PERSONALIZATION

The Science Behind Next-Best Action

By Silvio Palumbo, Yun Lim, and Nabeel Siddiqi

ARTICLE JUNE 17, 2026 15 MIN READ

This is Part 3 of a five-part series on next-best action.

The previous article in this series explained how next-best action (NBA) is shifting from marketer-orchestrated journeys to agent-composed actions and discussed the technology architecture needed to enable it. Agent-native NBA requires models capable of exploring pathways that humans never spelled out prescriptively, learning loops that compound without human intervention, and a layered architecture with different classes of algorithms.

This shift requires different applications and a fundamental evolution in the underlying decisioning science. This article, oriented toward technical leaders, describes that scientific

evolution and explores the nuances of advanced decisioning engines.

Three Layers of Action Selection

NBA action selection occurs across three layers: propensity and uplift scoring, contextual bandits, and agent-based reasoning. These layers are additive, not alternative. Systems don't necessarily need to graduate from one to the next, however, as each layer handles a fundamentally different type of intelligence.

Propensity and uplift models estimate the likelihood and incremental value of a customer's responding to a given action. Contextual bandits optimize the exploration-exploitation tradeoff across large sets of structured signals, learning in real time which actions work in which contexts. Foundation model agents interpret unstructured content, compose novel action combinations, and exercise judgment in situations that structured models cannot encode. The most effective systems will run all three layers simultaneously. (See Exhibit 1.)

EXHIBIT 1

Each Layer Expands What the System Can Learn and Act On

	Where most organizations are today		Where the market is moving		The emerging frontier	
	Layer 1	Propensity score and uplift model	Layer 2	Contextual bandits	Layer 3	Agent-based reasoning
Core question	"Who will respond?"		"What action maximizes reward?"		"Which action fits this moment?"	
Key algorithms	XGBoost, LightGBM		Thompson sampling, UCB		Foundation model reasoning	
Exploration	None (exploit only)		Systematic (exploration-exploitation tradeoff)		Reasoning-guided (agent priors and bandit validation)	
Action space	Fixed, marketer-defined		Fixed but dynamically ranked		Composable	
Vendors in market	Pega, Salesforce		Braze, Hightouch, Adobe, Pega		Salesforce, Braze	

Source: BCG analysis.
Note: Vendor examples are illustrative, not exhaustive; inclusion does not constitute endorsement. UCB = upper confidence bound.

Layer 1: Propensity and Uplift Scoring

The dominant paradigm today is to train a model per action (or a multi-output model across actions), score each customer on likelihood to respond, rank each, and select each. The models

are typically gradient-boosted trees (such as XGBoost or LightGBM) or logistic regressions trained on historical response data. The most widely deployed enterprise NBA platforms operate at this layer. Adaptive models use classifiers with online learning, scoring each action by formulas that weight propensity, expected value, and business levers. This approach works well when the action space is small, the customer context is stable, and the goal is single-step conversion. Most organizations should master this layer before adding the others. The returns from handling propensity scoring well are substantial.

Layer 2: Contextual Bandits

Rather than predicting response probability, contextual bandits directly optimize which action to take in light of the customer's context. The key difference is the exploration-exploitation tradeoff: the system deliberately allocates a fraction of decisions to less-certain actions in order to learn their true value, rather than always exploiting the prevalent best guess. This sampling approach has been researched for decades, but only recently have software-as-a-service solutions begun to adopt it widely.

Several marketing platforms have adopted bandit-based architectures in the past two years. One common approach is a "community of bandits" architecture, decomposing the decision into separate bandit dimensions for channel, timing, creative, and offer, each optimizing independently. Others use contextual bandits with gradient-boosted reward models for nonlinear feature interactions, or they implement Thompson sampling to auto-optimize offer selection. These approaches represent a meaningful step beyond propensity scoring because they learn from their own decisions, not just from historical data.

The algorithmic choice within Layer 2 matters. In most NBA contexts, Thompson sampling, which draws actions from the posterior distribution of expected rewards, outperforms simpler approaches such as epsilon-greedy because it naturally adapts its exploration rate to uncertainty: exploring more where it knows less, and exploiting more where it is confident.

For the reward model underlying the bandit, linear contextual bandits offer interpretability and computational efficiency but assume linear reward functions. Neural contextual bandits capture nonlinear interactions between customer features and actions at the cost of computational complexity. In practice, a pragmatic approach involves using neural reward models in the batch layer (where computational budgets are larger) and distilling them into lighter linear or tree-based models for real-time serving.

Consider a credit card issuer that wants to test balance-transfer offers. A propensity model, trained on historical responses, learns that customers with high balances and long tenure respond best. A contextual bandit discovers something that the historical data never revealed: when paired with a specific creative treatment, recently activated cardholders with moderate balances respond at even higher rates because they are actively evaluating offers. The propensity model never explored this combination; the bandit did because it was designed to.

Layer 3: Agent-Based Reasoning

Foundation models add value to the decisioning stack in an area where structured models hit a ceiling: dealing with unstructured or highly disjointed data. A contextual bandit excels at optimizing across large sets of structured signals (such as account tenure, transaction frequency, channel preference, and response history), but it can't interpret the tone of a customer's last service chat, assess whether a browsing sequence suggests confusion or comparison shopping, or weigh the strategic intent behind a brand campaign. These are judgment calls, not optimization problems.

A foundation model agent reasons in natural language about the customer's situation, the available options, and the likely outcomes. It processes context that would be prohibitively expensive to encode as features for a structured model. Organizations should consider adding Layer 3 when they see evidence that their models' decisioning quality is constrained by unstructured context that bandits cannot ingest. Layer 3 is not a replacement for Layers 1 and 2, but a complementary capability. The first platform bets on Layer 3 are emerging, but no vendor has deployed the full vision at scale yet. This represents both a risk and an opportunity.

The three layers compose a stack, not a sequence. Propensity and uplift models provide foundational scores. Contextual bandits use those scores as inputs while adding exploration and real-time learning across structured signals. Foundation model agents sit on top, interpreting unstructured context that neither of the lower layers can process and making compositional decisions that optimization alone can't reach. The agent doesn't replace the bandit; it decides when to defer to the bandit's quantitative recommendation and when to override it because unstructured context has changed the calculus. When an organization runs all three layers simultaneously, scoring, optimization, and reasoning work in concert.

Limitations of Propensity and Uplift Models

Most organizations operate exclusively at Layer 1, and well-executed propensity and uplift scoring already delivers meaningful lift. It is interpretable, auditable, and well understood. But even the best combinations of propensity and uplift models have three structural limitations as organizations scale their personalization ambitions:

- **Propensity models (without uplift models) do not provide prescriptive recommendations.** A propensity model answers "How likely is this customer to respond?" but not "Will this

action change behavior relative to doing nothing?" A customer with a 90% propensity to purchase may have planned to buy anyway; if so, targeting that customer with a discount destroys margin without changing behavior. The critical upgrade within Layer 1 is the shift to uplift modeling, which estimates the incremental effect of an action versus no action. Organizations that make this shift typically discover that 20% to 40% of their active programs deliver hardly any incremental lift, instead just capturing intent that would have materialized anyway. This requires different training data (randomized holdout groups or quasi-experimental designs) and different model architectures. The most effective approaches include T-learners (separate models for treatment and control groups), S-learners (a single model with the treatment as an input), and doubly robust learners, which combine outcome modeling with propensity weighting to reduce bias from either component alone.

- **They cannot discover novel combinations.** Propensity and uplift models score within a predefined action space. If the marketer has defined 15 offers, a model of this type will rank just those 15. It can't discover, for example, that combining a smaller offer with a specific content frame and a particular channel at a nonstandard time would outperform any of the 15 predefined options. As the composable shelf grows to hundreds of atomic assets that can be linked in thousands of possible combinations, this limitation becomes a binding constraint.
- **They struggle with sequential decisions.** These models score each action independently. A propensity model doesn't account for the fact that sending an offer today changes the optimal choice for tomorrow. Instead, it treats each decision in isolation, although actual customer engagement is a multistep setup in which current actions shape future options. More complex uplift modeling implementations can calculate conditional probabilities that are contingent on a prior step, but the formulation requires constant oversight and learning cycles.

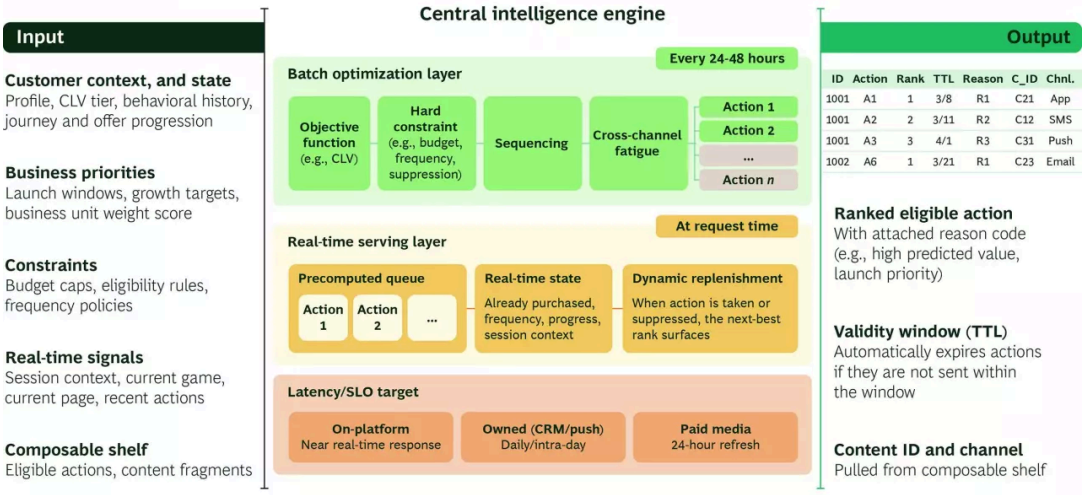
These limitations are not bugs. They are inherent to the paradigm. Moving past them requires both a different system architecture and an additional (not alternative) class of decisioning models.

The Two-Track Architecture

The production system that makes Layer 2 and Layer 3 models operational requires a two-track architecture that separates batch optimization from real-time serving. Each track has different latency requirements, different model types, and different roles in the overall system. (See Exhibit 2.)

EXHIBIT 2

Batch Optimization and Real-Time Serving Operate on Different Cadences with Different Roles



Source: BCG analysis.
Note: CLV = customer lifetime value; SLO = service-level objective; TTL = time to live.

Batch Optimization Layer

The batch optimization layer runs on a 24- to 48-hour cadence. Its job is to compute policies in advance, update model parameters, and solve portfolio-level optimization problems that are too computationally expensive for real-time execution. This is where the system answers strategic questions such as “Given the current shelf composition, customer base distribution, and business constraints (budget ceilings, contact frequency limits, and channel capacity), what is the optimal allocation of actions across the population?”

In practice, the batch layer runs several processes. For example, it retrains bandit policies on the latest interaction data, incorporating new reward signals. Then it refreshes propensity and uplift models with expanded training windows. And finally, it uses portfolio optimization to solve a constrained allocation problem: how to maximize expected total value subject to inventory constraints on offers, frequency caps per customer, and budget limits per business unit. This last step is crucial and often overlooked. Without portfolio-level optimization, the real-time layer will assign the best action to each individual customer, quickly exhausting high-value offers on easy-to-convert customers while starving harder-to-reach segments.

Algorithmically, the portfolio optimization is typically formulated as a mixed-integer linear program (MILP) for tractable problem sizes, using Lagrangian relaxation to decompose the problem when the customer-action matrix exceeds solver limits. In practice, most organizations need both: MILP to make precise allocations within business units or segments, and Lagrangian relaxation to coordinate across the full portfolio. Relying on either by itself introduces bias. MILP

alone cannot scale to millions of customers, and Lagrangian relaxation alone may produce suboptimal allocations within smaller subproblems for which exact solutions are feasible.

Real-Time Serving Layer

The real-time serving layer operates at the moment of customer interaction, with latency typically under 100 milliseconds. When a customer opens the company's banking app, visits its website, or triggers an event, the serving layer must immediately decide what to display. To do so, it takes the policies and parameters that the batch layer precomputed and adapts them to the live context: what the customer is doing right now, what channel the customer is in, what time it is, and what has happened since the batch layer last ran.

The real-time layer is where contextual adaptation happens. A customer whose batch-computed policy says "show a balance transfer offer" might instead receive a service recovery message if the real-time layer detects that the individual has just completed a frustrating support interaction. The batch layer sets the strategic frame; the real-time layer exercises tactical judgment within it.

Foundation model agents operate primarily in the real-time layer, handling the compositional reasoning that determines exactly how to assemble a particular action from shelf components. Policies and budgets computed in the batch layer serve as guardrails. This separation is essential because it prevents the agent from making locally optimal decisions that are globally suboptimal (such as burning the entire monthly budget on high-value offers in the first week).

The interaction between the two layers constitutes the engineering backbone of agent-native NBA. Organizations that try to build purely real-time systems without batch optimization will discover that per-customer decisions do not translate into coherent portfolio-level strategies. But organizations that rely solely on batch processing will miss the contextual moments that drive the highest incremental value.

Objective Function for Reinforcement Learning and the Bandit Layer

The batch processing layer is where reinforcement learning (RL) operates. Its core function is to manage the exploration-exploitation tradeoff. Should the system exploit the best action given current knowledge, or should it explore a less-certain action to gather information that may improve future decisions? This nontrivial tradeoff is the mechanism that separates systems that learn from their own decisions from systems that just replay patterns found in historical data.

The most practical deployment of RL for NBA is the contextual bandit, in which the system observes a customer's context, selects an action, and soon receives a reward signal such as a click, a conversion, or an app open. The policy model at the center of this process operates in a continuous loop. First, it takes customer context as input and outputs an action selection. Then it uses the observed reward to update its beliefs about which actions work in which contexts. Over thousands of iterations, the policy converges on increasingly effective action assignments.

Not all business objectives produce immediate reward signals. In attempts to optimize for customer lifetime value (CLV), the true outcome takes months or years to observe. Multistep RL can handle this, but it is substantially more complex to implement and requires significantly more exploration data to perform well. In most NBA situations, a contextual bandit with a well-chosen proxy reward is the pragmatic choice. It is critical that the proxy links causally to the ultimate business objective, because optimizing for the wrong reward will produce a policy that seems to perform well on the proxy metric but fails to move the outcome that actually matters. Getting this right is one of the most consequential design decisions in the system.

The algorithmic architecture of the policy model itself presents a meaningful design choice. Two families dominate: value-based methods and policy gradient methods. Value-based methods estimate the expected reward for each possible action in a given context and then select the highest-value option. They work well when the action space is relatively constrained (fewer than 1,000 discrete actions) and when the organization needs a reliable policy quickly. They also offer more direct control, enabling implementation of business-specific constraints and nuances in the policy logic. For most NBA deployments today, value-based methods are the right starting point.

In contrast, policy gradient methods learn a probability distribution over actions and then update that distribution based on observed rewards. They are the better choice when actions are continuous (such as determining the discount percentage to offer a customer) or when the action space is very large, because they can represent the policy compactly through distributional parameters rather than enumerating every possible action. The tradeoff is sample efficiency. Policy gradient methods require more exploration data to converge, and this translates into a longer learning period before the policy can reach peak performance.

The method chosen will determine the system's learning speed, its ability to handle business constraints, and the speed at which it can evaluate new actions on the composable shelf. Organizations that are building their first bandit layer should default to value-based methods for their core action selection and reserve policy gradient approaches for the specific dimensions—such as pricing, discount depth, and contact timing—where continuous optimization delivers the highest marginal value.

Foundation Models as the Reasoning Layer

Foundation models introduce a qualitatively different capability into the NBA stack: the ability to reason over unstructured context and to compose actions that structured optimization cannot reach. A common first instinct is to use them for content generation, which is valuable but incremental.

The deeper value lies in three capabilities that restructure the decisioning process itself:

- **Contextual Understanding Beyond Feature Vectors.** Traditional models operate on structured features, such as account balance, days since last purchase, and segment membership. Foundation models can also process unstructured context, such as the text of a customer's last service chat, the sequence of pages browsed, and the tone of a subsequent social media post. This is not tantamount to adding more features to an existing model. It represents a qualitatively different kind of understanding. When an agent reads that a customer's service interaction ultimately reached resolution, but the customer expressed frustration about the process, it can adjust the next action in a way that no feature vector could encode.
- **Compositional Reasoning.** Foundation models can reason about combinations at a scale that optimization alone can't match. For a shelf of 200 atomic assets across offers, creative treatments, content blocks, and channels, the number of possible compositions is astronomical. Consequently, a brute-force optimization approach collapses under the dimensionality. A foundation model instead reasons about plausible compositions: "This customer is in a trust-recovery moment, so pair a security-focused message with a value-reinforcement offer, skip promotional language, and use an intimate channel such as in-app rather than email."
- **Explainable Decision Rationale.** Unlike black-box models that produce a score, foundation model agents can articulate why they made a decision. In regulated industries such as banking, insurance, or health care, explainability is a compliance requirement. And for the technical leaders governing these systems, understanding why the system chose a particular action is essential for debugging, calibrating trust, and adjusting constraints. The agent's reasoning trace becomes an audit artifact that structured models can't provide.

In the practical architecture, foundation models occupy the top layer in the decisioning stack. Data and features flow up from the customer data platform, structured models produce scores and policy recommendations, and the foundation model agent reasons over these inputs, together with unstructured context, to make the final compositional decision. The agent's output

is not a score but a fully composed action with a specific combination of offer, creative, channel, timing, and message assembled from the shelf.

The Cold-Start Problem and Intelligent Exploration

Every decisioning system faces a version of the cold-start problem. When a new action, a new customer segment, or a new context emerges, the system has no historical data to inform its decisions. In systems that rely solely on Layer 2, human operatives manage cold starts manually. The marketer designs a test plan, allocates budget, runs the experiment, and updates the models. With Layer 3, the system can handle cold starts autonomously.

This is where the interaction between foundation models and contextual bandits becomes critical. The foundation model provides a reasoned "prior"—an estimate of how a new action might perform, given its characteristics, the characteristics of similar actions, and the context in which it would be deployed. The bandit uses this prior as a starting point and updates it through controlled exploration. Instead of engaging in purely random exploration (which is wasteful) or purely model-based exploitation (which is biased toward the known), the system explores intelligently, guided by the agent's reasoning about where the highest-value information lies.

In practical terms, this means that when the composable shelf adds a new offer construct, the system does not need weeks of manual testing to understand its value. The agent reasons about which customer contexts are most likely to benefit, the bandit allocates a small fraction of traffic to test those hypotheses, and the system converges on a performance estimate within days rather than quarters. The speed of learning delivers a competitive advantage.

Organizations that can onboard and optimize new actions faster can respond to market shifts more quickly and maintain a constantly refreshed shelf.

The Path Forward

Every technique described in this article is widely available today. Contextual bandits are deployed at scale in recommendation systems across technology companies and, increasingly, in enterprise marketing platforms. Foundation models are commercially available and rapidly

improving. The scientific building blocks for agent-native NBA are not theoretical. They are deployable.

The problem is organizational. Most marketing analytics teams are not staffed for reinforcement learning or bandit optimization. And most organizations have not yet built the two-track architecture necessary to make real-time agent-based decisioning possible. Technical leaders who start building this infrastructure now—hiring RL and causal inference expertise, standing up the batch and real-time architecture, and piloting contextual bandits—will position the organization to lead rather than follow the transition to agent-native NBA.

This is Part 3 of a five-part series on the future of next-best action. Part 1 explores why most NBA programs underdeliver and identifies four structural gaps. Part 2 discusses the shift from marketer-orchestrated journeys to agent-composed actions and the technology architecture that enables it. Part 4 examines how marketing organizations can restructure around the new operating model. Part 5 addresses measurement: from campaign-level attribution to agentic measurement.

Authors

Silvio Palumbo

Managing Director & Senior Partner
New York

Yun Lim

Associate Director
Chicago

Nabeel Siddiqi

SCT Outside Consultant
BCG X – Manhattan Beach

ABOUT BOSTON CONSULTING GROUP

Boston Consulting Group bridges the gap between ambition and outcomes for the world's leading companies and organizations. We are built for this era of unprecedented change — bringing strategic clarity rooted in over 60 years of deep domain knowledge, combined with applied AI shaped by our practitioners. BCG works shoulder-to-shoulder with CEOs across industries and geographies to deliver transformative impact at scale: stronger returns, transferred capabilities, and change that sticks. For more information, visit bcg.com.