Paper 2: Autocorrelation Penalty Module

Formulation

The autocorrelation penalty discourages allocation adjustments that introduce predictable or autocorrelated return patterns, supporting statistical efficiency and randomness in signals.

The penalty is defined as:

$$\text{Penalty} = \lambda \left(1 - p_{\text{LB}}\right)$$

where:

- p_{LB}: p-value from the Ljung-Box test for no autocorrelation.
- λ: scaling parameter controlling penalty severity.

Why this Works

A higher p-value ($p_{\text{LB}} \approx 1$) suggests no autocorrelation, leading to a smaller penalty. Low p-values indicate significant autocorrelation, resulting in a larger penalty to discourage potential overfitting or unstable weight updates.

Algorithm Steps

1. Compute cumulative return series up to current time.
2. Perform Ljung-Box test on recent window to obtain p_{LB}.
3. Calculate penalty as $\lambda(1 - p_{\text{LB}})$.
4. Adjust weights using $(1 - \text{Penalty})$ scaling.
5. Normalize the weights to maintain the sum constraint.

Code Equivalent

```
from statsmodels.stats.diagnostic import acorr_ljungbox

def autocorr_penalty(cum_returns, penalty_scale=0.05):
    if len(cum_returns) < 20:
        return 0
    test_res = acorr_ljungbox(cum_returns[-20:], lags=[5], return_df=True)
    pval = test_res["lb_pvalue"].values[0]
    penalty = penalty_scale * (1 - pval)
    return penalty

# Example usage:
penalty = autocorr_penalty(cum_returns, penalty_scale=0.05)
adjusted_weights = weights * (1 - penalty)
adjusted_weights = np.maximum(adjusted_weights, 1e-5)
adjusted_weights /= np.sum(adjusted_weights)
```

Summary

- Reduces risk of predictable patterns in returns.
- Improves robustness against temporal dependencies.
- Proven effective in empirical signal de-correlation studies.