Mathematical Framework for Skill-Based Number Betting and Payout Liabilities

Darshan P.
Independent Researcher, Computer Programming Specialist
drshnp@bayesiancorp.com

April 16, 2025

Abstract

We develop a mathematical model for a number betting mechanism where the winning outcome is determined by minimizing the aggregate wager on a digit. The overall betting pool is redistributed among winners according to a contribution function that raises the wager to a power and applies a logistic time-decay weight.

1 Introduction

In traditional betting systems the outcome is random, yet available market information can be exploited to drive outcomes by means of strategic decisions. Here, we propose a system in which participants wager on a digit (0 to 9) and the winning digit is chosen as the one with the minimal aggregate wager. Payouts are redistributed based on a contribution function that incorporates both the magnitude and timing of each wager. This approach draws on existing work in prediction markets [1, 2, 3] and techniques used to extract risk-neutral densities from option prices [4], with potential applications in decision support [5].

2 Mathematical formulation

2.1 Bet Structure

Let each participant choose a digit $n \in \{0, 1, \ldots, 9\}$. Each bet i is characterized by:

- A wager $A_i > 0$,
- A timestamp t_i with $t_i \in [0, T]$, where T is the betting round duration.

For a given digit n, denote by B_n the set of bets placed on n.

2.2 Winning digits selection

The total wager on digit n is defined as:

$$S(n) = \sum_{i \in B_n} A_i.$$

The winning digit n^* is then:

$$n^* = \arg \min_{n \in \{0,\ldots,9\}} S(n).$$

This criterion selects the digit with the least total investment, effectively incentivizing strategic avoidance of popular choices.

2.3 Contribution function

For a bet i on the winning digit n^*, the contribution is defined by:

$$C_i = A_i^{\gamma} \, W(t_i),$$

where:

- $\gamma > 1$ is a parameter that magnifies differences in wager sizes,
- $W(t_i)$ is a logistic time-decay function:

$$W(t_i) = \frac{1}{1 + e^{\alpha\left(\frac{t_i}{T} - 0.5\right)}},$$

  with $\alpha > 0$ determining the sensitivity to timing.

2.4 Payout allocation

The total betting pool is:

$$\text{Pool} = \sum_{j \in \{0,\ldots,9\}} \sum_{i \in B_j} A_i.$$

For bets on the winning digit, the payout P_i is allocated proportionally:

$$P_i = \frac{C_i}{\sum_{j \in W} C_j} \times \text{Pool},$$

where W is the set of bets on n^*.

2.5 Return computation

The net profit for a winning bet is:

$$\text{Net Profit}_i = P_i - A_i,$$

and the return percentage is given by:

$$\text{Return}\%_i = \left(\frac{\text{Net Profit}_i}{A_i} \right) \times 100.$$

2.6 Python random implementation

```python
import random
import math
import pandas as pd

# Simulation Parameters
num_bets = 700
num_numbers = 10
round_duration = 10000  # T
gamma = 1.3             # Amount scaling
alpha = 5               # Time weight steepness for logistic
num_users = 200

# Step 1: Generate Random Bets
bets = []
for i in range(num_bets):
    bet = {
        "bet_id": i,
        "user": f"User_{random.randint(0, num_users - 1)}",
        "bet_number": random.randint(0, num_numbers - 1),
        "amount": random.uniform(5, 100),
        "time": random.uniform(0, round_duration)
    }
    bets.append(bet)

df_bets = pd.DataFrame(bets)
print("All Bets in This Round:")
print(df_bets)

# Step 2: Determine Winning Number
grouped = df_bets.groupby("bet_number").agg(
    total_amount=("amount", "sum"),
    count=("amount", "count")
).reset_index()

print("\nTotal Bets by Number (sorted by total amount):")
print(grouped.sort_values("total_amount"))

winning_number_row = grouped.loc[grouped["total_amount"].idxmin()]
winning_number = winning_number_row["bet_number"]
print("\nWinning Bet Number (Least Total Bet Amount):", winning_number)

# Step 3: Compute Contributions
winning_bets = [bet for bet in bets if bet["bet_number"] == winning_number]

for bet in winning_bets:
    t = bet["time"]
    normalized_t = t / round_duration
    time_weight = 1 / (1 + math.exp(alpha * (normalized_t - 0.5)))
    contribution = (bet["amount"] ** gamma) * time_weight
    bet["normalized_time"] = normalized_t
    bet["time_weight"] = time_weight
```

```
    bet["contribution"] = contribution

total_contribution = sum(bet["contribution"] for bet in winning_bets)
pool = sum(bet["amount"] for bet in bets)

for bet in winning_bets:
    bet["payout_share"] = (bet["contribution"] / total_contribution) * pool if
        total_contribution > 0 else 0

df_winning = pd.DataFrame(winning_bets)
print("\nWinning Bets Details (with Logistic Time Weight):")
print(df_winning[[
    "bet_id", "user", "bet_number", "amount", "time", "normalized_time",
    "time_weight", "contribution", "payout_share"
]])

# Step 4: Aggregate by User
user_summary = df_winning.groupby("user").agg(
    total_bets=("bet_id", "count"),
    total_amount_bet=("amount", "sum"),
    total_contribution=("contribution", "sum"),
    total_payout=("payout_share", "sum")
).reset_index()

user_summary["net_profit"] = user_summary["total_payout"] - user_summary["
    total_amount_bet"]
user_summary["return_percentage"] = user_summary["net_profit"] / user_summary["
    total_amount_bet"] * 100

print("\nTotal Payout Summary by User (with Net Profit):")
print(user_summary)
```

Listing 1: Python Random Results Payouts

3 Simulation and parameter analysis

To validate the model, we simulate multiple betting rounds:

1. Generate bets for digits $n \in \{0, \ldots, 9\}$ with A_i drawn from a specified distribution and timestamps t_i uniformly sampled from $[0, T]$.
2. Compute $S(n)$ and determine the winning digit n^*.
3. For bets on n^*, calculate contributions C_i and determine individual payouts P_i.

4 From randomized script to structured simulation

The Python code presented above is not intended as a full simulation of real-world betting behavior. It generates random wagers and computes payouts solely to illustrate how the contribution and payout functions operate. It does not model strategic decision–making based on

real-time information, nor does it account for fees, taxes, or coordinated actions. To move from a "random-reward" script toward a more realistic simulation, we suggest the following extensions:

1. **Early, uninformed bettors.** Introduce a group of agents who place initial bets at random, reflecting users without access to aggregate wager data. These agents benefit from higher time-decay weights but make uninformed choices.

2. **Strategic counter-bettors.** A second group of agents, implemented using multiprocessing, observe partial wager data and try to identify less crowded digits. Some may place decoy bets to influence crowding perception. Returns should account for platform fees and taxes.

3. **Last-minute risk-takers.** Model a third group that waits until late in the round to place larger wagers, exploiting complete information at the cost of reduced time-decay weight. Their strategy trades higher potential payout against greater risk of capital loss.

4. **Multi-account and fear-driven behavior.** Allow agents to place multiple bets (simulating multi-account users) and to react to losses with increased urgency or fear, thereby capturing regret aversion and herding under stress.

These enhancements would better reflect the layered strategic interactions and behavioral biases present in an operational system.

Preliminary results indicate:

- Elevated values of γ disproportionately emphasize larger wagers, which can skew outcome distributions when participants act independently.

- The logistic time-decay function $W(t_i)$ moderates timing effects, rewarding early wagers and reducing incentives for last-minute manipulation.

- The strategic avoidance behavior mirrors recursive reasoning in Keynesian beauty contests: participants attempt to predict which digit will be least chosen by others [6].

5 Discussion

The proposed method deterministically selects outcomes based on aggregate betting behavior, bridging analysis and skill over random guesses. This approach aligns with established prediction market techniques [1, 2, 3] and with methods for extracting risk-neutral densities from option prices [4].

Keynesian beauty contest logic is evident: bettors base their choices on expectations of others' bets rather than on intrinsic digit value [6]. The logistic time-decay function introduces an additional strategic dimension by weighting early bets more heavily, which shapes participants' anticipatory reasoning.

In practice, participant decisions often reflect behavioral patterns that deviate from purely strategic reasoning. For example, loss aversion may lead users to avoid placing large bets, even when the contribution function ($\gamma > 1$) favors higher stakes. Anchoring effects may occur when early bets on certain digits influence subsequent choices, regardless of their actual strategic

value. Additionally, herding behavior can emerge when players imitate perceived trends, even in a setup that rewards choosing less popular options. These behaviors align with bounded rationality models from behavioral economics [7].

6 Conclusion

We have presented a mathematical framework for a number betting system where outcomes are driven by the collective strategic behavior of participants. Through formal definitions, simulation, and parameter analysis, we show how the system shifts the paradigm from random chance to analysis-based performance. Future work will include empirical validation and refinement of parameters for deployment in real-world settings.

References

[1] Wolfers, J., & Zitzewitz, E. (2004). Prediction Markets. *Journal of Economic Perspectives*, 18(2), 107–126.

[2] Berg, J., Nelson, F., & Rietz, T. A. (2008). Prediction Market Accuracy in the Long Run. *International Journal of Forecasting*, 24(3), 465–480.

[3] Berg, J., Forsythe, R., Nelson, F., & Rietz, T. (2007). Chapter 80: Results from a Dozen Years of Election Futures Markets Research. *ScienceDirect*. Available at: `https://www.sciencedirect.com/science/article/abs/pii/S1574072207000807`.

[4] Aït-Sahalia, Y., Wang, Y., & Yared, F. (2000). Do Option Markets Correctly Price the Probabilities of Movement of the Underlying Asset? *ScienceDirect*. Available at: `https://www.sciencedirect.com/science/article/abs/pii/S0304407600000919`.

[5] Prediction Markets as Decision Support Systems. (n.d.). *Springer*. Available at: `https://link.springer.com/article/10.1023/A:1022002107255`.

[6] Keynes, J. M. (1936). *The General Theory of Employment, Interest and Money*. Harcourt, Brace.

[7] Kahneman, D., & Tversky, A. (1979). Prospect Theory: An Analysis of Decision under Risk. *Econometrica*, 47(2), 263–292.