Getting the Picture

Robert Akerlof, Richard Holden, and Hongyi Li*

May 9, 2024

Abstract

In the early 20th century, Gestalt psychologists seriously challenged prevailing notions regarding human perception. They showed that there is a difference between seeing the pixels that make up a picture and understanding what a picture represents. We have all had that "aha" moment, for instance, where a scene suddenly becomes clear (e.g. "oh, it's a smiley face"). The more general point is that people may have all of the information needed to draw a conclusion yet—in contrast to standard economic models—they fail to connect the dots. We build a model that conceptualizes this idea. An agent's task is to learn whether a picture possesses some feature (such as whether it depicts a smiley face). They have a knowledge set consisting of "codewords" that they think apply to the picture. This set initially contains codewords for each pixel's color, but no codewords describing the larger picture. The agent adds to their knowledge set by loading existing codewords into working memory and drawing conclusions. Importantly, the agent has limited working memory, which bounds their ability to draw conclusions. We show that the model captures a number of important phenomena, such as multi-stable perception, and provides a useful conceptualization of narratives as "big-picture statements." We explore several applications, including to the politics of persuasion.

JEL Classification: D01, D80, D90.

Keywords: Cognition, reasoning, perception, narratives.

*Akerlof: University of Warwick, r.akerlof@warwick.ac.uk. Holden: UNSW Business School, richard.holden@unsw.edu.au. Li: UNSW Business School, hongyi@unsw.edu.au. We are grateful to Sharun Mukand and Raghav Malhotra for helpful conversations, as well as seminar participants at Monash University.

1 Introduction

In the early 20th century, a new school of psychology emerged that challenged prevailing notions regarding human perception. The so-called Gestalt theorists, such as Max Wertheimer, realized that there is a fundamental difference between perceiving the *parts* of a picture and perceiving its *whole*. People may struggle to see the big picture. We have all had that "aha" moment where a scene suddenly becomes clear (e.g. "oh, it's a smiley face"). In addition to the struggles all people face with seeing the whole, neuroscientists have classified a variety of disorders ("agnosias") such as *face blindness* ("prosopagnosia"), where individual facial features—eyes, nose, and mouth—are recognizable but it's difficult or impossible to put a name to the face. As Gazzaniga et al. (2014) put it, "it's somewhat like going to Legoland and—instead of seeing the integrated percepts of buildings, cars, and monsters—seeing nothing but piles of legos." On the flipside, Gestalt psychologists noticed that people may integrate the parts into a whole in multiple ways. The famous duck-rabbit illusion (Figure 1) is a prime example. Some people initially see a rabbit, others a duck. With prolonged viewing, the interpretation may switch back and forth, a phenomenon known as multi-stable perception.

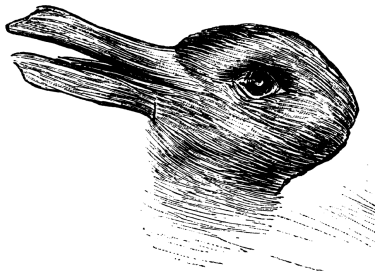


Figure 1: Rabbit-Duck Illusion

The main focus of the Gestalt psychologists was visual perception, but their insights apply more broadly. In many domains, people have all of the information needed to draw a conclusion yet fail to connect the dots. Consider Fermat's Last Theorem, which remained unproven for over three centuries despite the best efforts of mathematicians.

This contrasts sharply with standard economic models, which assume that agents have perfect information processing capabilities and can immediately derive all logical consequences of their knowledge.

This paper develops a framework for understanding how people analyze pieces of information and reason their way to a big picture. We also aim to understand how a big picture—which may be profitably termed a "narrative"—informs an agent's view of the parts.

We consider a model in which an agent is presented with a picture consisting of a grid of black and white pixels. Their task is to identify whether the picture possesses some feature—such as whether it depicts a smiley-face. The agent uses codewords to represent features of pictures. For instance, the agent has codewords corresponding to "the top-right pixel is black" and "the picture depicts a smiley face."

The agent has a knowledge set—to which they may add over time—consisting of codewords that they think apply to the picture and its regions. The agent's initial knowledge set consists of codewords for each pixel's color—but no codewords that apply to the larger picture. In this sense, the agent starts with complete information about the picture but no understanding of what it represents (e.g. "it's a smiley-face").

The agent adds codewords to their knowledge set by loading existing codewords into *working memory* and drawing conclusions. For instance, if the agent loads into working memory "the number of white pixels is even" and "the number of white pixels is prime," they draw the conclusion that "there are two white pixels."

Importantly, the agent has limited working memory, which restricts the number of codewords they can load. If the agent could load their knowledge of every pixel's color into working memory, they could immediately deduce all of the codewords that apply to the picture. However, limited working memory bounds their ability to draw conclusions.

We consider two variants of the model. In the first, the conclusions the agent draws are purely deductive, while in the second, the agent extrapolates. The pure-deduction model yields two key insights. First, the agent may develop only a piecemeal understanding of

the picture. They may, like a person with face blindness, recognize various parts of the picture but fail to integrate them and see the whole.

Second, there are some conclusions that can only be reached in multiple steps. The reason the agent can see more in two steps than one is that the agent can use the first step to "chunk" information: combine knowledge represented by multiple codewords into one codeword. Chunking allows the agent to store more information in working memory—and thereby deduce more. We use the term "chunk" in light of a closely related literature in cognitive psychology, initiated by William Chase and Herbert Simon.

In the version of the model with extrapolation, the agent adopts codewords when they "fit the data" sufficiently well—in the sense that there are relatively few alternative explanations. Extrapolation expands the set of conclusions that can be reached with limited working memory or limited initial information; but it also introduces the possibility of mistakes.

When the agent extrapolates, multi-stable perception is a possibility. To see why, consider the rabbit-duck illusion (Figure 1). Depending upon which part of the picture the agent processes first, the agent can get into one of two steady states. In one, the agent applies a "rabbit" codeword to the full picture and "rabbit ears" to the left-hand side. The "rabbit ears" codeword and the "rabbit" codeword mutually reinforce each other. For instance, the presence of "rabbit ears" in working memory blocks the adoption of alternative explanations to "rabbit"—such as "duck." In the other steady-state, the agent applies a "duck" codeword to the full picture and "duck's bill" to the left-hand side. These codes, likewise, are mutually reinforcing.

More generally, we refer to the coding of the parts of the picture (e.g. "rabbit ears" or "duck's bill") as *mental scaffolding*. Mental scaffolding supports the agent's overall interpretation of the picture—or "narrative." This scaffolding, once formed, is hard to remove—which lends stability to the agent's perception. When the agent sees a rabbit, in other words, it becomes hard to see a duck.

The model also suggests that agents gravitate towards *simple* narratives. In fact, a sim-

ple narrative that is *wrong* may be adopted over a complex narrative that is *right*. Simple narratives are attractive because they conserve working memory; consequently, it is easy to employ them and see their consequences. Politicians especially understand the importance of simple narratives. Virtually every successful politician has a memorable line that distills their philosophy. Ronald Reagan, for instance, is known for: "Government is not the solution to our problem, government is the problem." As the political consultant Frank Luntz puts it: "the most memorable political language is rarely longer than a sentence."[1]

In this paper, our initial focus, like the Gestalt psychologists, is on visual perception; but the model provides a general way of thinking about how agents aggregate and make sense of information. "Pixels" can be any type of data the agent observes. We show, for instance, how the model can be extended to capture a causal inference task in which "getting the picture" involves discerning (i.e. building a theoretical understanding of) the relationship between actions and outcomes.

We conclude the paper with a discussion of potential applications—including to multi-agent settings. In a standard persuasion model, influencing another agent involves controlling the information they receive. Our model suggests that it also matters how an agent *processes* information. Supplying an agent with a simple narrative, for example, might affect whether they interpret the data they receive one way or another (e.g. as rabbit or duck). Another interesting setting is one where agents encode features into memory differently—which leads them to see different things in the same data.[2] Such agents can benefit from back-and-forth communication in which each shares insights with the other.

There is, of course, a vast body of work in economics on the limitations of human reasoning. Simon (1955) famously spoke of economic agents as being "intended rational" but boundedly so. Simon explicitly pointed to "limits on computational capacity" as an important constraint on "actual human choice." The highly influential line of work

[1] Frank Luntz, "Words That Work: It's Not What You Say, It's What People Hear" (2007), p. 7.

[2] Depending upon the agent's coding system, storing a given feature in working memory can take up more or less space. As a result, the coding system affects what the agent is able to see.

on "systematic biases" beginning with Kahneman and Tversky[3], and developed further by Thaler, emphasizes the idea that (to use an aphorism of Thaler's): "it is possible to strengthen economics by incorporating the idea that some people behave like humans, at least some of the time."[4] The key observation in this line of work is that "if errors are predictable, then departures from rational choice models can also be predictable" so that "it would be possible to improve the explanatory power of economics by adding psychological realism" (Thaler (2017)).

A comparatively recent strand of work by Shleifer and coauthors seeks to "get inside people's heads" (Shleifer (2010)). For instance, Mullainathan et al. (2008) shows how "coarse thinkers" can be exploited by those seeking to persuade them. Specifically, when decision-makers group different situations into categories and apply the same model of inference within category they may transfer information from a situation in which it is useful to one in which it isn't ("transference"). Coarse thinkers are also subject to "framing effects" where the persuader influences the analogy used by the decision-maker. A second strand of work in this vein involves salience (see, for instance, Bordalo et al. (2012), Bordalo et al. (2013a), Bordalo et al. (2013b)). Here, "bottom-up" thinking by a decision-maker focuses their attention on salient factors that may be prominent or surprising. This contrasts with the standard approach to bounded rationality, where a "top-down" thinker optimizes subject to a constraint that reflects a specific deviation from perfect rationality—such as imperfect memory (Mullainathan 2002 and Wilson 2014), limited information (Sims 2003), limited attention to decision-relevant variables (Gabaix 2014), or limited ability to reason about other agents' strategies (Stahl and Wilson (1994), Crawford and Iriberri (2007)). Psychologists have, in fact, long debated the extent to which humans are wired for top-down or bottom-up thinking.[5] The presence of bottom-up thinking means that what is salient can potentially be altered by external parties such as firms or politicians, to the advantage of that party. Relatedly, knowledge of past choices can inter-

[3]See, for instance, an early summary in Tversky and Kahneman (1974)

[4]This quote is contained in Thaler's poster on the ground floor of the Harper Center at the University of Chicago Booth School of Business.

[5]See Broadbent (2013) on the former and Deutsch and Deutsch (1963) on the latter.

act in interesting ways with bottom-up thinking (Bordalo et al. (2019) and Bordalo et al. (2020)).

Relative to this work, our contribution is to offer a theory of a central aspect of the human reasoning process: how one goes from seeing the parts to seeing the whole. Our theory highlights, in particular, the role that limited working memory plays in the reasoning process.

Our paper also relates to a literature on narratives and mental models. Schwartzstein and Sunderam (2021), for instance, conceive of an agent as choosing a mental model from an available set that is potentially influenced by a persuader. Eliaz and Spiegler (2020) conceive of narratives as causal interpretations of events (specifically, directed acyclic graphs). Bénabou et al. (2018) model the process by which narratives disseminate. Relative to this literature, we adopt a distinct definition: we think of narratives as understandings of the big picture.

The paper proceeds as follows. Section 2 presents a version of the model in which the agent is purely deductive. Section 3 considers the possibility that the agent engages in extrapolation as well. Section 4 extends the model to capture a causal inference task, where the agent develops an understanding of the relationship between actions and outcomes. Section 5 discusses a variety of applications, including to multi-agent settings. Section 6 concludes. Proofs not offered in the main text are contained in the appendix.

2 Deduction

2.1 Model

An agent is presented with a black-and-white picture P. The picture consists of an $M \times N$ matrix of binary-valued variables p_{xy} that we call pixels. Each pixel is either black or white ($p_{xy} \in \{\text{black}, \text{white}\}$).

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & \cdots & p_{1N} \\ p_{21} & p_{22} & p_{23} & \cdots & p_{2N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{M1} & p_{M2} & p_{M3} & \cdots & p_{MN} \end{bmatrix}$$

As shown in Figure 2, we will use R to denote a particular region of the picture and P_R to denote the corresponding subpicture.

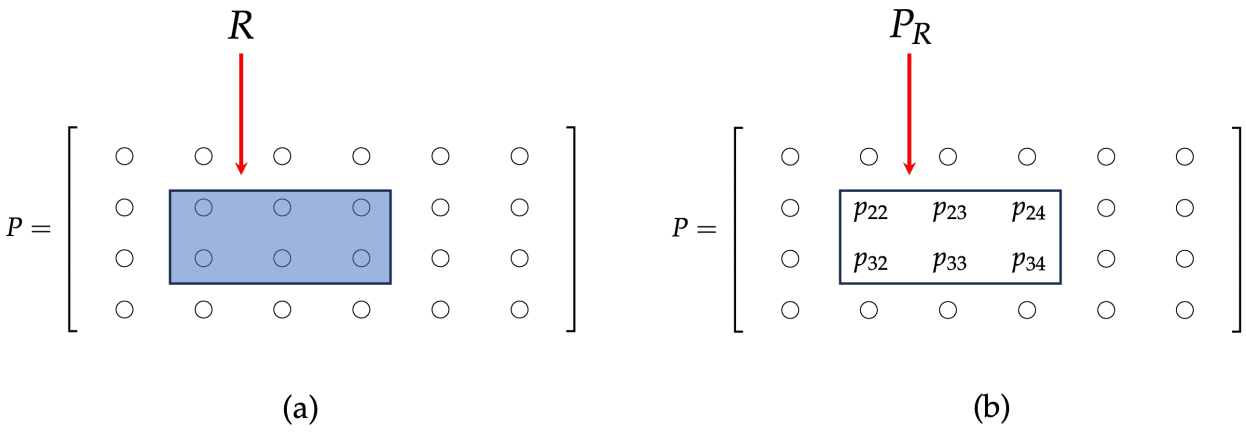


Figure 2

The agent's task is to work out whether the picture possesses some feature or set of features. For example, the agent might be shown the picture in Figure 3 and asked whether it depicts a smiley face.

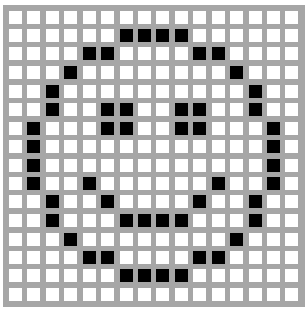


Figure 3: Example of a Picture

When we say that a feature f applies to picture P—or subpicture P_R—what we mean

is that the picture (or subpicture) belongs to a particular set. For instance, f might be the "smiley face" feature; saying that f applies to P means that P belongs to the set of smiley-face pictures. Formally, we define features as follows.

Definition 1.

1. *Let $\mathcal{P}_{m\times n}$ denote the set of all pictures/subpictures of size $m \times n$ (that is, all black-and-white matrices of size $m \times n$). We refer to any proper, non-empty subset of $\mathcal{P}_{m\times n}$ as an "$m \times n$ feature."*

2. *We say that feature f "applies to P_R" or "applies to region R of P" if $P_R \in f$.*

The agent uses *codewords* to represent features, mapping each feature f to a codeword c_f. We refer to this mapping as the agents "encoding" of features. These codewords are binary strings; for instance, "01101" might be the codeword for "smiley face." For expositional ease, codeword c_f must uniquely identify f amongst the set of $m \times n$ features—but not all features (in other words, we allow the agent to have a distinct code for each size of region). However, our conclusions hold regardless of whether codes are size-specific or there is a single code.

The agent has a *knowledge set* consisting of codewords that they think apply to the picture and its regions. The agent may add codewords to this knowledge set over time. We denote the agent's knowledge set at time t by:

$$K_t = \{K_t(R)\}_{R\in\mathcal{R}},$$

where $K_t(R)$ denotes the set of codewords for region R, and $\mathcal{R}$ denotes the set of all regions of P.

The agent's initial knowledge set K_0 consists of codewords for each pixel's color. In other words, the agent possesses all of the codewords that apply to 1x1 regions of the picture, but no codewords for larger regions. In this sense, the agent starts with complete

information about the picture but no understanding of what it represents (e.g. "it's a smiley face").

The agent adds codewords to their knowledge set by loading existing codewords into *working memory* and drawing conclusions. For example, suppose the agent loads the following two codewords for P into working memory: "the number of white pixels is even" and "the number of white pixels is prime." The agent concludes that "there are two white pixels."

Formally, at time t, the agent loads some of their knowledge into working memory:

$$W_t = \{W_t(R)\}_{R \in \mathcal{R}},$$

where $W_t(R) \subseteq K_t(R)$. From the codewords in working memory, the agent deduces that a set of codewords $\Delta_t(R)$ apply to region R.[6] At time $t+1$, these deducible codewords are added to the agent's knowledge set: $K_{t+1}(R) = K_t(R) \cup \Delta_t(R)$.

Importantly, the agent has limited working memory, which restricts the number of codewords they can load. The agent has the following working memory constraint:

$$length(W_t) \leq L,$$

where $length(W_t)$ is the total length of the codewords in working memory and L is the agent's working memory capacity. Shorter codewords (e.g. "0") take up less working memory than longer codewords (e.g "1101"). In this sense, we can think of features of the picture with shorter (longer) codewords as features that are simpler (more complex) for the agent. Whether a feature is simple or complex depends, of course, on how the agent encodes it.

We allow the agent to encode features in many different ways; but we make the as-

[6]Formally, we define $\Delta_t(R)$ as follows. For a feature f on R, let f^P denote the implied feature of the full picture: $f^P = \{\hat{P} \in \mathcal{P}_{M \times N} : \hat{P}_R \in f\}$. Let $f(c, R)$ denote the feature on region R corresponding to codeword c. Let $f_t^\Delta = \bigcap_{R \in \mathcal{R}} \bigcap_{c \in W_t(R)} f(c, R)^P$, which is the finest feature that is deducible on the full picture P. Given a codeword c on region R, we will say that $c \in \Delta_t(R)$ if $f(c, R)^P \supseteq f_t^\Delta$ and $c \notin K_t(R)$.

sumption that the agent's encoding is *efficient*. That is, there is no alternative encoding with the property that every codeword is weakly shorter and one codeword is strictly shorter. Since "white pixel" and "black pixel" are the only two features for 1x1 regions, efficiency implies that these codewords have length one (i.e. one feature is assigned codeword "1" and the other "0"). Hence, the parameter L corresponds to the number of pixels the agent can store in working memory.

2.2 The Agent's Problem

First, let us describe the agent's task in greater detail. The agent's task, denoted $\tau = (F, \hat{\mathcal{P}})$, is defined by a particular set of features (F) and a particular set of pictures ($\hat{\mathcal{P}}$). For each picture $P \in \hat{\mathcal{P}}$, the task is to identify all features $f \in F$ that apply to P (i.e. all f for which $P \in f$).

Next, let us discuss what the agent chooses. We opt to view the agent's encoding of features as something with which they are endowed—rather than a choice—although, as we will discuss later, there is reason to think the encoding is affected by past experience. We have in mind that W_t (what the agent loads into working memory) is potentially a choice, but we do not consider the usual optimization problem over W_t.[7]

To elaborate: the standard approach to modeling the agent's choice of W_t would be to consider an optimization problem where $W_0, W_1, ...$ are chosen to maximize some appropriate objective function (subject to the limited working memory constraint and the constraint that $W_t \subseteq K_t$) given P (or a prior over P). However, solving this optimization problem would entail perfectly anticipating (or forming correct expectations over) which "big picture" conclusions would be reached at the end of the deductive process (e.g. whether it is a smiley face)—which we consider to be inconsistent with our premise that the agent initially does not understand anything about the picture beyond seeing the individual pixels.

[7] W_t might also be influenced by the agent's environment. For instance, a conversation with another agent might impact what the agent considers.

Consequently, we are less ambitious in pinning down the agent's choice of W_t. Instead, we ask whether a given task $(F, \hat{\mathcal{P}})$ is *achievable*—specifically, whether for each $P \in \hat{\mathcal{P}}$ and each $f \in F$ that applies to P, there exists a working memory sequence $W_0, W_1, ..., W_t$ such that $f \in K_{t+1}$.[8] While this approach does not tell us what W_t's the agent chooses, it tells us whether the agent will complete the task.

An alternative approach one might take is to ask whether there is an algorithm the agent could employ for choosing W_t with desirable characteristics (i.e. an algorithm that performs well over a large set of pictures). This is an interesting question—but one that is beyond the scope of this paper.

2.3 What can be deduced?

Having outlined our approach to the agent's problem, let us consider what the agent is capable of deducing—as well as how long deductions take. We start with the following definition.

Definition 2.

1. *A codeword c is deducible in k steps ($c \in D_k$) if there exists a sequence of working memories of length k ($W_0, ..., W_{k-1}$) such that $c \in K_k$.*

2. *A codeword c is deducible ($c \in D$) if it is deducible in k steps for some finite k.*

In order to talk about learning through the deduction process, it is useful to define what it means for two codewords to be equivalent (i.e. contain the same information).

Definition 3.

1. *Codeword c on region R is "equivalent" to codeword c' on region R' if the codewords imply the same restrictions on the picture. Formally:*

$$\{\hat{P} \in \mathcal{P}_{M\times N} : \hat{P}_R \in f\} = \{\hat{P} \in \mathcal{P}_{M\times N} : \hat{P}_{R'} \in f'\},$$

[8] We consider the task to be trivially achievable for P if no $f \in F$ apply to P.

where f and f' are the features corresponding to c and c'.

Let us also define what it means for the agent to know "everything" (i.e. have a complete understanding of the picture's features).

Definition 4. *We say that the agent knows "everything" if, for any feature f that applies to the picture, the agent knows an equivalent feature.*

We obtain the following proposition.

Proposition 1.

1. *If $L \geq M \times N$: everything is deducible in a single step.*
2. *If $L = 1$: the agent cannot deduce anything (i.e. every codeword in D is equivalent to some codeword in K_0).*
3. *If $1 < L < M \times N$, for some pictures and encodings:*
    - *The agent can deduce something, but less than everything.*
    - *There are codewords that are only deducible in multiple steps.*

Part 1 of the proposition says that if the agent has sufficient working memory ($L \geq M \times N$), everything is deducible in a single step. Intuitively, if $L \geq M \times N$, the agent can store the codeword for every pixel's color in working memory. If the agent has every pixel loaded in memory, they can deduce every codeword. Therefore, only when $L < M \times N$ does working memory bound the agent's ability to make deductions.

Part 2 of the proposition says that if the agent has just one unit of working memory ($L = 1$), the agent cannot deduce anything. The reason is that the agent learns the big picture by *combining* or *integrating* codewords. If $L = 1$, they can store only a single codeword in working memory.

To understand Part 3, consider Figure 4, which shows a picture P consisting of a 2x2 grid of white pixels. First, let us use the figure to show that there are deductions that take

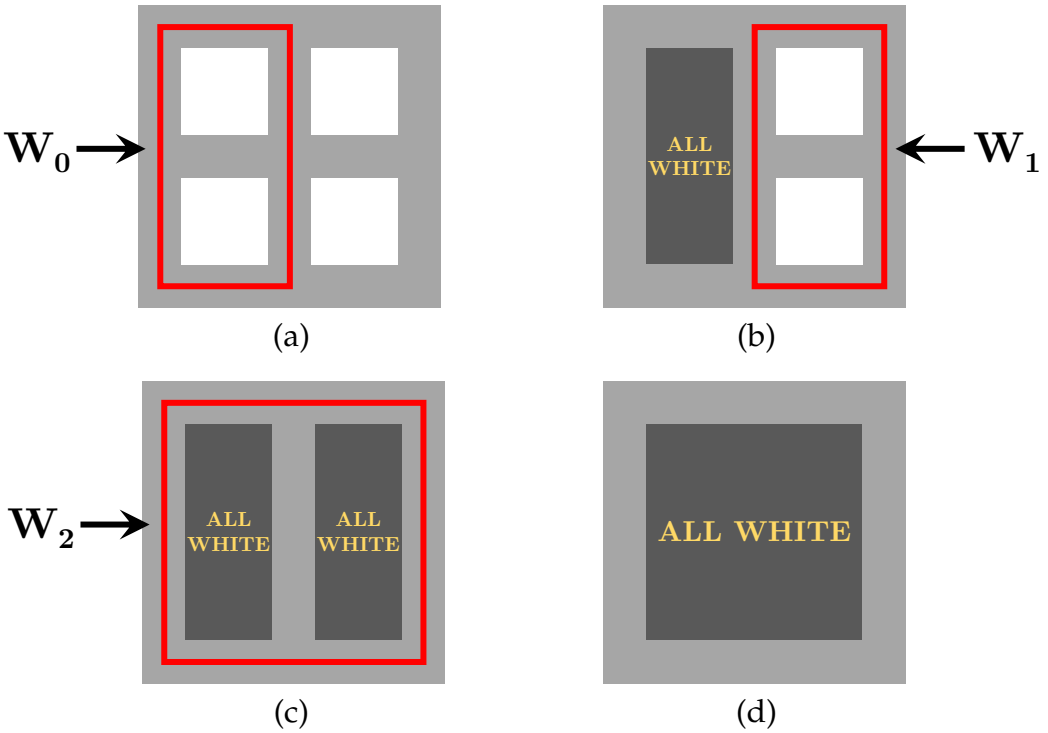


Figure 4: A Multi-step Deduction

multiple steps. Suppose the agent's working memory capacity is two ($L = 2$). Because the agent can only load two of the four pixels of P into working memory, they cannot deduce in a single step that a 2x2–"all white" codeword applies to P. It may be possible to deduce the 2x2–"all white" codeword in multiple steps, however.

Let R_L denote the left-hand side of P and R_R denote the right-hand side. The agent can load their knowledge of the two pixels on the left side (panel a) and deduce that an "all white" codeword applies to R_L (panel b). The agent can do the same for the right side (panels b and c). Suppose the 2x1–"all white" codeword that the agent applies to the 2x1-regions R_L and R_R has length one. Then, the agent has sufficient memory capacity to load the 2x1–"all white" codewords for R_L and R_R and deduce that an 2x2–"all white" codeword applies to P (panel d).

Now, let us use Figure 4 to show that the agent may be able to deduce something but not everything. If the "all white" codeword for 2x1 regions is longer than one, the agent cannot simultaneously load the 2x1–"all white" codewords for R_L and R_H: this path to

deducing a 2x2–"all white" codeword for P is blocked. There is one alternative path the agent could use, focusing on the top and bottom of the picture (R_T and R_B) rather than the left-hand side and right-hand side. However, this path is also blocked if the 1x2–"all white" codeword for 1x2 regions is longer than one. Thus, if both of these 1x2–"all white" codewords are longer than one, the agent cannot deduce everything; but they can still deduce that R_L, R_R, R_T, and R_B are "all white"—hence they can deduce something.

2.4 Chunking

If we examine the multi-step deduction depicted Figure 4, the key thing the agent does between steps is condense information. Specifically, the agent combines multiple codewords, with a high memory demand, into fewer codewords with a lower memory demand. For instance, the two pixels on the left-hand side of P are initially represented by two codewords—one codeword for each pixel—which require two units of working memory. These codewords are then combined into a single "all white" codeword, which requires only one unit of working memory.

We refer to this process of combining codewords as "chunking" in light of a closely related literature in cognitive psychology. The term "chunk" was introduced by Miller (1956), who made a remarkable discovery about short-term memory. Miller found that the ease or difficulty of remembering information depends upon how it is stored. For example, a telephone number can be stored as a string of individual digits (e.g. 5, 1, 2, 3, 4, 8, 2) or as a set of larger numbers, or chunks, (e.g. 512, 3482). Chunking the phone number, he discovered, significantly reduces the strain on memory.

William Chase and Herbert Simon further developed the concept of chunking in a famous 1973 paper.[9] They conducted an experiment in which expert and novice chess players were shown board positions for five seconds and then asked to reconstruct the positions from memory. When participants were presented positions from *actual* chess games, the experts significantly outperformed the novices. On average, experts correctly

[9]Chase and Simon (1973) build on earlier experimental results of De Groot (1965).

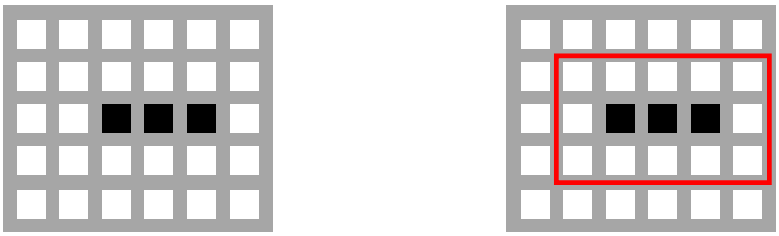
recalled the locations of 16 out of 24 chess pieces, while novices only recalled 4 out of 24. However, when participants were shown *random* board positions, there was no difference between experts and novices; furthermore, both groups performed even worse than the novices did on actual board positions.

Chase and Simon posited that expert players have a greater ability to chunk typical chess configurations. In terms of the model, we can think of experts and novices as having different encodings of features. Experts assign short codewords to common board positions, whereas novices do not. This allows experts to represent a board position with just a few short codewords, in contrast to novices.

(a) Game Board (b) Game Board - Chunked

Figure 5: Chunking a Chess Board

To make this point more concrete, consider Figure 5. Panel (a) shows a constellation of pixels that we might think of as akin to the position of pieces in a board game (obviously, a slightly different game from chess). Panel (b) shows that one way of representing this configuration is as an "H" and an "I." Suppose one agent (akin to the expert chess player) has short codewords for "H" and "I"—as they might if they are familiar with the Latin script. Say both codewords have length three. If this agent stores the board as an "H" and "I", it takes up just six units of memory—far less memory than the fifty units needed to store fifty individual pixels. Suppose a second agent (akin to the novice) has long codewords for "H" and "I." This agent may not find it any better to store the board as an "H" and an "I" than as fifty pixels.



(a) (b)

Figure 6: Example - Line Measurement

2.5 Detecting a Line and Measuring its Length

In order to build some further intuition about the model, let us consider a slightly more involved perception task. Suppose an agent's task is to detect whether a picture P, such as the one shown in panel (a) of Figure 6, contains a horizontal line surrounded by white space, and if so, to determine its length.

One way the agent could complete this task is by storing all of the relevant pixels. Panel (b) of Figure 6 shows all of the pixels the agent would need to store—fifteen in total—to detect the line of length three. Notice that the agent needs to store all of the pixels surrounding the line; otherwise, they might draw an incorrect conclusion about the line's length or whether it is a line at all.

This method of identifying horizontal lines requires a working memory capacity of $L = 3(n+2)$ for a line of length n—which is obviously quite large. Let us see whether there is a method that requires less working memory—in particular, a method where the memory requirement is not growing in n.

Figure 7 shows two types of picture features, which we denote by S_n and L_n respectively. We say that a $3 \times (n+1)$– region has feature S_n if it contains a line of length n and there is one column of white pixels immediately to the left of the line (see panel a). We say that a $3 \times (n+2)$–region has feature L_n if it contains a line of length n and there is one column each of white pixels on both sides (see panel b).

Features S_n and L_n could have long or short codewords—it all depends upon how the agent encodes features. In principle, though, the agent could use codewords of length

(a) Feature S_3 (b) Feature L_4

Figure 7: Two Features

one for both S_n and L_n features: for instance, by assigning each S_n with codeword "0" and each L_n with codeword "1", or vice versa.

Figure 8 shows how these two codewords can be used to complete the line measurement task. In panel (a), the agent stores six pixels in working memory and deduces that S_1 applies to the associated region. In panel (b), the agent stores codeword S_1 plus three additional pixels and deduces a codeword S_2. The agent continues until they reach the end of the line—at which point they deduce L_3, which corresponds to a line of length 3. If the codewords for S_n and L_n have length one, the point where working memory is most strained is panel (a), where the agent must store six pixels. Hence, a line of arbitrary length can be detected and verified with working memory capacity of $L = 6$. Observe

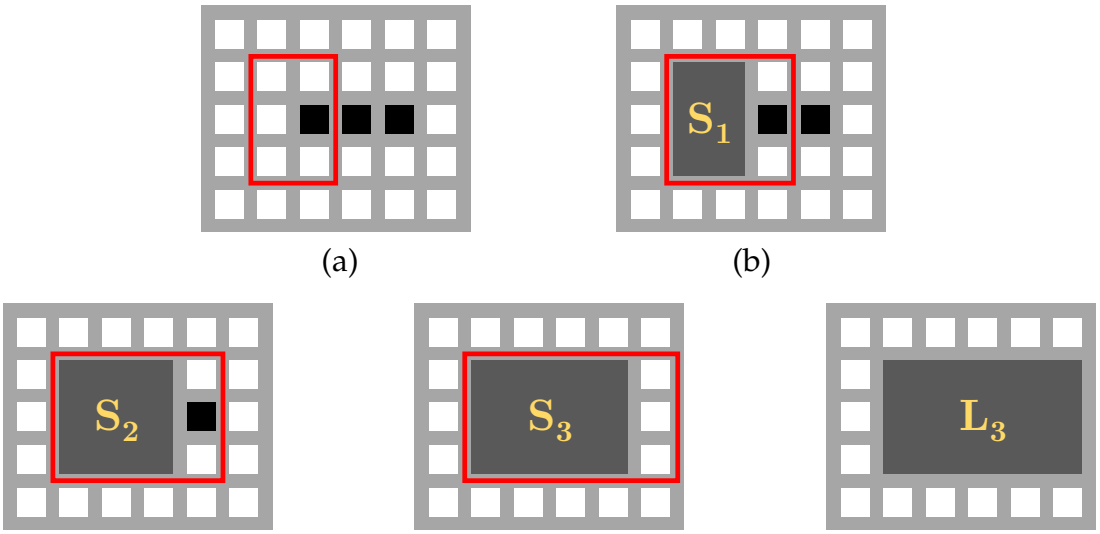


Figure 8: Line Measurement

that the working memory required is fixed now rather than growing in n.[10]

This example illustrates how the agent's encoding can generate more or less efficient strategies for detecting features. Assigning short codes to features S_n and L_n allows the agent to efficiently chunk lines—just as expert chess players are able to efficiently chunk and remember board positions.

2.6 Optimal Codes

As we noted in Section 2.2, we opt to think of the agent's code as something they are endowed with—rather than a choice. Nonetheless, there is reason to think that it evolves. Chase and Simon (1973)'s experiment provides suggestive evidence. Their results could be purely the result of selection (i.e. better chess players are endowed with better codes); but it seems more likely that better chess players have better codes because they have more experience with chess.

This raises a variety of questions—such as how codes evolve. However, an even more basic question is what we mean by a "better" code or an "optimal" code. Here, we give one possible definition of optimality and discuss its properties.

Recall that we defined a task in Section 2.2 by $\tau = (F, \hat{\mathcal{P}})$, where F is the set of features that need to be identified and $\hat{\mathcal{P}}$ is the set of pictures on which identification needs to be achievable. We denote the agent's code by $\mathcal{C}$. For a given task and code, let $L^{\min}(\tau, \mathcal{C})$ denote the minimum amount of working memory capacity needed for the task to be achievable.

We say that a code is τ-optimal if it minimizes the amount of working memory needed for the agent's task τ.

Definition 5. *Let $\mathcal{C}^*(\tau) = \arg\min_{\mathcal{C}}\{L^{\min}(\tau, \mathcal{C})\}$ denote the code (or set of codes) that minimize the working memory needed for task τ. We will refer to $\mathcal{C}^*(\tau)$ as the optimal code(s) for task τ.*

[10] In a version of the model where the agent has a single code—rather than a code for each size of region—the same incremental deductive process can still be used to save working memory. However, the savings are slightly less since each S_n and each L_n requires its own codeword. In this case, the required working memory would instead be on the order of $\log n$.

It is relatively easy to show that the optimal code depends upon the task, which we state as Proposition 2.

Proposition 2. *The optimal code is task-specific. That is, for some tasks τ and τ', $\mathcal{C}^*(\tau) \cap \mathcal{C}^*(\tau') = \varnothing$.*

Proof. See appendix. □

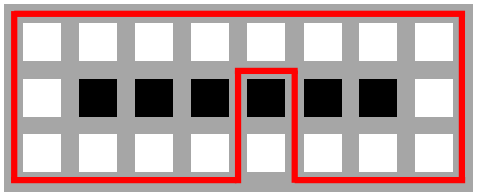
To gain intuition for this result, recall some of the tasks we have considered. In Figure 4, the agent's task is to identify whether a picture is "all white." In Figure 5, the agent's task is to identify board positions composed of letters. In Figure 6, the task is to detect a line and measure its length. In each case, we showed that working memory is conserved by assigning short codewords to particular features. In the first case, it makes sense to assign a short codeword to "all white" regions. In the second case, it is ideal to assign short codewords to the letters "H" and "I." In the line identification task, it makes sense to assign S_n and L_n short codewords.

Recall that the number of short codewords is limited. For instance, there are only two codewords of length one ("0" and "1"). Consequently, a code that is geared towards the first task (detecting all-white pictures) is less geared towards the second task (detecting letters).

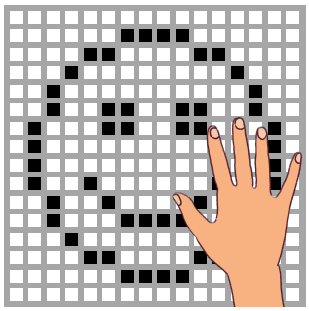
This result suggests that agents may have different competencies. One agent might have a code that leads them to excel at chess, while another might have a code that leads them to excel at Go. We will discuss later how the model might be profitably extended to environments where agents have different competencies.

3 Extrapolation

So far, we have assumed that the agent uses deduction alone to learn about the picture. Deduction, in some cases, may be a powerful tool for learning about a picture; but in other cases, it may not take the agent very far. When the agent struggles to learn about a picture through deduction alone, extrapolation may take them further.



(a) Limited working memory.



(b) Limited observation.

Figure 9: Benefits of Extrapolation

Figure 9 shows two cases where extrapolation is useful. In panel (a), the agent's task is to detect whether the picture contains a horizontal line. Their working memory capacity is twenty-two pixels; if they could load just two more pixels, they could deduce that the picture contains a horizontal line. To keep the example simple, suppose loading pixels is the only way the agent can detect the line. It might make sense in this case for the agent to extrapolate (i.e. decide that there is a line even though they cannot be sure).

In panel (b), the agent is not able to see all of the pixels of the picture. Formally, we can think of this as a case where the agent's initial knowledge set K_0 contains only some of the pixels of P. Even if the agent has unlimited working memory, they cannot deduce that the picture depicts a smiley face; nonetheless, this would be a reasonable conclusion to draw. Thus, here too, extrapolation has value.

We now consider a version of the model where the agent makes extrapolations, adding codewords to the knowledge set whenever they "fit" sufficiently well with the facts in working memory.

3.1 Model

As in the deduction model, the agent loads a subset of their knowledge into working memory at time t ($W_t \subseteq K_t$). In addition, the agent considers a particular codeword c on region R for inclusion in, or exclusion from, the knowledge set (inclusion when $c \notin K_t(R)$

or exclusion when $c \in K_t(R)$).

The agent bases their decision to include/exclude codewords on whether their "fit" with the facts in working memory is high or low. We denote codeword c's fit by $\Phi_R(c, W_t)$ (we will say more about how this fit function is defined momentarily). The agent includes codeword c if fit weakly exceeds a threshold α and excludes codeword c if fit is weakly below a threshold β (with $0 \leq \beta < \alpha \leq 1$). Formally, $K_{t+1}(R) = K_t(R) \cup \{c\}$ if $\Phi(c, W_t) \geq \alpha$ and $K_{t+1}(R) = K_t(R) - \{c\}$ if $\Phi(c, W_t) \leq \beta$.

The fit function we adopt takes the following form.[11] We define $Z_R(W_t)$ as the number of subpictures on region R that are consistent with W_t. We define $Z_R(W_t, c)$ as the number of subpictures on region R that are consistent with W_t and c.[12] The fit of code c is the ratio of these quantities:

$$\Phi_R(c, W_t) = \frac{Z_R(W_t, c)}{Z_R(W_t)}.$$

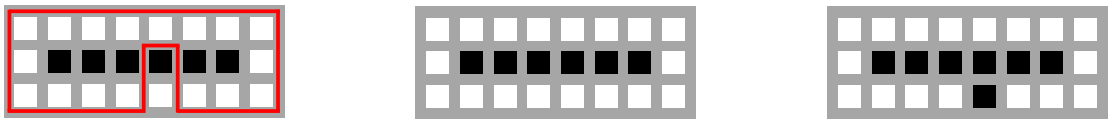
In the special case where no picture is consistent with the "facts" ($Z_R(W_t) = 0$), which is possible given that the agent extrapolates—hence their "facts" may be wrong—we assume that fit is equal to zero.

According to this definition, fit depends upon the fraction of subpictures that have feature c among those that meet the facts in working memory. Notice that deduction is equivalent to adopting codewords only when they have perfect fit ($\Phi_R(c, W_t) = 1$).

To get a better sense of how the fit function works, consider Figure 10, which expands on the horizontal line example. Panel (a) shows the agent's working memory, which contains twenty-two of the picture's twenty-four pixels. Panels (b) and (c) show two codewords, c_1 and c_2, the agent is considering applying to P. Observe that $Z_P(W_t) = 4$: there are four pictures consistent with W_t since two pixels are not pinned down by W_t. There is a single picture consistent with c_1 and W_t, so $Z_P(W_t, c_1) = 1$. Likewise, there is a single picture consistent with c_2 and W_t, so $Z_P(W_t, c_2) = 1$. Thus, both c_1 and c_2 have fit

[11] The appeal of the fit function we adopt here is its simplicity; but there may be other fit functions worth considering.

[12] By "consistent with W_t (W_t and c)" we mean that a subpicture satisfies all of the conditions in W_t (all of the conditions in W_t, as well as c).



(a) Working memory (W_t). (b) Codeword c_1. (c) Codeword c_2.

Figure 10: Example - Fit Function

of $\frac{1}{4}$. Codeword c_2 does not actually apply to the picture; but it still has high fit since it is consistent with the facts in working memory.

In contrast to the deduction model, we assume that both the facts the agent loads (W_t) and the codeword the agent considers for inclusion/exclusion (c) take up space in working memory. Thus, the agent's working memory constraint is:

$$length(W_t) + length(c) \leq L.$$

Notice that this constraint tends to privilege short codewords for inclusion in knowledge. To illustrate, consider codewords c_1 and c_2 from Figure 10. Both have the same fit; but perhaps the horizontal line codeword, c_1, is short and c_2 is long. It might be hard for the agent to stick c_2 into working memory, and this might favor the adoption of c_1.

Thus, we see a new way in which the agent's code matters. When the agent extrapolates, they tend to opt for extrapolations coded as simple over extrapolations coded as complex. If the agent thinks of lines as simple, the agent will look for lines; or if the agent thinks of faces as simple, the agent will look for faces.

3.2 The Agent's Problem

We will not discipline the agent's problem beyond the rules governing extrapolation. In particular, we do not impose a task upon the agent, and instead allow the reasoning process to unfold without reference to objectives. Put another way, we examine which sequences of knowledge sets $\{K_t\}_{t=1,2,...}$ are achievable—given the agent's initial knowl-

edge K_0, code $\mathcal{C}$, working memory constraint L, and the fitness parameters α and β.

Consider that under extrapolation, the agent's knowledge set may either expand (as codewords are added) or shrink (as codewords are removed)—unlike the case of deduction, where codewords are only ever added. Consequently, unlike under deduction, a long-run steady-state may never arrive. Instead of simply examining a snapshot of the agent's long-run knowledge set to check whether specific features are successfully learned—as our previous approach to analyzing deduction entails—our approach to extrapolation also examines the agent's knowledge *sequences*: whether they exhibit dynamics such as cycling where codewords are learned then unlearned, or path dependence in what knowledge is eventually acquired.

3.3 Useful Codewords

Some codewords may be more "useful" than others to the agent for the purposes of making deductions or extrapolations. Figure 11 provides an example. Suppose the agent knows that codeword "smiley-face" applies to picture P and they are considering what codewords to apply to region R. Clearly, "smiley-face" is a very useful codeword as it makes it easy to conclude that "two eyes" is a feature that applies to region R.

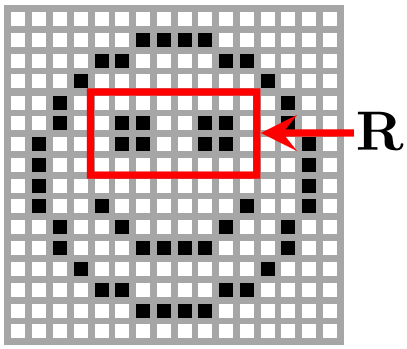


Figure 11: Example of Usefulness

More generally, we can think of a codeword as useful for a region R if it reduces the set of possible subpictures on R. Formally, we define usefulness as follows.

Definition 6. *We define codeword's c's "usefulness" for an $m \times n$-region R as:*

$$\Gamma_R(c) = 1 - \frac{1}{m \times n} \log_2 Z_R(c)$$

Usefulness takes the value zero when codeword c does not reduce the possibility set at all (since, if c does not reduce the set of subpictures, $Z_R(c) = 2^{m \times n}$). Usefulness takes the value one when codeword c perfectly pins down what happens on region R (since, in this case, $Z_R(c) = 1$).

As we have already mentioned, analyzing how the agent chooses W_t is a difficult problem. Nonetheless, *usefulness* is a feature that clearly makes a codeword attractive for inclusion in working memory. Another feature that makes a codeword attractive is *short length*. Put another way, the agent is likely to gravitate towards codewords that are simple and have a lot to say about what is going on. We capture this idea with the following assumption.

Assumption 1. *Suppose, at time t, the agent is evaluating a codeword c that applies to region R for inclusion/exclusion. Suppose $c' \neq c$ and $c'' \neq c$ are codewords in the agent's current knowledge set K_t; c' is weakly more useful than c'' on region R; c' is weakly shorter than c''; and at least one of these two inequalities is strict. If the agent loads c'' into working memory (W_t), then they also load c'.*

What Assumption 1 says is that there may be some codewords that the agent is highly prone to use once they are part of the knowledge set. If we return to Figure 11, for instance, Assumption 1 says that, once the agent has seen that the picture is a smiley face, it may be hard for them *not* to think of the picture as a smiley face.

3.4 Possible Outcomes

Let us examine the types of outcomes that can arise when the agent extrapolates. We start by defining three concepts.

Definition 7.

1. ***Path dependence****: given initial sequences of working memory (i) $W_0, ..., W_k$ and (ii) $W'_0, ..., W'_k$ we say that there is "path dependence" if there is a codeword c such that, for any subsequent working memory sequence, $c \in K_t(R)$ for $t > k$ under initial sequence (i) and $c \notin K_t(R)$ for $t > k$ under initial sequence (ii).*

2. ***Cycling****: given an initial sequence of working memories $W_0, W_1, ...$, we say that there is "cycling" if there is a codeword c such that $c \in K_t(R)$, $c \notin K_{t'}(R)$, and $c \in K_{t''}(R)$, with $t < t' < t''$.*

3. ***Patchwork quilt****: given an initial sequence of working memories $W_0, ..., W_k$, we say that there is a "patchwork quilt" if there are codewords $\{c_1, ..., c_l\}$ that apply respectively to regions $\{R_1, ..., R_l\}$ such that, for any subsequent working memory sequence, $c_j \in K_t(R_j)$ for all j and all $t > k$ and $\{c_1, ..., c_l\}$ are not mutually consistent (i.e. there is no picture P with all of these features).*

When there is path dependence, there are multiple stable perceptions. The agent might, for instance, see either a rabbit or a duck in Figure 1. When there is cycling, the agent flips between unstable perceptions. For example, the agent might see a rabbit, a duck, and then a rabbit again. When there is a "patchwork quilt," the agent has a piecemeal understanding of the picture, interpreting the pieces in inconsistent ways (e.g. part rabbit, part duck). It turns out that all of these are possibilities. We state this as proposition 3.

Proposition 3. *Under Assumption 1, there exist pictures P and codes $\mathcal{C}$ such that the following are possible: (i) path dependence, (ii) cycling, and (iii) patchwork quilts.*

Proof. See appendix. □

To gain intuition for this result, let us continue with the rabbit-duck example. Figure 12 shows how path dependence might arise. Suppose in early periods the agent puts

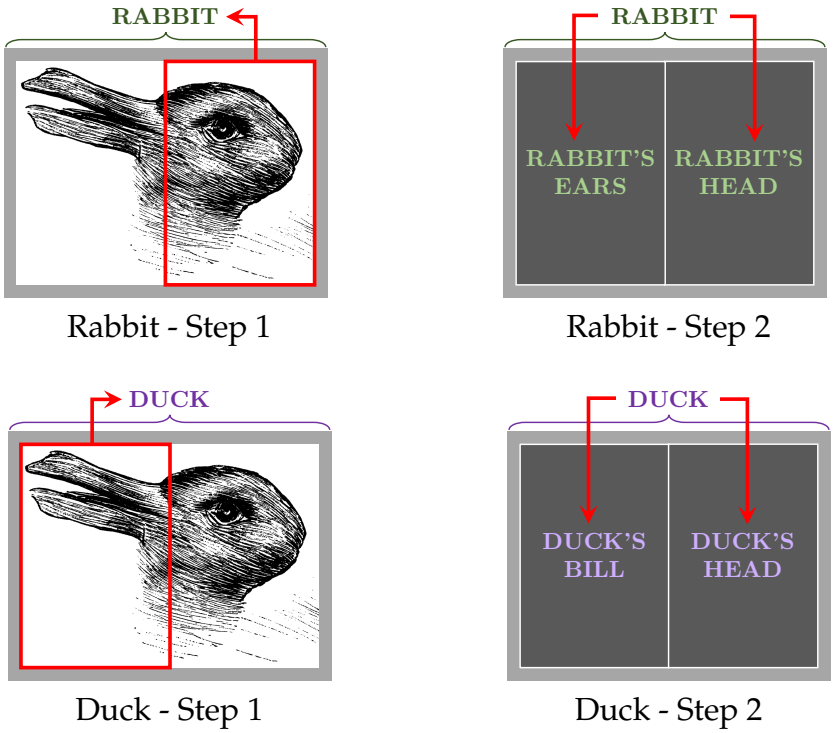


Figure 12: Example of Path Dependence

into working memory features of the right-hand side of the picture ("Rabbit - Step 1"). The right-hand side is arguably the more rabbit-like side; focusing on this side might lead the agent to extrapolate that the picture depicts a rabbit. "Rabbit" is presumably a useful codeword; imagine it is also short. Then, under assumption 1, the agent will keep "rabbit" in working memory as they continue analyzing the picture. Keeping "rabbit" in mind may then lead the agent to conclude that the left-hand side depicts "rabbit ears" and, if they have not concluded so already, that the right-hand side depicts a "rabbit's head" ("Rabbit - Step 2"). If, by contrast, the agent starts their analysis of the picture on the left-hand side, the agent may extrapolate that the picture depicts a duck ("Duck - Step 1"); the agent may then interpret the parts of the picture accordingly ("Duck - Step 2").

To understand why cycling might arise, consider Figure 13. Imagine the agent reaches a rabbit interpretation of the picture and its parts ("Step 1") by a process along the lines illustrated in Figure 12. In contrast to the previous case, however, suppose that there is some set of features from the left-hand side that, if loaded into working memory, leads

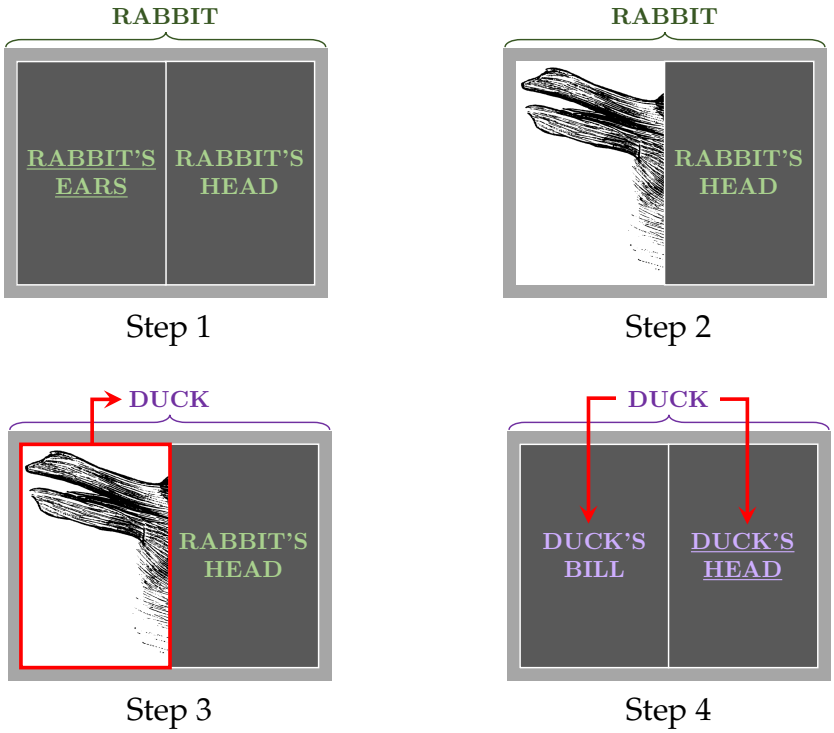


Figure 13: Example of Cycling

the agent to *exclude* "rabbit's ears" from the knowledge set—even if "rabbit" is loaded in working memory, per assumption 1 ("Step 2"). "Rabbit's ears" is, in this sense, an *unstable* interpretation of the left-hand side of the picture—which is why the codeword is underlined in Figure 13.

Eliminating "rabbit's ears" from knowledge may destabilize "rabbit." To see why, suppose that "rabbit's ears" was a short, useful codeword. Per assumption 1, it would have been loaded into working memory if the agent ever considered excluding "rabbit" from the knowledge set. This would have helped the agent maintain "rabbit" in the knowledge set. Suppose that, with the elimination of "rabbit's ears," the agent uses the left-hand side to eliminate "rabbit"—and also extrapolate to "duck" ("Step 3"). The "duck" codeword then leads the agent to reinterpret the parts of the picture accordingly ("Step 4"). Notice that the "duck's head" codeword could be unstable too. If so, the whole process can run in reverse, taking the agent back to "rabbit" ("Step 1").

Finally, consider how the agent might end up with a patchwork quilt. Suppose the

agent starts by loading features of the right-hand side into working memory, leading to a rabbit-like interpretation (e.g. "rabbit's head"). However, they might not extrapolate a "rabbit" code for the whole picture. Similarly, they might load features of the left-hand side and develop a duck-like interpretation (e.g. "duck's bill")—but again, they might not extrapolate a code for the whole picture. Notice that, even if "rabbit's head" and "duck's bill" are inconsistent, the agent might not realize the inconsistency. Realizing the inconsistency requires loading certain facts in memory—and this may exceed the agent's working memory capacity.

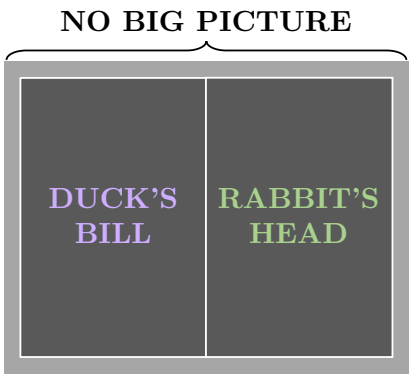


Figure 14: Example of a Patchwork Quilt

3.5 What does the picture actually depict?

Notice that we have not actually taken a stance on which features apply to the rabbit-duck picture. There are several potential answers one could give. Which answer one gives affects one's interpretation of the results.

1. The picture is either "rabbit" or "duck": one possibility is that one of the two features applies to the picture. For instance, maybe it is a "rabbit" but not a "duck" (despite looking duck-like). The implication of path dependence then is that the agent might land on a correct interpretation ("rabbit") or an incorrect one ("duck").

2. The picture is neither "rabbit" nor "duck": another possibility is that the picture is neither (despite looking rabbit-like and duck-like). The implication of path depen-

dence in this case is that there are multiple incorrect interpretations the agent can reach.

3. The picture is both "rabbit" and "duck": another possibility is that there are very few pictures that are both "rabbit" and "duck" but this picture happens to be one of them. The implication of path dependence here is that the agent only sees a partial truth. They see "rabbit" but not "duck," or "duck" but not "rabbit."

3.6 Mental Scaffolding

When there are two stable perceptions, as illustrated in Figure 12, a natural question is what gives them stability.

To see where stability comes from, suppose "rabbit" and "rabbit's ears" are short, useful codewords that, per Assumption 1, are both likely to be invoked. Then, if the agent considers removing the "rabbit's ears" codeword from the knowledge set, they will keep the "rabbit" codeword in mind—and vice-versa. This keeps the agent from rejecting either codeword. Effectively, the agent says to themself: "I know this is a rabbit, so these must be the ears" and "I know these are rabbit's ears, so this must be a rabbit." In this sense, "rabbit" and "rabbit's ears" are mutually reinforcing codewords. In the same way, "duck" and "duck's bill" are mutually reinforcing.

At a more general level, we can think about a "narrative" (i.e. codeword for the whole picture) that is supported by sub-narratives (i.e. codewords for parts of the picture). We refer to the sub-narratives that play this role as "mental scaffolding." Mental scaffolding is key to providing stability to the agent's overall narrative.

4 Causal Inference

Our initial focus is on visual perception because it provides useful intuition. However, "pixels" can be any type of data the agent observes; and the model can be used to consider any setting where an agent tries to make sense of data and understand its meaning.

Here, we illustrate this point by considering a simple extension of the framework to a setting where the agent is presented with k observations. Each observation i is an experiment undertaken by the agent where they take a set of actions $(a_{1i}, ..., a_{ni})$ and observe a set of outcomes $(o_{1i}, ..., o_{mi})$. We show how the model can be adapted to examine how the agent learns about the connection between actions and outcomes.

Let $V = \{v_i\}_{i=1}^{k}$ denote the k observations presented to the agent, where v_i denotes observation i. Observation i takes the following form:

$$v_i = \begin{bmatrix} a_{1i} \\ \vdots \\ a_{ni} \\ o_{1i} \\ \vdots \\ o_{mi} \end{bmatrix}.$$

We assume that each action and each outcome takes the value zero or one—since this is equivalent to pixels that are black or white—but this is easily generalized.

Observe that a vector is just a picture with a single column. Our definition of features for pictures carries over to vectors. In addition to features of vectors, we can define a new concept—a "common feature"—which we denote $\kappa = (f, S)$. We say that κ is "applicable to V" if f is a feature that is common to a subset S of the observations in V ($v_i \in f$ for all $i \in S$).

A special type of common feature is one that applies to all of the observations. We will refer to this special type of common feature as a "general feature." To illustrate, a general feature of V might be $a_{1i} = o_{1i}$. If the agent has this general feature in their knowledge set, it is akin to having a model of the world that says that a particular action (a_{1i}) determines a particular outcome (o_{1i}). Another type of general feature might be $o_{1i} = o_{2i}$. We can

think of this as a model of the correlation between certain types of outcomes.[13]

The equivalent of a "region R" in this case is a set S of observations; and the equivalent of region size is the cardinality of set S. We assume that for common features of a given cardinality, the agent has a code that uniquely maps those common features to binary strings (i.e. codewords). Notice that a common feature κ can capture a single element of a single vector i by setting $S = \{i\}$ and selecting f to represent a single vector element. Let us refer to such common features as "pixels." We assume that the agent's initial knowledge set K_0 consists of codewords for pixels only.

The agent learns in the same way they do in the original model. They add codewords from their current knowledge set to working memory ($W_t \subseteq K_t$), and they consider whether to include a codeword c in (or exclude a codeword c from) $K_{t+1}(S)$, where $K_{t+1}(S)$ denotes the common codes applied to S at time $t+1$. The same fit function ($\Phi_R(c, W_t)$) can be used again, only replacing R (region) with S (set).

If we allow some pixels to be missing from the agent's initial knowledge set K_0, this accomplishes three things. First, it captures the possibility that the agent is simply missing some data (i.e. there are outcomes or actions they failed to measure).

Second, we can use missing pixels to capture an agent who makes predictions. To illustrate, suppose there is a vector j where all of the actions are part of K_0 but none of the outcomes are part of K_0. Suppose the agent, through extrapolation, adds a general feature to their knowledge set such as $a_{1i} = o_{1i}$. This general feature is now a model that makes a prediction about the outcomes that will arise if the agent takes the actions in vector j (i.e. $o_{1j} = a_{1j}$).

Finally, allowing some pixels to be missing means that the agent's model can be probabilistic rather than deterministic. For example, suppose there is an action j that is not part of K_0 for any i (i.e. a_{ji} is not part of K_0 for any i). When the agent extrapolates, they can treat action j like an unobserved random variable that influences outcomes.

Recall that, in the original visual-perception model, we found that the agent may be

[13]Note that we can eliminate actions from the vectors in V and have outcomes only. In this case, the agent is building a purely correlational model of the world.

able to see something but not everything (see Proposition 1). Analogously, in this extension, the agent may understand some of the implications of their actions but not all of the implications. In addition, as in the rabbit-duck example, the agent may make mistakes. They may draw incorrect inferences about the consequences of their actions and find it difficult to see why their model of the world is flawed. We consider some of the consequences of this idea in the next section.

5 Discussion

Here, we discuss a variety of applications and possible extensions of the model.

5.1 Multi-agent Settings

Our focus thus far has been on an environment with a single agent; but there are many settings of interest with multiple agents.

Back-and-Forth Communication.

An important idea in organizational economics is that boundedly rational individuals can achieve superior outcomes when working together (see Simon (1947)). A version of our model with multiple agents speaks to this idea; moreover, it makes sense of why it is important for agents to have back-and-forth communication.

Consider, for instance, a setting where two agents have different codes. They are presented with the same picture P and have the same initial knowledge, consisting of all of the pixels of P. Both agents are purely deductive. Suppose both agents are able to deduce something but not everything about the picture on their own; and because of their different codes, the agents deduce different things. Let F_1^0 and F_2^0 denote the features agents 1 and 2 deduce respectively.

Clearly, the agents benefit from sharing their deductions: agent 1 can add F_2^0 to their knowledge set and vice-versa. However, the benefits do not stop there. Notice that agent

i, having added features from F_j^0 to their knowledge set may be able to make further deductions. Let F_i^1 denote the additional deductions of agent i. The agents can communicate these additional deductions; and perhaps even more deductions will result ($F_i^2, F_i^3, ...$).

We see then that communicating back-and-forth can generate continual revelations. This captures a feature of communication that is intuitive, yet absent from existing models. In models where agents simply have different initial information, they share what they know and there is no further reason to communicate.

Notice that agents do not need to communicate back-and-forth, or even at all, when their codes are the same—since neither agent sees something the other agent fails to notice. If agents' codes are very different, they may benefit from one round of communication, but there is no need for multiple rounds. Agent 1 shares features that are simple for them but complex for agent 2. Thus, agent 2 cannot put agent 1's insights to use to learn more about the picture. Consequently, the realm where back-and-forth communication is most fruitful is where agents' codes are different, but not too different.

Influence Activities.

In a standard economic model of persuasion, a principal influences an agent by controlling the information the agent receives (e.g. Kamenica and Gentzkow (2011)). In our model, what an agent believes depends not only on the information they receive but also on how they interpret that information. This suggests a variety of additional channels through which persuasion can operate.

First, notice that it matters whether a principal intervenes early or late. As we saw in our discussion of path dependence, before an agent has a big-picture view, they are somewhat impressionable. They are open to seeing a "rabbit" or a "duck." Once a big-picture view forms, though, an agent is harder to move.

Consider two ways in which a principal can intervene to influence an agent. One approach is to present information in a particular order. Recall, for example, that it matters whether the agent first focuses on the left-hand side or the right-hand side of the rabbit-duck image. One implication is that a politician with damaging information about an

opponent may want to get it out early. Should voters get to know the opponent—and form a favorable view—they may no longer be receptive to the damaging information.

Another way in which a principal might exert influence is by supplying a narrative. In terms of the model, we might think of this as getting an agent to consider a particular codeword c for inclusion in knowledge. Supplying a particular narrative—such as "rabbit," "rabbit ears" or both—may tilt the agent towards a rabbit interpretation and away from duck.

Notice that simple narratives are more likely to be resonant than complex ones—even when they are wrong. Simple narratives conserve working memory; hence, it is easier for agents to think in terms of them and consider their implications. All politicians know the importance of keeping their message simple. For instance, take Ronald Reagan's famous line: "Government is not the solution to our problem, government is the problem." Similarly, John F. Kennedy's "Ask not what your country can do for you, ask what you can do for your country" is a simple narrative with far-reaching implications.

In 2013, the Italian programmer Alberto Brandolini wrote: "The amount of energy needed to refute bullshit is an order of magnitude bigger than that needed to produce it." This has since come to be known as Brandolini's law.[14] We might interpret Brandolini in terms of the power that simple narratives have. Sometimes the truth is complex (i.e. agents may not have short codewords for the true state of the world). In such instances, simple, false narratives may be hard to counteract.

5.2 Categorization

The psychological literature on categorization, starting with the work of Eleanor Rosch and colleagues, highlights a fascinating phenomenon (see, for example, Rosch (1973)). People normally perceive categories in binary terms: an item either belongs or not. At the same time, some items are considered more exemplary than others. For instance, a robin

[14] A similar adage—often misattributed to Mark Twain—goes "A lie can travel halfway around the world while the truth is putting on its shoes."

is generally perceived as a more exemplary "bird" than an ostrich.

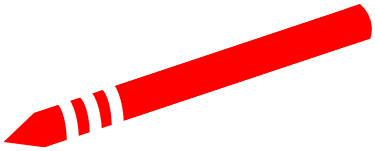
In fact, the most exemplary items—rather than being the most common or average—are often exaggerated forms. Young boys often dress up as soldiers and young girls as princesses precisely because these exaggerated images are particularly exemplary of the categories "male" and "female."

Our model speaks to these findings. Features are binary in the model: a feature f, and associated codeword c, either apply to a picture P or not. At the same time, given some aspects of the picture (say, W_t), codeword c may have higher or lower fit (i.e. $\Phi_P(c, W_t)$ may be higher or lower). If many of the features of a picture P, and its subpictures, have high fit, we might say that P is particularly exemplary of codeword c. Such a picture is one where the agent is particularly likely to extrapolate to codeword c. It may be easier, for instance, for an agent to extrapolate to "bird" from a picture of a robin than a picture of an ostrich.

We might ask what pictures, in this sense, most *scream* "bird." One might guess the picture that most screams "bird" *is* a picture of a bird. However, this is not necessarily the case. In fact, the birdiest picture may be highly exaggerated or abstracted: Picasso's take on a bird rather than a photograph. Note that the exact nature of this abstraction will depend critically upon the agent's code.

A classic experiment by the biologists Niko Tinbergen and Albert Perdeck illustrates the importance of exaggeration and abstraction (see Tinbergen and Perdeck (1951)). To induce a parent to regurgitate food, herring gull chicks peck at a red spot on the parent's

(a) Herring gull.



(b) Red Rod.

Figure 15: Tinbergen and Perdeck's Experiment

bill (see Figure 15, panel a). Tinbergen and Perdeck ran a series of experiments to test the pecking behavior of these chicks. They found that the chicks peck just as much at a model of an adult gull's head, or a model of the beak alone. However, most surprisingly, they found that a red rod with three white stripes (see panel b) induced 25 percent more pecks from the chicks than any of the other alternatives. In other words, the abstracted beak—which wasn't a beak at all—was most exemplary to the chicks.

5.3 Visual Perception

Finally, consider two extensions of the model that capture additional features of visual perception.

Perceiving depth.

People often perceive two-dimensional images as three dimensional. Take Figure 16, for instance, which shows the Necker cube.

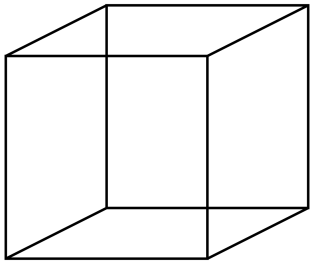


Figure 16: The Necker Cube

It turns out that the model is easily tweaked to account for this phenomenon. Suppose the agent thinks of the two-dimensional image they are presented with as the projection of a three-dimensional image onto a two-dimensional plane (akin to a photograph). We can modify the agent's task so that their goal is to learn about the underlying three-dimensional image from the photograph.

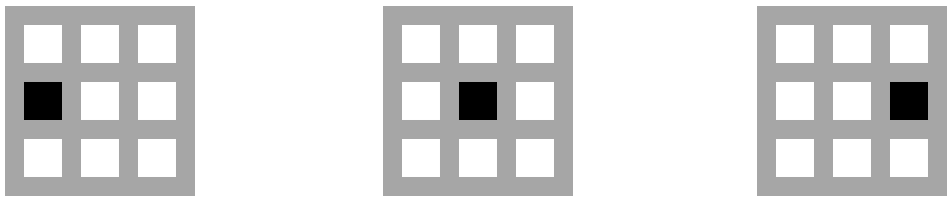
Formally, let $G : \mathbb{R}^3 \to \mathbb{R}^2$ be a projection of a three-dimensional space onto a two-dimensional plane. Let $\hat{P}$ be the true, underlying three-dimensional picture and let $P =$

$G(\hat{P})$ be the corresponding two-dimensional picture that the agent is shown (i.e. the photograph). Observe that for any feature f of P, there is a corresponding feature $\hat{f}$ of $\hat{P}$, where $\hat{f} = \{\hat{P}' : G(\hat{P}') \in f\}$. Suppose the agent's initial knowledge of P is K_0, consisting of the pixels of P. There is a corresponding knowledge set regarding $\hat{P}$, which we denote $\hat{K_0}$. Imagine the agent's task is to start from $\hat{K_0}$ (i.e. codewords on the three-dimensional picture) and develop a better understanding of the three-dimensional picture.

Notice that it is ambiguous whether an image such as Figure 16 is a three-dimensional cube or flat. Consequently, if the agent is purely deductive, they will not reach a firm conclusion. However, the agent might extrapolate to "cube" despite this ambiguity—for instance because they assign "cube" a short codeword. The agent might also look at whether a two-dimensional scene obeys the laws of perspective (e.g. whether there are vanishing points) in deciding whether to deem it three dimensional.

Perceiving motion.

When people view a series of pictures, they may perceive motion. This is the principle on which animation is based. Take Figure 17, for instance. Someone might look at this series of images and perceive a black dot moving from left (in Picture 1) to right (in Picture 3).



Picture 1 Picture 2 Picture 3

Figure 17: Perceiving Motion

Recall that we defined the concept of "common features" in Section 4. We can apply this concept here. Notice that a common feature of Pictures 1, 2, and 3 is "there is a single black pixel." Suppose that, in addition to allowing the agent to learn about common features, we enrich the common feature space, allowing for statements such as "the black

pixel in Picture 1 is the same as the black pixel in Pictures 2 and 3." Of course, it is not identified whether the black pixel is the same or different across pictures—just as it is not identified whether the Necker cube is a three-dimensional cube or flat. Nonetheless, the agent could have a common feature space of this type and use extrapolation to select in cases where there is ambiguity. Observe that if the agent has such a common feature space and views the black dot as the same across pictures, they will view the black dot as *moving* between pictures.

Depth and motion are both examples of how the agent's feature space can be enriched and extrapolation can be used as the tool for selection when there is ambiguity.

6 Conclusion

The Gestalt psychologists' observation that seeing the parts is different from seeing the whole has become a foundational idea in cognitive psychology and neuroscience. Integrating information, and developing an understanding of what it means, is seen as one of the key challenges our brains face. Computer science and information theory are also based on the idea that processing information is far from straightforward. Economics, in contrast to these other disciplines, typically treats the information processing problem as a black box.

This paper builds a framework that helps bridge economics with these other disciplines. We assume that agents have unlimited ability to store information—unlimited hard-drive space, so to speak. However, they have limited working memory (akin to RAM). Working memory is the place where agents integrate information and develop a big-picture understanding of what it means. We show that limited working memory bounds agents' ability to draw conclusions.

We also explore, in this context, the use of extrapolation by agents. On the one hand, extrapolation allows agents to reach further than they can with deduction alone, potentially reaching more correct conclusions. In this sense, the agent avoids Type I error. On

the other hand, extrapolation introduces the possibility of wrong conclusions (Type II error).

This paper leaves a number of unanswered questions. In particular, we do not take a stance on how people make decisions when they have a partial, evolving, and potentially conflicting understandings of the world. There is evidence that people find it difficult to make decisions when they lack an understanding of the situation or feel conflicted (see Shafir et al. (1993)).

In addition, there are aspects of the reasoning process itself that require further exploration. How, for instance, are the contents of the agent's working memory determined? The work of Shleifer and co-authors suggest that there are bottom-up processes at work (i.e. the salience of information) while the literature on rational inattention emphasizes the role of top-down processes (i.e. people make decisions regarding what deserves attention).

As we hinted at in Section 5, multi-agent settings may be a particularly fruitful application of the theory. For instance, the model speaks to Simon's idea that agents' collective rationality may exceed their individual rationality. The model also suggests important levers that can be utilized for persuasion.

7 Proofs

To establish Proposition 2, we will first state and prove two lemmas.

Lemma 1. *Consider a* 100×1 *grid. Let task* $\tau = (F, P)$*, where* F *consists of all* 2^{100} *possible pictures and* P *is any given picture. (In words: the agent's task is to identify picture* P *perfectly.) Then* $L_{\min}(\mathcal{C}^*(\tau), \tau) = 2$*.*

Proof of Lemma 1. Let the pixels of P, from top to bottom, be $\{p_1, p_2, \ldots, p_{100}\}$. For each of $m = 1, \ldots, 100$: (i) denote the $m \times 1$ codeword that corresponds to a single $m \times 1$ subpicture consisting of pixels (from top to bottom) $\{p_1, p_2, \ldots, p_m\}$ by c_m; (ii) denote the 1×1 codeword that corresponds to the single pixel p_m by c'_m; (iii) denote the $m \times 1$ region corresponding to the first m pixels of P from the top as R_m; and (iv) denote the 1×1 region corresponding to the pixel p_m as R'_m.

Choose code $\mathcal{C}$ so that each $m \times 1$ codeword c_m has length 1. (This is always possible because up to two $m \times 1$ codewords can have length 1.)

The following sequence of working memories will successfully identify P. In the initial period $t = 0$, load the individual pixels c_1 for region R_1 and c'_2 for region R'_2 into working memory W_0, so that codeword c_2 for region R_2 is added to the knowledge set. In each subsequent period $t \geq 1$, load the codewords c_t for region R_t and c'_{t+1} for region R'_{t+1} into W_t, so that codeword c_{t+1} for region R_{t+1} is added to the knowledge set. At the end of $t = 98$, the agent will have learned c_{100} for R_{100}, which corresponds to the whole picture P. Notice that in each step, exactly two length-1 codewords were loaded into working memory; so $L = 2$ suffices. □

Lemma 2. *Consider a* 100×1 *grid. Let task* $\tau_S = (F, \hat{\mathcal{P}})$ *where* F *and* $\hat{\mathcal{P}}$ *both consist of all* 2^{100} *possible pictures. (In words: the agent's task is, given any* 100×1 *picture, to identify the picture exactly.) Then* $L_{\min}(\mathcal{C}^*(\tau_S), \tau_S) \geq 3$*.*

Proof of Lemma 2. Assume towards a contradiction that there exists a code $\mathcal{C}$ under which the task τ_S can be achieved with working memory 2. Then for each picture P, there must be some pair of length-1 codewords c and c' and corresponding regions R and R' (implying corresponding features f and $f' \neq f$) that uniquely identify P, in the sense that $f \cap f' = \{P\}$. This implies that there must be at least as many distinct quartets $\{(c, R), (c', R')\}$ (consisting of two codeword-region pairs) as there are distinct pictures, of which there are 2^{100}. For each $m \in \{1, \ldots, 100\}$, there are only two distinct length-1 $m \times 1$ codewords and $101 - m$ distinct $m \times 1$ regions, corresponding to $2(101 - m)$ distinct $m \times 1$ codeword-region pairs; thus there are no more than

$$(2 \sum_{m=1}^{100} (101 - m))^2 = 10100^2$$

distinct quartets, which is fewer than the required 2^{100} pictures. The result holds by contradiction. □

Proof of Proposition 2. Consider a 100×1 grid. Denote the number of possible pictures on this grid as $S = 2^{100}$. Enumerate the S possible 100×1 pictures as $P_1, \ldots, P_S$, in any arbitrary order. Consider a series of tasks $\tau_i = (F, \hat{\mathcal{P}}_i)$, $i = 1, \ldots, S$, where

$$\hat{\mathcal{P}}_i = \{P_1, \ldots, P_i\}.$$

Let $L_i = L_{\min}(\mathcal{C}^*(\tau_i), \tau_i)$ be the minimum working memory required to achieve task τ_i.

Observe that $L_1 = 2$ (by Lemma 1); that L_i is weakly increasing in i; and that $L_S \geq 3$ (by Lemma 2). Let $j = \min\{i : L_i \geq 3\}$, and let $\tau_j' = (F, P_j)$. Assume towards a contradiction that $\mathcal{C}^*(\tau_{j-1}) \cap \mathcal{C}^*(\tau_j')$ contains at least one element $\mathcal{C}$. Notice that, under this assumption, every picture P_i with $i \in \{1, ..., j\}$ can be exactly identified using code $\mathcal{C}$ and working memory constraint $L = 2$; thus it must hold that $\mathcal{C}$ is an optimal code for τ_j with $L_{\min}(\mathcal{C}, \tau_j) = 2$. This contradicts our construction of j; thus $\mathcal{C}^*(\tau_{j-1}) \cap \mathcal{C}^*(\tau_j') = \varnothing$. □

Proof of Proposition 3.

Path dependence. Consider the following setting. (a) $\alpha = 0.01$, and $\beta = 0$. (b) P is a 3×1 picture where the top two pixels p_1 and p_2 are black and the bottom pixel p_3 is white; (c) c_1 is a 3×1 codeword corresponding to the feature "the picture is either black-white-black or white-black-white", and c_2 is a 3×1 codeword corresponding to "the picture is either all black or all white"; (d) c_3 is a 2×1 codeword corresponding to the feature "the pixels in this 2×1 region are not all black", and c_4 is a 2×1 codeword corresponding to the feature "the pixels in this 2×1 region are not all white"; (e) each of c_1, c_2, c_3, c_4 have length one; (f) the agent's working memory capacity is $L = 2$.

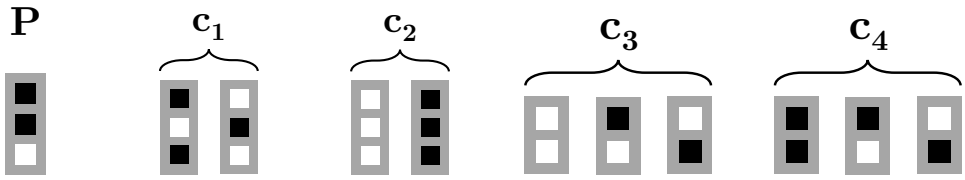


Figure 18: Proof of Path Dependence

Consider an initial working memory where the codeword for the lone white pixel is loaded into W_0, and the codeword c_1 is evaluated. In this case, the fit of codeword c_1 is

$$\Phi_R(c_1, W_0) = \frac{Z_R(W_0, c)}{Z_R(W_0)} = 1/4,$$

which exceeds the inclusion threshold $\alpha = 0.01$; so c_1 is added to the knowledge set.

We claim that in all subsequent extrapolation steps, (a) c_2 is never added to the knowledge set, and (b) c_1 is never removed from the knowledge set. To see why, observe the following. (a) When evaluating c_2, the codeword c_1 has usefulness $2/3$ which exceeds

that of any other length-1 codeword-region pair, and thus is always included in working memory; but then c_2's fit equals zero, and is always rejected. (b) We may confirm case-by-case that every other length-1 codeword c—applied to any region R that is compatible with c— is consistent with c_1, in the sense that the feature implied by c and R has at least one common element with the feature implied by c_1. This means that regardless of which length-1 codeword is loaded into working memory when evaluating c_1, the fit is at least $1/8$ and thus exceeds the threshold α, and thus that c_1 is never subsequently excluded.

Alternatively, consider an initial working memory where the codeword for the lone white pixel is loaded into W_0, and the codeword c_2 is evaluated. In this case, the fit of codeword c_2 is

$$\Phi_R(c_1, W_0) = \frac{Z_R(W_0, c)}{Z_R(W_0)} = 1/4,$$

which exceeds the inclusion threshold $\alpha = 0.01$; so c_2 is added to the knowledge set. We can further retrace our argument from the first case (where c_1 instead of c_2 is initially evaluated), with appropriate adjustments, and conclude that in all subsequent extrapolation steps, (a) c_1 is never added to the knowledge set, and (b) c_2 is never removed from the knowledge set.

Thus, path dependence is possible.

Cycling. Consider the following setting. (a) $\alpha = 1$, and $\beta = 0.99$. (b) P is a 2×1 picture where the top pixel is black and the bottom pixel p_2 is white; (c) c_1 is a 2×1 codeword corresponding to the feature "the bottom pixel is white"; (d) c_1 has length one; (e) the agent's working memory capacity is $L = 2$.

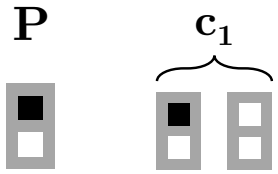


Figure 19: Proof of Cycling

Consider the following sequence of extrapolations.

1. In period 0, the agent evaluates c_1 by loading the bottom (white) pixel into working memory W_0. The fit of c_1 is then

$$\Phi_R(c_1, W_0) = \frac{Z_R(W_0, c_1)}{Z_R(W_0} = 1,$$

   which meets the threshold $\alpha = 1$; so c_1 is successfully added to the knowledge set.

2. In period 1, the agent again evaluates c_1, this time by loading the top (black) pixel into working memory W_0. (Note that the only features in the knowledge set besides c_1 correspond to individual pixels, so the usefulness criterion has no bite in

determining which codeword gets loaded into working memory.) The fit of c_1 is then

$$\Phi_R(c_1, W_1) = \frac{Z_R(W_1, c_1)}{Z_R(W_1} = 0,$$

which fails to meet the threshold $\beta = 0.99$; so c_1 is removed from the knowledge set.

3. In period 2, the agent repeats their extrapolation from period zero: they evaluate c_1 by loading the bottom (white) pixel into working memory W_2. The fit of c_1 is then

$$\Phi_R(c_1, W_2) = \frac{Z_R(W_2, c_1)}{Z_R(W_2)} = 1,$$

which meets the threshold $\alpha = 1$; so c_1 is successfully added to the knowledge set.

It follows that cycling occurred: $c_1 \in K_1(R), c_1 \notin K_2(R), c_1 \in K_3(R)$ where R is the region covering the entire picture.

Patchwork Quilt Consider the following setting. (a) $\alpha = 0.01$, and $\beta = 0$. (b) P is a 2×1 picture where the top pixel is black and the bottom pixel p_2 is white; (c) c_1 is a 2×1 codeword corresponding to the feature "there is exactly one white pixel", c_2 is a 2×1 codeword corresponding to the feature "the picture is either all black or all white", c_3 is a 2×1 codeword corresponding to the feature "at least one pixel is white", and c_4 is a 2×1 codeword corresponding to the feature "at least one pixel is black"; (d) each of c_1 and c_2 have length two, while each of c_3 and c_4 have length one; (e) the agent's working memory capacity is $L = 3$.

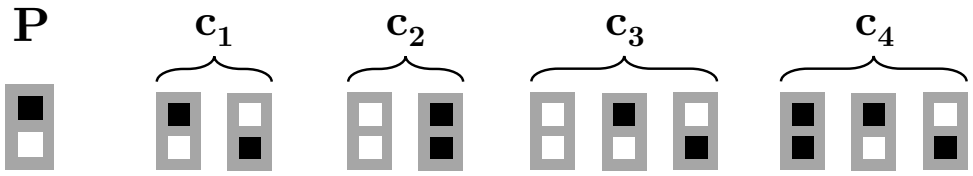


Figure 20: Proof of Patchwork Quilt

Consider the following sequence of extrapolations.

1. In period 0, the agent evaluates c_1 by loading the bottom (white) pixel into working memory W_0. The fit of c_1 is then

$$\Phi_R(c_1, W_0) = \frac{Z_R(W_0, c_1)}{Z_R(W_0)} = 1/2,$$

which meets the threshold $\alpha = 0.01$; so c_1 is successfully added to the knowledge set.

2. In period 1, the agent evaluates c_2, again by loading the bottom (white) pixel into working memory W_1. (Note c_1 cannot be used to evaluate c_2 because they both have length 2, so cannot both fit into working memory given $L = 3$.) The fit of c_2 is then

$$\Phi_R(c_2, W_1) = \frac{Z_R(W_1, c_2)}{Z_R(W_1)} = 1/2,$$

which meets the threshold $\alpha = 0.01$; so c_2 is added to the knowledge set.

At the end of the second period, c_1 and c_2 are both in the knowledge set. We claim that they will both remain forever in the knowledge set. To see why, note that each of the two codewords have length two, and thus (given $L = 3$) can each only be removed by evaluation against a length-one codeword. However, we may check case-by-case that every codeword-region pair (c, R) where c is a length-one codeword is compatible with each of c_1 and c_2, in the sense that they share at least one common element; which means that any evaluation of c_1 or c_2 using a length-one codeword will generate a fit of at least $1/4$, which exceeds the rejection threshold $\beta = 0$. The claim holds, and we conclude that a patchwork quilt persists. □

References

Bénabou, Roland, Armin Falk, and Jean Tirole, "Narratives, imperatives, and moral reasoning," *National Bureau of Economic Research*, 2018.

Bordalo, Pedro, Nicola Gennaioli, and Andrei Shleifer, "Salience theory of choice under risk," *The Quarterly Journal of Economics*, 2012, *127* (3), 1243–1285.

_ , _ , and _ , "Salience and asset prices," *American Economic Review*, 2013, *103* (3), 623–628.

_ , _ , and _ , "Salience and consumer choice," *Journal of Political Economy*, 2013, *121* (5), 803–843.

_ , _ , and _ , "Memory and reference prices: An application to rental choice," in "AEA Papers and Proceedings," Vol. 109 2019, pp. 572–576.

_ , _ , and _ , "Memory, attention, and choice," *The Quarterly Journal of Economics*, 2020, *135* (3), 1399–1442.

Broadbent, Donald Eric, *Perception and communication*, Elsevier, 2013.

Chase, William G and Herbert A Simon, "Perception in chess," *Cognitive Psychology*, 1973, *4* (1), 55–81.

Crawford, Vincent P. and Nagore Iriberri, "Level-K Auctions: Can a Nonequilibrium Model of Strategic Thinking Explain the Winner's Curse and Overbidding in Private-Value Auctions?," *Econometrica*, 2007, *75* (6), 1721–1770.

De Groot, A D, *Thought and choice in chess*, Mouton, 1965.

Deutsch, J Anthony and Diana Deutsch, "Attention: Some theoretical considerations.," *Psychological Review*, 1963, *70* (1), 80.

Eliaz, Kfir and Ran Spiegler, "A model of competing narratives," *American Economic Review*, 2020, *110* (12), 3786–3816.

Gazzaniga, Michael S., Richard B. Ivry, and George R. Mangun, "Cognitive neuroscience: the biology of the mind," 2014.

Kamenica, Emir and Matthew Gentzkow, "Bayesian persuasion," *American Economic Review*, 2011, *101* (6), 2590–2615.

Mullainathan, Sendhil, Joshua Schwartzstein, and Andrei Shleifer, "Coarse thinking and persuasion," *The Quarterly Journal of Economics*, 2008, *123* (2), 577–619.

Rosch, Eleanor H, "Natural categories," *Cognitive Psychology*, 1973, *4* (3), 328–350.

Schwartzstein, Joshua and Adi Sunderam, "Using models to persuade," *American Economic Review*, 2021, *111* (1), 276–323.

Shafir, Eldar, Itamar Simonson, and Amos Tversky, "Reason-based choice," *Cognition*, 1993, *49* (1-2), 11–36.

Shleifer, Andrei, "Remarks on the occassion of Gary Becker's 80th Birthday," *Unpublished panel discussion*, 2010.

Simon, Herbert A, *Administrative behavior*, Simon and Schuster, 1947.

_ , "A behavioral model of rational choice," *The Quarterly Journal of Economics*, 1955, pp. 99–118.

Stahl, Dale O. and Paul W. Wilson, "Experimental evidence on players' models of other players," *Journal of Economic Behavior and Organizations*, 1994, *25* (3), 309–327.

Thaler, Richard H., "From Cashews to Nudges: The Evolution of Behavioral Economics," *Nobel Prize Lecture*, 2017.

Tinbergen, Nikolaas and Albert C Perdeck, "On the stimulus situation releasing the begging response in the newly hatched herring gull chick (Larus argentatus argentatus Pont.)," *Behaviour*, 1951, *3* (1), 1–39.

Tversky, Amos and Daniel Kahneman, "Judgment under Uncertainty: Heuristics and Biases: Biases in judgments reveal some heuristics of thinking under uncertainty.," *Science*, 1974, *185* (4157), 1124–1131.