle in evaluating different hypothetical models [10].

MacKay in his thesis first introduced the notion of Occam factor., which explains how Bayes rule provides an automatic "Occam's razor" effect and penalizes unnecessarily complex models [11]. This idea extended the traditional evaluation of a model fitness criterion to evaluation of the parameter space and the way in which the parameters are combined into the model's equation [9]

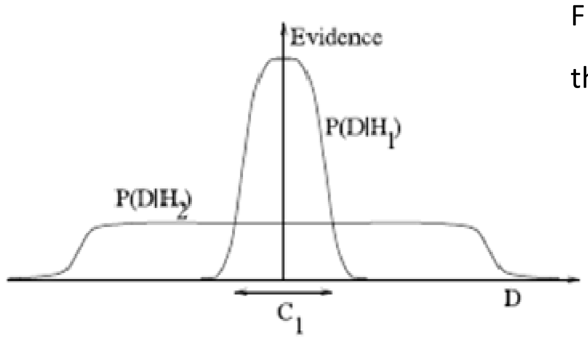


Figure 8 is a graph for demonstrating the Bayesian Occam Razor.

Figure 8: The D-axis indexes all possible data sets (under some idealized ordering). Each curve gives a probability distribution over data sets, so must enclose an area of 1. H_1 is a simple model focusing on data in region C_1. Given data is this region; H_1 has more evidence than a more powerful model H_2, which would be favored given more complex data (outside C_1) [11].

The Model fitting process is to maximize the probability of the data given a particular hypothesis; ie, $\max(P(D \mid H_i)) \approx P(D \mid w_{MP}, H_i) \times P(w_{MP} \mid H_i)\sigma_{w|D}$ where w is the parameter for hypothesis H_i, D is the data and $\sigma_{w|D}$ is the posterior uncertainty [4]. The model assumes the posterior has one strong peak at the most probable parameters w_{MP}, so using Laplace's method, the peak of the integrand multiplied by its width $\sigma_{w|D}$ is an approximation of the $P(D \mid H_i)$ [4]. Mackay represented the Occam factor as $\frac{\sigma_{w|D}}{\sigma_{w|H_i}}$ which represents the "collapsing speed" of H_i.'s hypothesis space when the data arrives or the speed of changing from a general and large hypothesis space such as H_2 to a more specific hypothesis like H_1. This model works remarkably well for systems with a finite number of parameters and creates a complexity that is almost equivalent to the minimal description length (MDL) principle [12]. However, challenges arise when learning happens in a continuous distribution [12].

After demonstrated the theory of Occam factor, here is a demonstration of how this framework might be applied in the next number game or even finite data point interpolation. Even though the number game is a complex combinatory math problem, human plays this game very fast with very simple answers. So we think the number game is a great example to formulate and test the Occam's razor framework with the Bayesian approach. For example we can choose the following hypothesis:

$\mathrm{H}_1: \ a_{n+1} = c_1^{\,n} + c_2$ $\quad c_1$ and c_2 are integers $\in (0,10)$ power

$\mathrm{H}_2: \ a_{n+1} = c_1 a_n + c_2 a_{n-1} + c_3$ $\quad c_1, c_2, c_3$ are integers $\in (0,10)$ sequence addition

$\mathrm{H}_3: \ a_{n+1} = c_1 n^3 + c_2 n^2 + c_3 n + c_4$ $\quad c_1, c_2, c_3, c_4$ are integers $\in (0,10)$ polynomial

If we choose the hypothesis parameters in a finite space, two important questions we need to solve are how to calculate prior probabilities, and how the Occam's razor factor affects the learning model compared to traditional Bayesian learning approach. If our hypothesis is on the continuous domain, then solving the Occam factor would involve a careful mathematical formulation and justification.

Another interesting background is that " Occam Factor" is elicited by the presence of the "Bayes factor." Back in 1961, in Jeffreys's book *Theory of Probability,* he developed a methodology for quantifying the evidence when choosing between two models. The centerpiece was a number, now called the *Bayes factor* [14].

$$\text{Bayes Factor} = \mathrm{B(x)} = \frac{p(D \mid H\ )p(H_1)}{p(D \mid H\ )p(H_2)}$$

Although there has been much discussion of Bayesian hypothesis testing in the context of criticism of P-values, less attention has been given to the Bayes factor as a practical tool of applied statistics. From Jeffreys's Bayesian point of view, the purpose of hypothesis testing is to evaluate the evidence in favor of a scientific theory [14] Bayes factors provide a way of incorporating external information into the evaluation of evidence about a hypothesis. Jeffreys suggested interpreting *B(x)* in half-units on the scale of $\log_{10}$ Pooling two of his categories together for simplification we have:

$\log_{10}(B^{-1})$	*Evidence against H_1*
0 to 0.5	*Not worth more than a bare mention*
0.5 to 1	*Substantial*
1 to 2	*Strong*
>2	*Decisive.*

Table 1: a demonstration of the interpretation of the Bayesian factor.

Because $\sqrt{10} = 3.16$, so the value of B^{-1} less than 3 should be considered no real evidence against the null hypothesis, while "strong" evidence is reserved for values of B^{-1} of at least is 3., as we have already indicated, Bayes factors can equally well provide evidence *in favor of* a null hypothesis. Probability itself provides a meaningful scale made operational through betting and, thus, the labeling of these categories is not a calibration of the Bayes factor but rather a rough descriptive statement about standards of evidence in scientific investigation

When modeling the Occam Razor concept, one basic question we have to considering is that in reality, what are the criteria for decision making. The interesting fact is that people are not necessarily dawn to the simplest solution to problems, sometimes we “multiply complexities.” If this is the case, then there is an inherent discrepancy between Occam’s Razor and the notion of inductive bias. In our framework, when we are talking about mathematical formulations, the mathematical niceties can be interpreted as simplest is still a doubtable question. So neat math might only be pure math, which might reveal very little information about cognitive behavior. In addition, the whether the human can interpret Bayesian factor cognitively as Table 1 is still doubted.

2.3 Probabilistic Linguistic

Before I start to explain our framework in the next chapter, I will introduce probabilistic linguistic, which is our inspiration for the project. Therefore, in this section, background knowledge for linguistic development particular probabilistic linguistic are introduced.

Human language has its significance as we communicate in language, learn in language, think in language and it is only unique to humans. So building models and understand how language is structured and how we can learn language very fast as a kid are all the amazing aspect of it. Back in 1957, B.F. Skinner established a theory that Everything can be learned from observation which is the experimental analysis of behavior in 1957. However, this theory is challenged by Noam Chomsky. In 1959 [15, 16, 17], Chomsky objects skinner's opinion and proposed his universal grammar which gives rise to generative grammar. He made the argument that the human brain contains a limited set of rules for organizing language. As a result, there is an assumption that all language have a common structural known as universal grammar [17]. After Chomsky's has introduced his theories, there are many other theoretical models for fitting linguistic data such as Minimum Description Length [19]. In addition, horning's proof using Bayes' Rule

$p(h \mid d) \propto p(h) p(d \mid h)$, which can be seen as a form of minimum description length. Currently, MDL and other statistical models are in application in linguistic and many related fields.

Chapter 3

A Framework for the Next Number Game

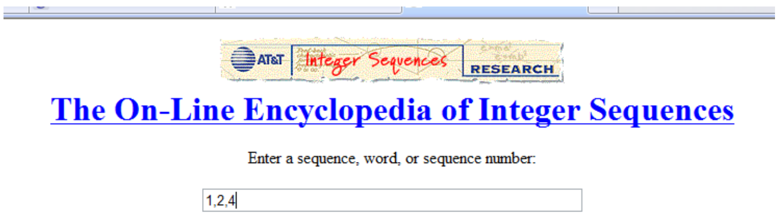
3.1 A Brief Demonstration.

In this chapter, I will mainly concentrate on our framework for solving the next number game. Firstly, I will compare several previous research work that are related to the topic. All these approaches have their uniqueness in their interpretation of the problem. They provided different views to us before we started our own approach. Then I will introduce several theories that explains how human play the next number game. In the end, I will explain how we adopted the theories and created our methods in modeling the next number game.

3.2 Related Research Work

3.21 AT&T lab development of the a sequence database

AT&T has created an on-line encyclopedia of integer sequences. Figure 9 is the home page of their software. This website is man-made database which has been developed by AT&T for several years. The advantage of it is that the wide and variety of range sequences the system covers, particular some very academic and uncommonly used ones are also included. That is a main reason for why it has become a popular tool among mathematicians. However, the system is very easy to be broke. For example, just simply by shifting an existing sequence. For example, if we type 2,3,5,7, the system will tell you it is prime numbers. However, if your input sequence is 5,6,8,10, then the system will be lost and not able to recognize it is prime number+3. Therefore, the system mostly performs as a database look up. It does not aim to mimic the human play.


The On-Line Encyclopedia of Integer Sequences

Enter a sequence, word, or sequence number:

1,2,4

Figure 9 This is the AT&T on-line encyclopedia of integer sequence. It is fun to

play around with the system and they offer some very interesting sequences.

3.22 Josh Tenenbaum: A Bayesian Framework for Concept Learning:

Another attempt to solve the next number game is by Josh Tenenbaum in his PhD thesis [3]. In his work, he created a system that used size principle, Bayesian framework to train and interpret how human play the next game. However, his approach does not involve order. So given a set of numbers, his system could rank a range numbers and output the likelihood of the probability of that number belongs to the set. His approach has four following ingredients.

The first ingredient in his model is in common with many other classical approaches to induction, is to assume a hypothesis space of candidate extensions for the concept to be learned. Without some kind of restriction on the hypotheses we consider, generalization from any finite evidence is impossible. In learning number concepts, the hypotheses included both mathematically special classes [3], such as all powers of two, and sets of numbers with similar magnitudes, such as all numbers between 10 and 20.

The second ingredients in his model of the Bayesian framework is to assign a prior probability to each element of the hypothesis space [3]. The prior embodies our beliefs about which hypotheses are the most likely candidates for new concepts in general, independent of any examples we have seen. In some sense, the hypothesis space itself is an extension of the prior; excluding logically possible hypotheses from our hypothesis space is equivalent to including them but assigning them a prior probability of zero. However, the prior allows us to encode finer degrees of preference. In the number game, our prior assigned higher weight to mathematically special sets of numbers, like square numbers or even numbers, relative to more psychologically generic interval-based hypotheses, like numbers between 12 and 32.

$$p(h) = 1 - \lambda \quad \text{for magnitude hypothesis}$$
$$p(h) = \lambda \quad \text{for mathematical hypothesis}$$
$$\lambda = \frac{\mathit{number} \text{ of mathematical properties}}{\mathit{number} \text{ of magnitude properties}}$$

The third ingredient is a generative model of the examples, which allows us to score hypotheses based on their likelihood of having produced the data that we observed [3]. Throughout, we made the assumption of strong sampling: the examples are a random sample from the concept's true extension. Strong sampling leads to the size principle for scoring hypotheses, which is how we were able to decide between two (or more) possible generalizations, each natural a priori and each consistent with the data. The size principle says that smaller hypotheses are more likely to be the true concept than larger hypotheses, and they become exponentially more likely as the number of consistent examples increases. The intuition is a statistical one: just as we would be much less likely to encounter a tight cluster of points if we were sampling from a large region than if were sampling from a smaller region, so would we be much less likely to encounter 6, 8, 2, 64 sampling from all even numbers as opposed to all powers of two. This idea is his version of the Occam Razor.

$$p(h \mid X) = \left(\frac{1}{size(h)}\right)^n$$

Finally, in Tenennbaum's framework, the actual generalization behavior in the Bayesian framework is determined by the posterior probability and the principle of hypothesis averaging [3]. The posterior probability of each hypothesis is equal to the product of its prior probability and size-based likelihood. This gives the rational degree of belief in each hypothesis as a function of both our prior knowledge about more or less natural candidate extensions and the statistical information carried by the examples. Then, in order to decide the probability that any new object belongs to the concept, we average the predictions of all our hypotheses, weighted by their posterior probabilities.

$$p(y \in C \mid X) = \sum_{h \in H} p(y \in C \mid h)\, p(h \mid X)$$

Josh's approach offers the first attempt in trying to solve the next number in a cognitive aspect. He took the human preference into consideration and developed his theory of representing the problem. Particularly, he applied size principle and Bayes rule in understanding the human mind. The problem with Josh's approach is that the model won't generalize to broader sets. For the next number game, given any finite number set, there are infinite many solutions to it. So in his Bayesian framework, it is impossible to keep track infinite many data sets or infinite many sequences that are infinite long. Therefore it only works on the data sets it has been trained on.

3.23 Solving inductive reasoning problems in Mathematics: Not-so-Trivial

Another piece of research done by Lisa A. Haverty from Carnegie Mellon University has investigated the cognitive processes involved in inductive reasoning [25]. They did experiments on sixteen undergraduates by asking them to solve the quadratic function finding problems. In their research, three fundamental areas of inductive activity were identified: data gathering, pattern finding, and hypothesis generation. These activities are evident in three different strategies that they used to successfully find functions. In all three strategies, pattern finding played a critical role not previously identified in the literature. In the most common strategy, called the pursuit strategy, participants created new quantities from x and y, detected patterns in these quantities, and expressed these patterns in terms of x. These expressions were then built into full hypotheses. The processes involved in this strategy are instantiated in an ACT-based model that simulates both successful and unsuccessful performance. Their protocols and the model suggest that numerical knowledge is essential to the detection of patterns. They also identified the ability to detect and to symbolically describe data patterns as a crucial aspect of inductive reasoning. Their conclusion is that numerical knowledge is a crucial component of inductive reasoning in mathematics.

3.3 Three Possible Theories

As many attempts have been tried to solve the next number game or related math games, the frameworks are based on the following theories about how we treat numbers sequences cognitively and how we store the information. In addition, the following three theories explain how human deals with the next number game from different perspective. Set theory is a mature theory that work in some condition. Similarity theory is fairly new comparing to set theory and it deals with pattern matching. The last theory is our framework for solving the next number game--Context free grammar from an linguist point of view, which explains the next number game computationally quite well.

3.31 Set theory.

Set theory is the branch of the mathematics that studies sets, which are collections of objects. Although any type of objects can be collected into a set theory, in our framework, it is mostly numbers. Set theory is a foundation for the similarity theory which explain how human brain solve the next number game [18]. When given a sequence of numbers, some people believe that matching the sequence of numbers to a known set is cognitively plausible. These sets can be predefined sets such as prime, natural numbers, even numbers, and etc. However, this theory is limited to only a small amount of numbers and most importantly, it did not consider the human mind's computations and how we do math which involves more complex operations.

3.32 Similarity Theory

The capacity to judge one stimulus, object or concept as similar to another is thought to play a pivotal role in many cognitive process, including generalization, recognition, patter matching, categorization and inference, Consequently, modeling subjective similarity judgments in order to discover the underlying structure of stimulus representation in the brain and mind holds a central place in contemporarily cognitive science [18,20,21]. Furthermore, people believe that two objects appear similar when they are likely to have been generated by the same process. That is also another reason why people base their model on this theory.

There are many types of Mathematical models of similarity, which can be roughly divided into two categories. Spatial models in which stimuli correspond to points in a metric space and similarity is treated as a decreasing function of distance and set-theoretic models, in which stimuli are represented as members of salient subject [21]. In the set theoretic models, the subsets presumably corresponding to natural classes or features in the world and similarity is treated as a weighted sum of common and distinctive subsets [18]. Based on this theory, we can try to explain how people play the next number. First assuming people have certain sets in their mind, for example prime number. When given sequence {2,3,5}, we would try to match the sequence with the sets already know and find out how similar they are. If there is a match, then we say we found the set and accordingly, the next number would be 7. If the sequence is generated by the knowledge sets they already know, people can expect a high accuracy of finding the right next number. However, for a given sequence that is not in the knowledge space, matching it to any sets would be impossible and coming up with the next number is highly unlikely. Therefore, this ignored the human cognitive intelligence of the computation aspect, we can do math and create combination of sets, which we have never seen before. In addition, similarity theory and pattern matching's framework, the computation complexity of coming up with the next number is the same for any given sequence which is not the case at all. According to the above discussion, we believe there should be a better theory to support the cognitive process of human playing the next number game.

3.33 Context Free Grammar

In the section, we are proposing a different framework for solving the next number game, which is also a framework we adopted in our implementation. This theory is call Context Free Grammar, a linguist notion. Firstly, we will look at how it is defined. Then we will explain why this theory is better in explaining the next number game.

The context-free grammar formalism developed by Noam Chomsky in the mid-1950s took the manner in which linguistics had described this grammatical structure, and then turned it into rigorous mathematics [15]. A context-free grammar provides a simple and precise mechanism for describing the methods by which phrases in some natural language are built from smaller blocks, capturing the "block structure" of sentences in a natural way [37]. Its simplicity makes the formalism amenable to rigorous mathematical study, but it comes at a price: important features of natural language syntax such as agreement and reference cannot be expressed in a natural way, or not at all. Context-free grammars are simple enough to allow the construction of efficient parsing algorithms which, for a given string, determine whether and how it can be generated from the grammar.

A formal definitions for CFG is the following: A context-free grammar *G* is a 4-Tuples $G = (V, \Sigma, R, S)$ where [37]

V is a finite set of *non-terminal* characters or variables. They represent different types of phrase or clause in the sentence.

1. $\sum$ is a finite set of *terminal*s, disjoint with V, which make up the actual content of the sentence.
2. S is the start variable, used to represent the whole sentence (or program). It must be an element of V.
3. R is a relation from V to $(V \cup \sum)^*$ such that $\exists\ \omega \in (V \cup \sum)^* : (S, \omega) \in R$
4. In addition, R is a finite set. The members of R are called the *rule*s or *production*s of the grammar.

Context Free Grammar provides much flexibility in explaining the next number game due to the way math is structured and math is recursive. Math is structured science with very stricter formula grammars. With specific notion for every operation, there is very few or no ambiguous in it. So treating it as a kind of language is very proper. Therefore, the fact that CFG can represent the structure of math is critical in its representation. In addition. CFG can represent the recursive property of language. When we are writing math formulas, we can use any of the operation symbols multiple times and it is hard to find a more easier framework that can capture this. Based on the advantages that CFG can brought to us, we choose it to represent math formulations in our system.

Furthermore, in our framework, we also add probability to the Context Free Grammar's terminal and nonterminals which leads to Probabilistic Context-Free Grammar. Given a grammar and a sentence, a natural question to ask it whether there is any tree that accounts for the sentence and if there are many trees, which is the right interpretation. For the first question, we need to be able to build a parse tree for a sentence if one exists. To answer the second, we need to have some way of comparing the likelihood of one tree over another. There have been many attempts to try to find inherent preferences based on the structure of trees. For instance, we might say we prefer tree that are smaller over ones that are larger. But none of these approaches has been able to provide a satisfactory account of parse preferences. This formalism is called a probabilistic context-free grammar (PCFG). PCFG provides convenience for mode selection. As we demonstrated in Chapter 2.1 that we any given finite data sets, there are infinite models that fits the data. Therefore, model selection would be critical and PCFG provides a probability value for doing do.

3.4 Our Approach: Probabilistic Context Free Grammar

3.40 Grammar Expansion

According to last section's explanation, in this section, we will demonstrate our Probabilistic Context Free Grammar. Figure 10 is a representation of the grammar for our framework. The super script is the probability assigned to each terminal and nonterminals.

- Expression → PrefixOp (Expression)^{p}
- Expression → Expression InfixOp Expression^{p}
- Expression → Sequence$_{i-1}$ ^{p} | Sequence$_{i-2}$ ^{p} | ^{p}Sequence$_{i-3}$ ^{p}
- Expression → Number^{p}
- Expression → index^{p}
- PrefixOp → exp^{p} | log^{p} | sin^{p} | cos^{p} | tan^{p}
- PrefixOp → floor^{p} | ceiling^{p} | mod^{p} | rem^{p} | prime^{p}
- InfixOp → +^{p} | -^{p} | x^{p} | /^{p} | ^^{p}
- Number → SmallNumber^{p} | SpecialNumber^{p} | LargeNumber^{p}
- SmallNumber → -9^{p} ... 9^{p} (except 0)
- SpecialNumber → -10^{p} | 10^{p} | π^{p} | 0^{p} | 100^{p} | -100^{p} | ½^{p} | ¼^{p} | 1/3^{p}
- LargeNumber → [-50^{p} ...-11^{p}, 11^{p}... 50^{p}]

Figure 10 is a representation of the grammar

When designing the grammar, we took consideration of the index number of the sequence. For generating a math functions, index number is crucial. Also for special sequence such as Fibonacci, in order to know the previous two numbers, index number is a key variable to consider. In addition, for our grammar, we mainly focused on integers and common fractions. Since there are infinite many integers, we limited our system's integer in the domain [-50,50]. The probability assigned to the terminals and nonterminals, each p value varies and their values are determined by later training. For prior probability, we will explain more at section 5.2 where we will particularly discuss about how to come up with priors and how important it is to calculate the accurate one.

3.41 Miller's Magic Number and Threshold

For our probabilistic context Free grammar, we took the magic Miller's number into consideration as well. If the grammar is recursive [36], then the program won't halt which can not be handle by any machines, Same with humans, our brains will not deal with infinite numbers or solutions as well. Back in 1956, Harvard University-based psychologist George A. Miller published a paper in Journal Psychology Review that provide a fascinating insight into the human memory and demonstrated our limitation in memory [36]. Miller found that the short-term memory of different people caries, but found a strong case for being able to measure short-term memory in terms of chunks and he proposed the miller's magic number 7 ± 2. This magic number is an indication of the maximum of terms that one can occur and remember instantly in a given context.

By borrowing the same idea, in the next number game, we assume that people has certain limit in the complexity of their operations. So for computers, this constrain is represented as threshold. However, the key question is how large or small should the threshold be that can best represent the limit in our brain? Individual difference can not be ignored. So the exact threshold should be determine by the training result. For now, we are assuming the threshold is 0.000001 in terms of probability.

Chapter 4

Implementation of the Framework

4.1 Choose A Language: prolog

When it comes to implementation, I am very surprised by how time consuming it is to select the proper tool. Even though we are very clear about the goal we want to achieve, without trying different implementation programs , it is hard to predict which one is better and more efficient. At first I wanted to stick with the programming language I know better, such as Java, Matlab; however, none of them can implement grammar rules or it would be too time consuming. Then we tried Jess, which is more friendlier than the previous languages when coding rules and grammar, but it is still very redundant. Afterward we decided to use prolog. Prolog has many versions provided by different software developers and the traditional prolog is very “dumb” when solving math problems because there are only limited operations. With some help from other faculties, we made our decision and we chose Prism--programming in statistical modeling. That is the software we finally used for implementation and it is amazing how good the tools is for fitting into our framework.

4.11 JESS

JESS is a rule engine and scripting environment written entirely in Sun's Java language by Ernest Friedman-Hill at Sandia National Laboratories in Livermore, CA. Using JESS, one can build Java software that has the capacity to "reason" using knowledge you supply in the form of declarative rules. The most obvious advantage for JESS is that it can access to all of Java's APIs which can be used in eclipse platform.

Therefore, JESS can provide different data structures which is more convenient than Prolog. Mainly, Jess can be used in two overlapping ways. First, it can be a rule engine. a special kind of program that very efficiently applies rules to data, particularly useful for building a rule-based program with hundreds or even thousands of rules. Jess will continually apply them to data in the form of a knowledge base. But the Jess language is also a general-purpose programming language, and furthermore, it can directly access all Java classes and libraries. For this reason, Jess is also frequently used as a dynamic scripting or rapid application development environment. When we actually start using JESS for rule based engine. Its efficiency is still not that obvious comparing to Prolog, there are still redundancies.

4.12 Prolog

By comparing JESS and Prolog, what Prolog offers is a more efficient tool that enables the user to put much attention on the logic aspect rather than programming syntax. This is mainly due to the fact that Prolog is a declarative language. There are many ways of organizing computations. Perhaps the most familiar paradigm is *procedural language*: the program specifies a computation by saying how it is to be performed. JESS, C, and even object-oriented languages fall under this general approach. Another paradigm is *declarative*: the program specifies a computation by giving the properties of a correct answer. Prolog, as an examples of declarative languages, emphasizes the logical properties of a computation, which are often called as logic programming languages. However, the declarative or procedural distinction is not rigid: Prolog of necessity incorporates some procedural features. As a conclusion, the nonprocedural programming is saying “what” instead of “How.”

One might wonder what are the merits of declarative language and why Java, Jess and other procedural language can not accomplish the task equality good. Declarative Languages offers three important advantages. Logic programming languages are inherently ``high-level'' because they focus on the computation's logic and not on its mechanics which makes prolog much easier to express CFG than JESS. The result is that they are well-suited to expressing complex ideas because the drudgery of memory management, stack pointers, etc., is left to the computational engine. In addition to representing many facts compactly, intentional representations lend themselves naturally to representations of mechanisms, or to ways of generating related representations. A consequence of the proceeding is that logic programming languages are particularly suited for rapidly prototyping data structures and code to express complex ideas. Reduced drudgery and compact expression means the developer --- and even the scientist! --- can concentrate on what should be represented and how. The use of the built-in inference engine permits rapid and straightforward evaluation of the code. We find repeatedly that difficulties in coding are due to vagueness and illogic in our thinking. Prolog is a declarative language. By stating the facts and rules which relate objects in the problem domain to each other, you construct your Prolog program. Its meaning is the set of logical consequences of these program statements, and this is computed by the inference engine at run-time. You do not have to be concerned with telling the machine how to solve the problem, nor where to put data in memory. This allows you to concentrate on the problem at hand rather than on software concerns. Scoping rules are simple and uniform in Prolog, and declaration of variable names is not required. This reduces code size and opportunities for error. Prolog programs tend to be from five to ten times smaller than the equivalent procedural programs. This reduces the opportunity for human error and reduces maintenance cost. For all the above reasons, we choose prolog over JESS and other procedural programming languages.

However, Prolog is still not the perfect language we are looking for. Currently, there are many different types of prolog versions such as yap and swi-prolog. Most of those prolog engines are not good at math. They can only solve some simple math calculation and does not support complex algebra. In order to solve the next number game, we need a programming language that can encode the grammar as well as good at math. Luckily, we found Prism which can satisfy both requirement.

4.13 Prism

PRISM is acronym of Programming in Statistical Modeling, which is a programming language for symbolic statistical modeling. The program's modeling is to target complex phenomena governed by rules and probabilities and gene-inheritance, stochastic NLP, consumer behavior and etc [29,30]. PRISM is based on b-prolog which is a version of prolog but with more flexible mathematical function. As a conclusion, PRISM=Prolog + probability + parameter learning [29].

PRISM is a general programming language intended for symbolic-statistical modeling. It is a new and unprecedented programming language with learning ability for statistical parameters embedded in programs. Its programming system is a powerful tool for building complex statistical models. The theoretical background of PRISM system is *distribution semantics* for *parameterized logic programs* [25]. The PRISM system is comprised of two subsystems, one for learning and the other for execution [35], just like a human being has an artery and a vein. The execution subsystem supports various probabilistic built-in predicates conforming to the distribution semantics and makes it possible to use a program as a random sampler. The learning subsystem on the other hand supports abductive reasoning [35]. PRISM is very convenient when searching for logical explanations of observations through a program and learning parameters associated with them for the adaptive behavior of the program. When I first learned PRISM, I find it not too difficult to catch up. The major convenience that PRISM provides is its ability to incorporate with probability with our framework. In addition, since it is build on b-prolog, its contains most commonly used mathematic functions.

4.3 Stochastic Logic Programming

The integration of logic and probability theory has been the subject of many studies. It has also been investigated within logic programming. Recently, the motivation for many such studies has been the development of formalisms for rule-based expert systems which employ uncertain reasoning. The motivation for Stephen Muggleton creating Stochastic Logic programming comes from machine learning, particularly for representing machine learning algorithm's bias over the hypothesis and instance space with probability distribution[35]. The previous approach has been taken in various ways within the frameworks of PAC-Learning, Bayesian learning and etc.

4.4 Viterbi Algorithm

In our framework, with the linguistic grammar expansion, when given a sequence of numbers, the search time is quite long. So in order to speed the process, we applied Viterbi algorithm to look for the most optimal path. This algorithm has been implemented in PRISM as a function call which is very convenient. The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states, which called the Viterbi path, which contains the results in a sequence of observed events. The algorithm is most commonly used in the context of Markov Models.

The Viterbi algorithm was conceived by Andrew Viterbi in 1967 as an error-correction scheme for noisy digital communication links. It is now also commonly used in speech recognition, computational linguistics and bioinformatics. A very intuitive way of understanding Viterbi algorithm is by the following story. Imagine you're a detective trying to determine the point of origin for a suspect arriving at the Seattle airport. He is from abroad and has been there for no more than an hour. You know during this period that four domestic flights arrived from four cities served by flights from 30 others. One way to determine the suspect's origin would be to go back to all 30 and make inquiries. The alternative: identify which of the four closest cities he arrived from, then investigate only places with connections to that one. (USC official website) Andrew J. Viterbi algorithm is not just about solving the puzzle in the story. The algorithm takes the result and backtracks, throwing away all the branches that, by the rules, couldn't have lead to the observed result. versions. Instead of analyzing all possible messages in this stack, the algorithm swiftly finds the most probable one, throwing away more and more possibilities with each layer.

In Prism, the build in Viterbi algorithm not only provide the most possible path as the solution and its corresponding probabilities. It can also return back several top most likely paths. The number of selections is specified by the user. In Prism, the function call for Viterbi is viterbif(G,P,Expl), where G is the information passed in, P is the probability for the most likely path and Expl is the explanation graph for the generating function. If the user wants the top N most likely answers, n_viterbi(N,G,Ps) is the build in function call.

4.5 Implementation Algorithm

In this section, I will explain the input and output of our program and also the algorithm that we applied in the implementation. The program takes in a sequence of numbers as input. It will first try to generate all the possible generating functions. In our framework, the number of generating functions is influenced by the complexity of the grammar as well as the threshold value (Chapter 3). Then we will evaluate the given sequence on every generation function. The generating function with the most likely result will be returned if with viterbif function call. The probability that guides the search is the priors assigned on the terminal and nonterminals. The priors are predetermined based on the United States mathematics education system from Elementary school to College. The details of this information will be explained l in chapter 5.

4.6 Running Result and Interpretation.

In this section, we will demonstrate the running result of our program. For any finite number of training set, there are infinite number of generating functions. However, since we provided with a threshold of the probability, the more data points one provide, the less generating functions there will be. For example, if the input is sequence {1,2}. There about 50 functions that fit the sequence, but with sequence {1,2,3}, there are only about 30 functions that fit the sequence.

The following example is parse tree for sequence {1,2,4,6}. The most likely generating function is primNumber -1 with confidence probability as 90%. The second more likely candidate is with 1%, the generating function is floor(index+previousNumber)/2. Even though the more likely generating function has a much simple parse tree expansion, the “simplest” parse tree does not necessarily the most likely answer. The actual probability is the real criteria for determine the most likely candidate.

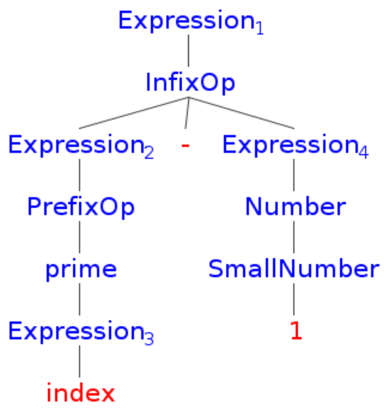


Figure 12 Most likely candidate: 90%: Prime numbers – 1

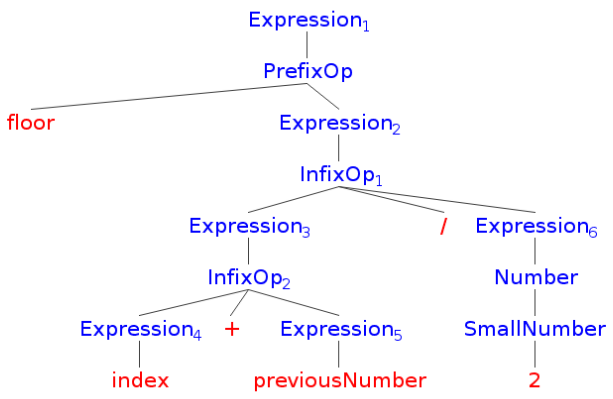


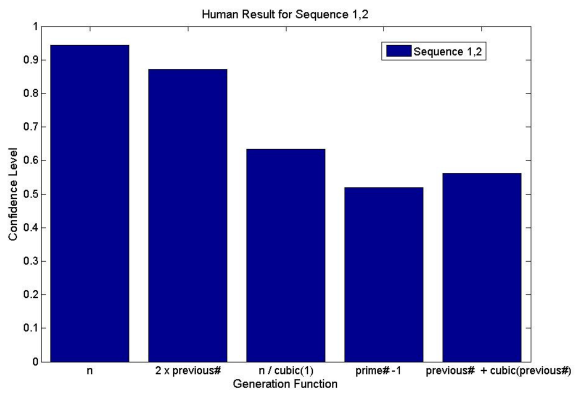
Figure 13 Second most likely candidate: 1% : floor((index+previousNumber) / 2)

4.7 Preliminary Human survey testing and result

Once we had the program set up, we need to collect human data to train and also compare with our system. However, human experiments are not as easy as it sounds. The design of the experiment is very crucial for drawing any conclusion. We had designed two preliminary human surveys.

The first survey is focusing on the “guess the next number game.” We presented a short sequence of numbers to the players and asked them to provide the next one and explain why they view it as the “correct” answer or ask them to write out their generation function. We provided two examples for demonstration purpose and did not impose time limit. Even though this is only a testing survey with five people, the result turned out to be pretty interesting. For example, for sequence {1,2,3}, our system with Viterbi function call said that the next number is 4 with generation function indexNumber.. 80% of the people will say the next number is 4 with generating function as previousNumber + 1, with 4 as the next number, while 20% of the people say it is prime number so the next number should be 5. Of course, the percentage interpretation is definitely not reliable because the sample size is too small, but just this simple example shows that human is not using minimum description length as their decision making criteria. The problem with this type of survey questions is that people can write anything they want, too much freedom will make the later training impossible. So instead, we decided to try another survey type.

The second survey is done on line. We give the human a sequence of numbers and provided them with several generation functions. Ask them to rank the generation function for how likely they will come up with that function. In this survey, we have more control of the human response. For example, given sequence {1,2} we asked seven people to rank the generating functions. Figure 14 is the human response with indexNumber as the most likely candidate. Figure 15 is the computer result. By comparing the computer result versus the human result, we can say that the first likely candidate is a math, but varies for the other three functions. This sequence's preference ranking also indicates that people does not prefer functions involve much computation. For sequence such as natural number, it might just a similarity look up in our minds which is O(1).



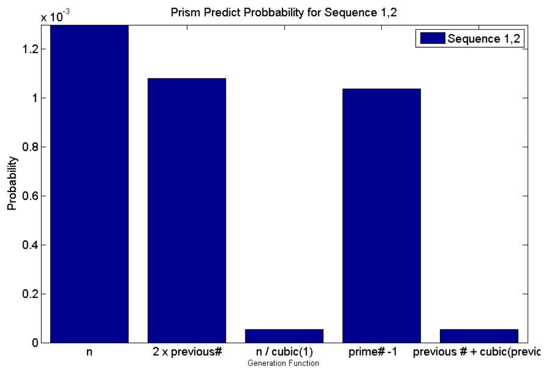
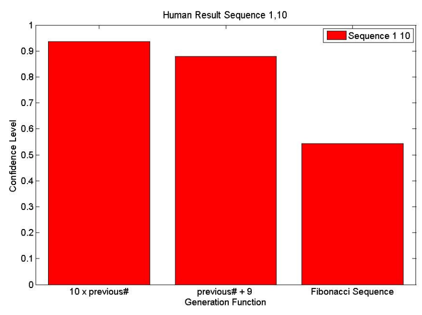


Figure 14 Human responses Figure 15 computer response

Figure 16 and Figure 17 are response for sequence {1,10}. Since human computation are in base 10,, we tend to think math involve with 10 is easier. That is why most people would choose multiply 10 rather than add 9.



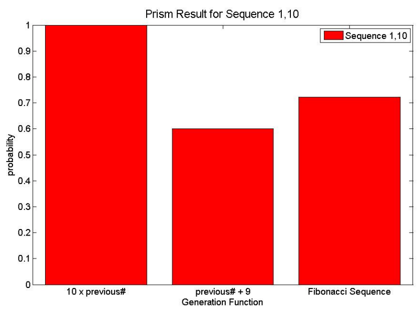


Figure 16 Human result

Figure 17 computer result

From the second type of survey , we can say that people prefer to do addition and multiplication over subtraction and division. We tend to related sequence to existing, well known ones. For example, by given a sequence such as even number, we have a strong bias towards it since no computation is need. A major difference between our computer system and human is that computers follows the with strongest probability of generating the function, which is no the case for human. Human somehow did not choose the simplest answer and maybe even do not have an exact concept for probabilities either.

Chapter 5

Bayesian Network Approach for Training

After we have built our model for the next number game, the next stage is to train the parameters based on human data. So far, our system's performance is based on the prior probability assigned to it. In the next stage, in order to mimic the human playing the game, we choose to use Bayesian Network to train the probability for the non-terminals. Therefore, in this chapter, we will focus on why we choose Bayesian Network, how important the prior is and what pervious work has been done. For training the network, we have not finished the process yet. This chapter's theory is a preparation for the future work.

4.1 Why choose Bayesian? Are we a Bayesian believer?

The interaction between prior beliefs (about the extensions of possible concepts) and a preference to avoid unexplained coincidences (in the relation between concepts and their examples) lies at the heart of the human. Applying Bayesian framework for concept learning is a common tool. One question we need to ask is whether Bayesian can truly represent the way to how people observe, learn and change according to new information from outside of the world. (josh thesis) The Bayesian framework addresses the following three crucial problems of knowledge in concept learning: Content: what constitutes the learner's (uncertain) knowledge about which entities a new concept refers to? Acquisition: how can the learner acquire that (uncertain) knowledge from the evidence provided { one or more positive examples of the concept, and possibly (but not necessarily) negative examples? Generalization: how does the learner use that (uncertain) knowledge to generalize the concept, that is, to decide whether a particular new entity falls under the concept?.

A Bayesian approach to probabilistic modeling is one that takes into consideration not only the probability that is assigned to the data by a model (or as we linguists say, by a grammar), but also the probability of the model. And this latter notion is one that takes us right into the heart of classical generative grammar, to the notion of an evaluation metric. But first we will look at the mathematical side of Bayes's rule. Bayes's rule involves inverting conditional probabilities, although from a mathematical point of view it is a very simple algebraic manipulation. We need first to state what it means to speak of the probability of an event X, given another event Y , written pr(X|Y). This means that we consider only those possible situations in which Y is true, and within that set of situations, we calculate what X's probability is. If we select an integer number at random from all the numbers, the probability will be about very small because there are infinite many numbers, but if we look a sequence of numbers and ask if a particular number is in the set or not, then the probability will be much higher. To calculate such probabilities, when we already have in hand a system which assigns probability mass to all possible representations, we do the following. To determine the probability of X, given Y , we ask: how much probability mass altogether is assigned to all of the events in which both X and Y are true? And we divide this quantity by the probability mass that is assigned to all of the events in which Y is true.

There are many variation of Bayesian related model and all of them involve Bayes rule, which is stated as $P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}.$. P(A) is the prior probability of A. It is "prior" in the sense that it does not take into account any information about event B. P(A|B) is the posterior probability because it is derived from or depends upon the specified value of B. P(B|A) is the conditional probability of B given A and P(B) is the prior of B which acts as a normalizing constant.

5.2 Where do Priors come from?

First consider the term p(h). Now one might ask, how could we have any idea about the probability that some hypothesis h is the true extension of the concept before we have seen any examples? Not only are priors empirically real, they are in principle necessary for any kind of inductive inference to succeed (Goodman, 1955; Watanabe, 1985; Mitchell, 1980). A prior probability is a marginal probability, interpreted as a description of what is known about a variable in the absence of some evidence (citation wiki).

Rational models have gone far by assuming that learning, reasoning, perception, decision-making and other cognitive processes can be understood as approximations to optimal statistical inference -- often some form of Bayesian inference (JOSH) . With appropriately chosen prior probabilities, Bayesian models can explain many aspects of behavior that have previously resisted a principled unifying account. But these successes also raise a challenge, which has been rightly voiced by critics of the approach. Where do the priors come from? Are they merely a choice of the modeler or assumed to be built-in constraints -- in which case they seem like a potentially endless source of flexibility? Or can we give some principled account of how human learners acquire appropriate priors from their experience in the world? If the latter, how do we avoid an infinite regress: where do the priors for the priors come from....?

In our framework, we used the students' math education procedure as the source which provides our priors. We believe that the operation functions which learned in grade one should be more familiar to us than the function that was learned in Calculus in general . and most of the priors among numbers are equal likely. So there is no preference imposed on it. Priors should be a base case in the Bayesian network training and should not be perfectly accurate or even impossible to be perfect. However, with certain evidence backup, as long as the prior can provide the overall trend, It is acceptable within or framework.

Addition: ages 5-7; more digits ages 8-9
Subtraction: ages 5-7; more digits ages 8-9
Multiplication: ages 7-8; more digits ages 9-10
Division: age 8; more digits ages 9-10
Pre-algebra: ages 11-13
Algebra I: ages 13+
Geometry: ages 14+
Algebra II: ages 15+
Trigonometry (or Statistics or Algebra 3): ages 16+
Calculus (or Pre-Calculus): ages 17+

Figure 11 is a table with current US student math education background. This is the source we based our initial prior for our framework. With the longer studying time for certain math operations, we assume people are more familiar with addition than calculus integral. Furthermore, we considered natural numbers with different probabilities as well. For certain numbers, as we called them small numbers, they are easier to interpret and the actual computation related to those numbers are faster. In addition, since our computation is based 10, then 10 is a number usually with a easier computation effort then for example 19. so 10 will initially assigned with a higher probability. As a conclusion, for infix and prefix operations, the probability varies depending on how familiar we are with them. With terminals containing numbers, other than some special numbers, most of them are unified probability. The initial set up for prior should not affect the final training result. It should also be noted that there is no clear line between the choice of a hypothesis space and the choice of a prior. Omitting a particular hypothesis from the hypothesis space is equivalent to including it but assigning it a prior probability of zero. Hence we can think of the choice of hypothesis space as an initial, qualitative prior which is then subject to further refinement in the assignment of p(h).

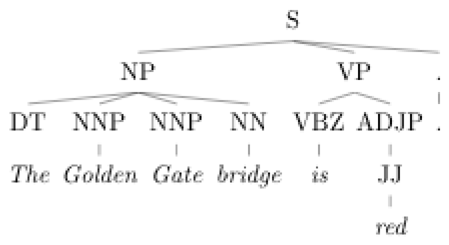
5.3 Bayesian approaches and its implementation

5.31 Josh's Theory-based Bayesian Models.

Josh argued that Bayesian modelers can give compelling answers to these questions, but only if they embrace an idea that has typically been cast in opposition to probabilistic approaches to learning and inference. The key is to understand how rational statistical mechanisms of inference interact with structured symbolic representations of abstract knowledge. I will present a hierarchical Bayesian framework for inductive learning, in which Bayesian inference operates over structured representations of knowledge at multiple levels of abstraction, with each level defining priors for the level below. As representations become more abstract, they become simpler and more general -- and more plausible candidates for built-in cognitive architecture. In many cases, a higher-level knowledge representation may be thought of as a kind of "intuitive theory" for a domain of entities, properties, and relations. The hierarchical Bayesian analysis shows how abstract knowledge of domain structure can generate strong priors for guiding generalization at lower levels, and how that abstract knowledge may itself be learned through rational statistical means. I will discuss applications of the framework to learning and reasoning in several domains, such as natural kind categories and their properties, social relations and causal relations. In the evidential evaluation of scientific theories, prior probabilities often represent assessments by agents of non-evidential, conceptually motivated *plausibility weightings* among hypotheses. However, because such plausibility assessments tend to vary among agents, critics often brand them as *merely subjective*, and take their role in probabilistic induction to be highly problematic. Bayesian inductivists counter that such assessments often play an important role in the sciences, especially when there is insufficient evidence to distinguish among some of the alternative hypotheses. And, they argue, the epithet *merely subjective* is unwarranted. Such plausibility assessments are often backed by extensive arguments that may draw on forceful conceptual considerations.

5.32 Bayesian Network in linguistic and Berkeley Parser.

Bayesian Network is used in linguistic for training their model. Parsing is the task of analyzing the grammatical structure of natural language. Given a sequence of words, a parser forms units like subject, verb, object and determines the relations between these units according to some grammar formalism. Many of the linguistics have focused on learning probabilistic context-free grammars (PCFGs) which assign a sequence of words the most likely parse tree. For example:



The Golden Gate bridge is red.

Currently, one of the most famous parser is the Berkeley Parser, which is the most accurate and one of the fastest parsers for a variety of languages. Parsers are typically trained on a standardized collection of hand parsed sentences. While a variety of techniques have been developed to learn accurate grammars from those examples, most of them require additional, linguistically motivated annotations. In contrast, the Berkeley parser contains data-driven approach, which does not require any additional human input. Beginning with the barest possible initial structure, their system can automatically refine the grammar, introducing more complexity only where needed. Even though there is no human input, the way the grammar structure is still induced. Once learned the grammar for a language, the next stage is to select a parser for a given sentence. The Berkeley parser implements a hierarchical coarse-to-fine scheme which allows the user to find the best parse tree very efficiently. It is constructed by a sequence of increasingly refined grammars by reversing the splits which were induced during training. Then it can parse with each refinement and use its predictions to limit the space of candidate parses, which will be considered by the finer grammars. The advantage of Berkeley Parser is it is particular designed for language. Many languages have been trained on it. However, it is very difficult to be applied in our framework and the amount of training data is huge and are hard to obtain that data from human.

5.3 Bayesian Net and Our approach

The Bayesian Net can reduce the complex relationships between different variables in the parse tree by explicitly specifying the direct influence of variables to other variables in the form of a parent-child connections(citation a Bayesian net for training). Any observations on variables goes into the model as evidence, and the probability distributions of the graph are updated accordingly. One drawback to the Bayesian Net is that a large training database is needed to automatically learn the structure or parameters for a Bayesian Net. For a particular problem domain such as the next number game, this approach may not be possible. Therefore, either of these steps can be substituted by human design by experts. The benefit of using Bayesian Nets for inference is that very limited information is needed to perform inference. Also, since this is a acyclic graph representation, it is easy to represent the relationships between various quantities as connections in this graph. Another area of difficulty for Bayesian Nets lies in active learning of the structure and parameters of a Bayesian Nets containing large numbers of continuous nodes. However, since we have our grammar preprogrammed, this will not be a problem for our approach.

As One of the crucial elements of developing the inference tool is to be able to provide the Bayesian Net with training data so that it could learn the probability associated with parameters. Without this data, the Bayesian Net would have no way of knowing how to act and the project would simply not be possible. More research is needed in order to determine how much data do our system actually need to train the data.

As we have explained in Chapter 4, so far, all the survey we have run are trials and have not received the IRB approval yet. Collecting human data is far more complex than it appears. So far, we decided to use the second approach with multiple choice or ranking type questions such that we could interpret the results. Before we start the actual collection, we have to determine how many people and how large our data set should be. Usually in linguistic Bayesian Network training, their training data is huge. However, their training data is not directly collected form human, instead, it is from Journals. For example, the famous Penn Treebank Project which annotates wall street journal's text. They examine each sentence for its linguistic structure. Their annotations of the text are includes part of speech tag and several other tags which involves much work. In our framework, we do not have a complex grammar tree expansion as the linguists face. Therefore, less data is required. However, the actual question about how many training data is sufficient is still a question to be solved.

Chapter 6

Conclusion

This is the last chapter of my thesis. I would like to talk more about future projects and applications for our research. Since this is my first experience of conducting a real research project, I will also share some personal experience,. It is not only the knowledge I learned from conducting this research, but also the experience of doing research will bring more confidence in my future study. However, it feels relieved to finish the chapter.

6.1 Future work: RNA construction, protein folding

So far, we focused on the next number game, but we would also like to expand this to a broader application. Here I will briefly introduce two possible applications. How could protein folding relate to the next number game?

A central goal of molecular biology is to understand the regulation of protein synthesis and its reactions to external and internal signals(citation). All the cells in an organism carry the same genomic data, yet their protein makeup can be drastically different both temporally and spatially, due to regulation. Protein synthesis is regulated by many mechanisms at its different stages(citation). These include mechanisms for controlling transcription initiation, RNA splicing, mRNA transport, translation initiation, post-translational modifications, and degradation of mRNA/protein. The protein folding problem is to predict the compact three dimensional structure. Even though there are finite number of amino acids, constructing protein has infinite many combinations. The interesting aspect is that our body always choose certain protein folding structure over others for specific functionality. Therefore, protein folding problem is sometimes interpreted optimization problem. That is why protein folding and the next number game are two quite similar problems. We are hoping with our approach, maybe we can solve other problems similarly. (GIT thesis)

In order to solve the problem, a common approach is Bayesian networks, which are a promising tool for analyzing gene expression patterns. As we stated in the last chapter, Bayesian Nets are particularly useful for describing processes composed of interacting components—the value of each component directly depends on the values of a relatively small number of components. Furthermore, Bayesian networks provide models of causal influence. (citation)

Another application is to discover more about human on the cognitive aspect and we want to know more about how people do math. One of the future plans for this work include gathering data from different age groups (e.g., third graders, high school students, math majors) to reveal their inductive biases unique to each population. Also we would like to conduct experiments on how aging reflect short term memory tries to figure our whether the threshold value for human mind will be different. We also need to determine the right threshold value for different age group or for different education background.

6.2 Outside of the Research

In 2008, I have spent a lot of time in this project. Of course, without my advisor professor Coen's support and help, I would not be able to finish it. In addition, as an undergraduate student, I really appreciate this experience. While completing the research project, I have learned an enormous amount about how to carry out a fairly complex research project. Most importantly, I have learned that it is okay to get stuck and ask for help. Professor Coen helped me appreciate the difference between working on a problem set and research. I also overcome the embarrassment to ask questions and express one's own thoughts. This research is a great practice for my future study.

6.3 summary of the project

At the beginning of the thesis, we introduced the next number game—given a sequence of numbers, come up with the next number. On this simple task, it is amazing that human can generalize with little given data. With the motivation that we want to capture the human inductive bias computationally, we investigated previous research on this topic. By learning from other approaches, we decided to come out a more generative framework. At first, we assumed people has certain knowledge about the concepts already according to the "No free Lunch Theorem", then we build our framework with Stochastic Context Free Grammar and Viterbi algorithm. In addition, we need to train our system with Bayesian networks in Matlab to learn the probabilities from observing people's preferences. There are still many possible topics to continue the study of this project.

7 References:

[1] J. B. Tenenbaum. Kearns, M., Solla, S., and Cohn, D. Bayesian Modeling of human concept learning.. *Advances in Neural Information Processing Systems 11.* (eds). Cambridge, MIT Press, 1999, 59-65

[2] Baum, E, What is thought? *The MIT Press*. 2004.

[3] J. B. Tenenbaum, A Bayesian Framework for concept learning. Ph.D. *Thesis, MIT*, 1999

[4] Baxter, J, A model of Inductive Bias Learning. *Journal of Artificial Intelligence Research*. 2000. 149(12).

[5] J. B. Tenenbaum, T. Griffiths, C. Kemp, Theory-based Bayesian Models of Inductive Learning and reasoning. *Trends in Cognitive Science.* 2006, 10(7), 309-318

[6] Wolpert, D.H. Macready, W.G. No free lunch theorems for optimization。 *Evolutionary Computation. IEEE Transactions on.* 1997 1(1)

[7] T. Mitchell. The need for biases in learning generalizations (1980)

[8] MacKay, J, Information Theory, Inference and learning algorithms. *Cambridge University Press*. 2003. Chapter 8

[9] Myung, I & Pitt, M, Applying Occam's razor in modeling cognition: a Bayesian approach. *Psychonomic Bulletin & review*. 1997, 4(1), 79-95.

[10] Tornay, S. C. 1938. Occam: Studies and Selections. La Salle, IL: Open Court.

[11] Murray,I & Ghabramani, Z. A note on the evidence and Bayesian Occam's razor. *Gatsby Unit Technical Report*. 2005

[12] Nemenman, I & Bialek, W. Occam factors and model independent Bayesian learning of continuous distribution. *Physical review,* 2002, 65, 6137-1

[13] Kemp, C., Bernstein, A., and Tenenbaum, J. B. (2005). A generative theory of similarity. *Proceedings of the Twenty-Seventh Annual Conference of the Cognitive Science Society.*

[14] Harold Jeffreys Theory Of probability. Oxford, Clarendon Press, 1961.

[15] Chomsky, Noam (Sept. 1956). "Three models for the description of language". *Information Theory, IEEE Transactions* 2 (3).

[16] Chomsky, N. *Aspects of the Theory of Syntax*. MIT Press, 1965

[17] Chomsky., Noam A Review of B. F. Skinner's Verbal Behavior *Language*, Vol. 35, No. 1. (1959), pp. 26-58.

[18] J. B. Tenenbaum, T. L. Griffiths (2001), Generalization, similarity and Bayesian inference. *Behavioral and Brain Sciences*, 24 pp. 629-641.

[19] J. Rissanen (1978) Modeling by the shortest data description. *Automatica 14*, 465-471.

[20] J. B. Tenenbaum, T. L. Griffiths (2001). Some specifics about generalization, *Behavioral and Brain Sciences*, 24, pages 772-778.

[21] J. B. Tenenbaum (1995),Learning the structure of similarity. *Advances in Neural Information Processing Systems 8.* Toretzky, D., Mozer, M., and Hasselmo, M. (eds). Cambridge, MIT Press, 1995, 3-9.]

[24] Lisa A. Haverity, Kenethh R. Koedinger Solving inductive reasoning problems in Mathematics: Not-so-Trivial Pursuits. *Cognitive Science: A multidisciplinary Journal*., 2000. 24(2), 249 -298

[25] Sato, T., Kameya, Y., Abe, S., and Shirai, K. Fast EM learning of a family of PCFGs. Technical Report TR01-0006., Tokyo Institute of Technology, May, 2001.

[26] Hyun-suk Yoon. Optimization Approaches to Protein folding. Ph.D. *Thesis*, GIT. 2006

[27] Nir Friedman, Michal Linial, Iftach Nachman, Dana Pe'er. Using Bayesian Networks to Analyze Expression Data. *Journal of Computational Biology*.

August 1, 2000, 7(3-4): 601-620.

[28] Kevin P. Murphy, Learning Bayes net structure from sparse data sets (2001)

[29] Smith, Tony C. and Cleary, John G. (1997) "Probabilistic Unification Grammars." *Workshop notes: ACSC '97 Australasian Natural Language Processing Summer Workshop*, pp25-32. Macquarie University, Sydney, February.

[30] L. De Raedt, A. Kimmig, H. Toivonen. ProbLog: A Probabilistic Prolog and its *Application in Link Discovery*. In Proc. 20th IJCAI, 2007, in Press.

[31] Rochel Gelman & Brian Butterworth. Number and language: how are they related? *Trends in Cognitive Scienc*e. 2006. 9(1) 6-10.

[32] Alan M. Leslie, Rochel Gelman and C.R. Gallistel, The generative basis of natural number concepts. *Trends in cognitive sciences*. 2008. 12(6) 213-218.

[33] Stephen Muggleton: Learning Stochastic Logic Programs. *Electron Trans Artif. Intell.* 4(B).

[34] Goldsmith, John. 2007. Towards a new empiricism 1.6. MS, U. Chicago.

[35] Sato, T., Abe, S., Kameya, Y., and Shirai, K.: A Separate-and-Learn Approach to EM Learning of PCFGs. *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium*, pp.255–262, 2001

[36] George A. Miller. The Magical number seven, plus or minus two: some limits on our capacity of processing information. *The Psychological Review.* 1956 63(2)

[37] Michael Sipser. Introduction to the Theory of Computation, PWS Publishing Company,