Cumulative Prospect Theory: A Simulative Demonstration

Joseph Mastromonica

March 2024

Contents

1 A Historical and Mathematical Overview: Cumulative Prospect Theory

Prospect Theory was first introduced by Kahneman and Tversky in 1979 as a descriptive model of decision making under uncertainty. The theory successfully explains behavioral phenomena that violate expected utility theory, including the famous Allais paradox and St. Petersburg paradox. While Prospect Theory achieved its goals, it sometimes violated stochastic dominance and transitivity assumptions. These limitations led to the development of Rank-Dependent Utility (RDU) by Quiggin in 1982, which later influenced Kahneman and Tversky's 1992 revision known as Cumulative Prospect Theory (CPT).

CPT rests on two fundamental observations about human decision-making:

1. **Reference Dependence**: People evaluate outcomes relative to a reference point rather than considering absolute final states. This leads to asymmetric treatment of gains and losses, with losses typically looming larger than equivalent gains - a phenomenon known as *loss aversion.*

2. **Probability Weighting**: Individuals tend to overweight extreme but unlikely events while underweighting moderate or certain outcomes. This non-linear probability weighting helps explain why people both buy lottery tickets and insurance policies.

2 Mathematical Formulation

We begin by considering a gamble represented as:

$$\bar{x} = (x_{-m}, p_{-m}, \ldots; x_{-1}, p_{-1}, x_0, p_0, x_1, p_1, \ldots; x_n, p_n)$$

where $x_i < x_j$ for $i < j$, $x_0 = 0$ serves as the reference point, and $\sum_{i=-m}^{n} p_i = 1$. Under CPT, an agent assigns this lottery the value:

$$\sum_{i=-m}^{n} \pi_i v(x_i) \tag{1}$$

where π_i represents the decision weight for outcome x_i, and $v(\cdot)$ is the value function. The decision weights are calculated differently for gains and losses:

$$\pi_i = \begin{cases} w^+(p_i + \cdots + p_n) - w^+(p_{i+1} + \cdots + p_n), & \text{for } 0 \leq i \leq n \\ w^-(p_{-m} + \cdots + p_i) - w^-(p_{-m} + \cdots + p_{i-1}), & \text{for } -m \leq i < 0 \end{cases}$$

The value function has a piecewise definition:

$$v(x) = \begin{cases} x^\alpha, & \text{for } x \geq 0 \\ -\lambda(-x)^\alpha, & \text{for } x < 0 \end{cases}$$

with probability weighting functions:

$$w^+(p) = \frac{p^\gamma}{[p^\gamma + (1-p)^\gamma]^{1/\gamma}}, \quad w^-(p) = \frac{p^\delta}{[p^\delta + (1-p)^\delta]^{1/\delta}}$$

The parameters are constrained as $\alpha, \gamma, \delta \in (0, 1)$ and $\lambda > 1$, capturing:

- α: Diminishing sensitivity to gains/losses
- λ: Degree of loss aversion
- γ, δ: Probability weighting for gains and losses respectively

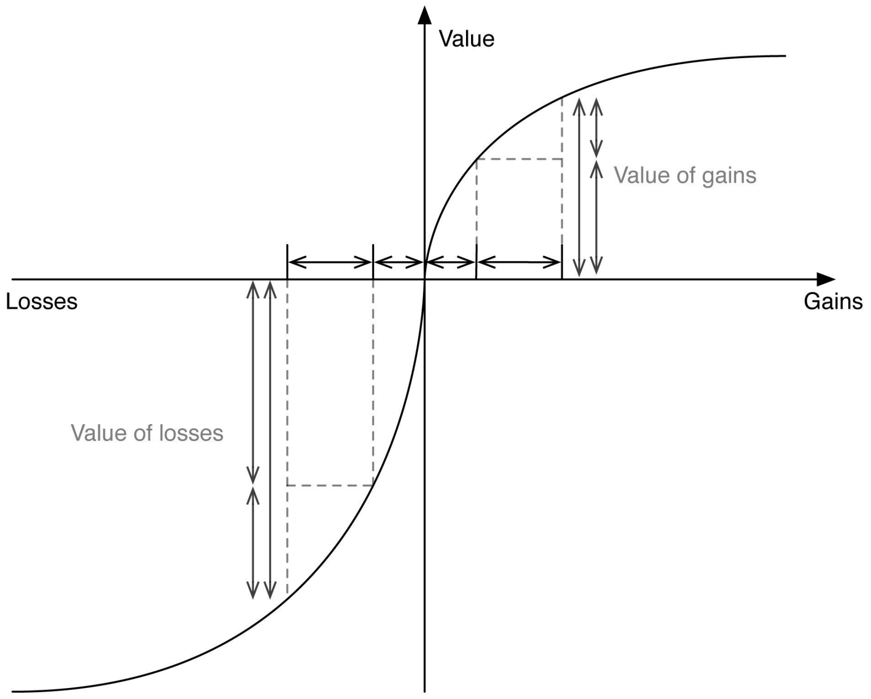

Figure 1: The CPT value function showing loss aversion (kink at reference point), concavity for gains, and convexity for losses

3 Comparison with Expected Utility Theory

Those familiar with Expected Utility Theory's (EUT) may already see parallels between the expected utility function, (2), and the value function, (1). The expected utility function described in EUT is

$$\sum_{t=1}^{n} p_t U(w_0 + x_t) \tag{2}$$

where w_0 represents initial wealth and $U(\cdot)$ is a concave utility function. There are a couple of key differences between CPT (1) and EUT (2):

- **Evaluation Framework**: CPT evaluates changes from a reference point, while EUT evaluates final wealth states
- **Value vs Utility**: $U(\cdot)$ in (2) is differentiable everywhere while the analogous value function $v(\cdot)$ in (1), is non-differentiable at origin to indicate loss aversion (more sensitive to losses than to equivalent gains). The parameter λ is typically used to determine the severity of loss aversion.
- **Probability Weighting**: CPT transforms probabilities non-linearly, while EUT uses objective probabilities. This aspect of our equation ensures that our agent over exaggerates tail-end probabilities; that is, he or she is especially averse to extreme outcomes.
- **Shape**: $U(\cdot)$ is concave throughout its domain, while $v(\cdot)$ is concave over positive x-values (gains) but convex over negative x-values (losses). This concavity is determined by our α parameter.

4 Applications of CPT

4.1 Healthcare Decision Making

According to “Neuroeconomics, health psychology, and the interdisciplinary study of preventative health behavior” by DeStasio, Clithero and Berkman (2019), the notion that decision making is a common denominator in preventable disease is “acutely applicable”. Furthermore, the paper discusses some of the main medical contributions of CPT. Firstly, patients will react differently to circumstances phrased as a string of potential gains v.s. potential losses, based on their reference point. Secondly, this reference point may impact patients’ preventive health behaviors that are “unpleasant themselves (e.g., vaccinations or invasive screening tests), where the risk of the immediate negative outcome (e.g., pain) is felt higher than the risk of the potential long-term outcome” (Gisbert-Pérez, Martí-Vilar, and González-Sala, 2022). These phenomena are qualitative indicators that framing a decision in a particular way around either gains or losses- can influence preventive and reactionary health behaviors.

4.2 Financial Markets

On the financial side, the article “Prospect Theory and Stock Returns: An Empirical Test” by Barberis, Mukherjee, and Wang (2016) introduces a mathematical model using CPT that supposedly captures an investor’s feelings towards a stock, assuming that he or she was able to gather historical data based on that stock. Given the historical return of a stock over the past 60 months, suppose that m of these returns are negative, while $n = 60 - m$ of these returns are positive. We can then formulate a gamble in which r_{-m} is the most negative return and is the r_{-n} most positive over the past 60 months as follows:

$$\bar{x} = (r_{-m}, \tfrac{1}{60}; \ldots; r_{-1}, \tfrac{1}{60}, r_0, \tfrac{1}{60}, r_1, \tfrac{1}{60}, \ldots; r_n, \tfrac{1}{60})$$

where r_i are monthly stock returns. The CPT valuation:

$$\sum_{i=-m}^{n} \pi_i v(r_i) = \sum_{i=-m}^{-1} v(r_i) \left[w^{-}\left(\tfrac{i+m+1}{60}\right) - w^{-}\left(\tfrac{i+m}{60}\right)\right] + \sum_{i=1}^{n} v(r_i) \left[w^{+}\left(\tfrac{n-i+1}{60}\right) - w^{+}\left(\tfrac{n-i}{60}\right)\right]$$

An important property of this model is that it is time insensitive; that is, the order of returns and in which month they occurred is of no importance to the model. In fact, this model only adequately describes an investor’s attitude towards a stock as a whole, with equal probabilities for each return and no probability weighting based on the particular time of a return. There exists modifications of this model that time-weight the returns.

5 Simulation Implementation

Cumulative Prospect Theory (CPT) suggests that behavior under uncertainty can be quantified and thus predicted. I intend to demonstrate this behavior with a simulation. Using the Python language, I will utilize object oriented programming to construct value and probability models according to Kahneman and Tversky.

5.1 Class Structure

The Python implementation uses object-oriented programming to model CPT agents:

```python
class cpt:
    def __init__(self, alpha, gamma, sigma, lam):
        self.alpha = alpha  # Utility curvature
        self.gamma = gamma  # Prob weighting (gains)
        self.sigma = sigma  # Prob weighting (losses)
        self.lam = lam      # Loss aversion

        # Parameter validation
        if lam <= 1:
            raise ValueError("Lambda must be > 1 for loss aversion")
        if not (0 <= alpha <= 1) or not (0 <= gamma <= 1) or not (0 <= sigma <= 1):
            raise ValueError("alpha, gamma, sigma must be in (0,1)")
```

Note that our parameters adhere to the aforementioned restrictions in section 2.

5.2 Core Functions

5.2.1 The Value Funciton

Recall that an agent assigns the value x^α for $x \geq 0$ (nonnegative outcomes), and $-\lambda(-x)^\alpha$ for $x < 1$ (negative outcomes):

```python
def value_function(self, x):
    """Calculate prospect theory value"""
    if x >= 0:
        return x ** self.alpha  # Concave for gains
    else:
        return -self.lam * (-x) ** self.alpha  # Convex for losses
```

where x is the outcome of the lottery. Note that $v(0) = 0$ holds.

5.2.2 Weighting Separately over Gains and Losses

In section 2, we saw that Kahneman and Tversky define a piecewise, cumulative weighting function consisting of two weighting functions designed to weigh probabilities over gains separately than the probabilities over losses. More specifically, we saw cumulative probabilities were weighted as $w^+(p_i)$ over gains and $w^-(p_i)$ over losses:

```python
def prob_weight_plus(self, p):
    """Probability weighting for gains"""
    if p == 0: return 0
    if p == 1: return 1
    return (p ** self.gamma) / ((p ** self.gamma + (1-p) ** self.gamma) ** (1/self.gamma))

def prob_weight_minus(self, p):
    """Probability weighting for losses"""
    if p == 0: return 0
    if p == 1: return 1
    return (p ** self.sigma) / ((p ** self.sigma + (1-p) ** self.sigma) ** (1/self.sigma))
```

5.3 Expected Value Calculation

Given a lottery and an agent's parameters, Kahneman and Tversky assert that it is possible to capture this agent's opinion about a lottery. Representing the lottery as a two dimensional list, we can traverse through the outcomes and probabilities, taking aggregate sums as we move along. We start by checking if the thoutcome is a gain or a loss. If it is a gain, we take the sum from the i^{th} probability to the n^{th} probability, then again from the $i^{th} + 1$ probability to the n^{th}. The difference between the two aggregate sums is then computed, followed by the product of this weighted probability by the corresponding outcome:

```python
def expected_value(self, lottery):
    total = 0
    for i, (xi, pi) in enumerate(lottery):
        if xi >= 0:
            # Defines the aggregate sum from the ith probability to the nth probability
            w_plus_i = self.prob_weight_plus(sum([p for _, p in lottery[i:]]))
            # Check if we are at the end of the lottery
            if i < len(lottery)-1:
                # Defines the aggregate sum from the ith+1 to the nth probability
                w_plus_i_next = self.prob_weight_plus(sum([p for _, p in lottery[i+1:]]))
                total += self.value_function(xi) * (w_plus_i - w_plus_i_next)
            else:
                total += self.value_function(xi) * w_plus_i
```

However, if the i^{th} outcome is a loss we similarly take the sum from the $-m^{th}$ probability to the $i^{th}+1$ probability, then again from the $-m^{th}$ probability to the i^{th} probability. We then compute the difference between the two aggregate sums and multiply this difference by the corresponding outcome:

```
        else:
            # Defines the aggregate sum from the-mth probability to the ith+1 probability
            w_minus_i = self.prob_weight_minus(sum([p for _, p in lottery[:i+1]]))
            # Check if we are at the end of the lottery
            if i > 0:
                # Defines the aggregate sum from the-mth probability to the ith probability
                w_minus_i_prev = self.prob_weight_minus(sum([p for _, p in lottery[:i]]))
                total += self.value_function(xi) * (w_minus_i - w_minus_i_prev)
            else:
                total += self.value_function(xi) * w_minus_i
    return tota
```

6 Simulation Results

6.1 Demonstrating Loss Aversion

Our newly constructed simulation enables us toconduct ahypothetical studywitha hypothetical subject.For this subject,we assign the parameters the average values obtained from Kahneman and Tversky's study in 1992. That is, ($\alpha = 0.88$, $\gamma = 0.61$, $\delta = 0.69$, $\lambda = 2.25$).

```
agent = cpt(alpha=0.88, gamma=0.61, sigma=0.69, lam=2.25)
```

6.1.1 Computing the Value of Equivalent Gains/Losses

With our new agent, when can access its value function and pass a gain of 5 and an equivalent loss of 5 with the following lines:

```
agent.value_function(5)
agent.value_function(-5)
```

Output shows strong loss aversion:

```
The value assigned to a gain of 5 is 4.121863483573453
The value assigned to a loss of 5 is-9.274192838040268
Loss aversion holds here, as v(5) > v(-5)
```

The loss hurts more than twice as much as the equivalent gain pleases.

6.1.2 Graphing the Value Function

Using `matplot.lib` we can graph the value function with

```
# Plotting the value function on the x-value axis
x_values = list(range(-50, 51))
y_values = []
for x in x_values:
    y_values.append(agent.value_function(x))
plt.plot(x_values, y_values)
plt.grid(True)
plt.show()
```

With a domain and range of-50 to 51, we pass x-values (arithmetically increasing by 1) to our value function, which returns y-values. We can then graph our value function and compare it to the image in Section 2:

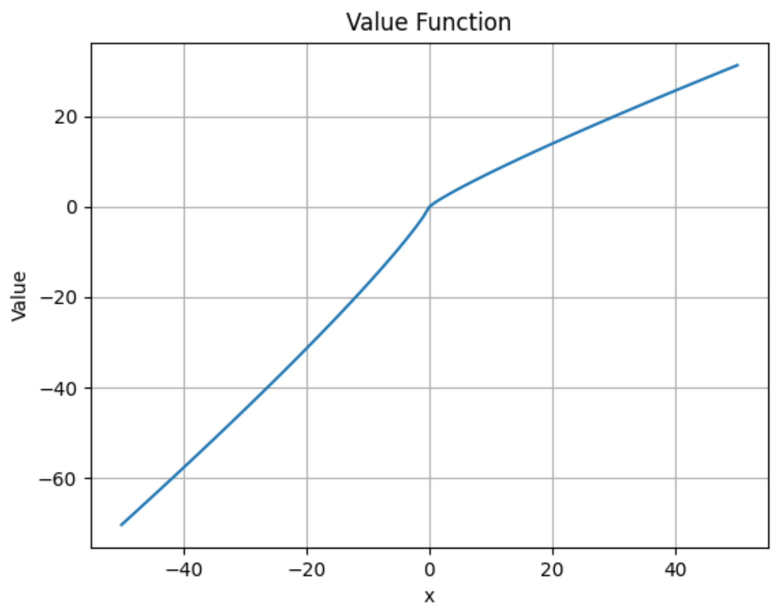

Figure 2: The CPT value function in our hypothetical simulation with $\alpha = 0.88$, $\gamma = 0.61$, $\delta = 0.69$, $\lambda = 2.25$

6.2 Probability Weighting Demonstration

The lotteries A and B represent a sure loss, and a highly unlikely but dramatic loss. C and D represent the same idea, but with a gain:

```python
# Small potential loss vs certain loss
A = [(-50, 0.001), (0, 0.999)]
B = [(-5, 1)]

# Small potential gain vs certain gain
C = [(50, 0.001), (0, 0.999)]
D = [(5, 1)]
```

Results show characteristic CPT patterns $E[v(A)] > E[v(B)]$ and $E[v(C)] < E[v(D)]$. That is:

```
Expected value of lottery A: -0.5987921826763616
Expected value of lottery B: -9.274192838040268
Lottery A is preferred over Lottery B
Expected value of lottery C: 0.019076915553650448
Expected value of lottery D: 4.121863483573453
Lottery D is preferred over Lottery C
```

6.3 Extreme Outcomes Test

We will now alter the gambles to contain more drastic outcomes.:

```python
lotteryA = [(-5000, 0.001), (0, 0.999)] # Example lottery [(xi, pi), ...]
lotteryB = [(-5, 1)]
lotteryC = [(5000, 0.001), (0, 0.999)]
lotteryD = [(5, 1)]
```

When the program is executed, we see that $E[v(A)] < E[v(B)]$ and $E[v(C)] < E[v(D)]$. That is:

```
Extreme loss lottery: 34.4569  # Overweighted tiny chance of big loss
Extreme gain lottery: 1.0978   # Underweighted tiny chance of big gain
```

Notice how the preference is the same over gains, but our agent is no longer comfortable taking a risk over losses, regardless of how unlikely that loss is. This is not only consistent with loss aversion, but it also shows how decision makers tend to exaggerate tail-end probabilities, as the chance of losing money in lottery A is small.

References

[1] Kahneman, D., & Tversky, A. (1992). Advances in prospect theory. *Journal of Risk and Uncertainty*, 5(4), 297-323.

[2] Barberis, N., Mukherjee, A., & Wang, B. (2016). Prospect theory and stock returns. *Review of Financial Studies*, 29(11), 3068-3107.

[3] DeStasio, K. L., Clithero, J. A., & Berkman, E. T. (2019). Neuroeconomics and health psychology. *Health Psychology Review*, 13(3), 313-324.