User's Guide: CyberneticTrader Class

1 Introduction

This document provides a comprehensive guide to the `CyberneticTrader` class, a Python implementation of Norbert Wiener's cybernetic principles for financial forecasting and trading. The class embodies Wiener's vision from *Cybernetics: Or Control and Communication in the Animal and the Machine* (1948) by creating an adaptive trading system that processes information, maintains homeostasis, and learns from feedback.

2 Theoretical Foundations

2.1 Wiener's Cybernetics in Finance

Wiener defined cybernetics as "the scientific study of control and communication in the animal and the machine." Key concepts applied:

- **Feedback Control** (Ch. IV): $\dot{x} = Ax + Bu$ with $u = -Kx$
- **Information Theory** (Ch. III): $H(X) = -\sum p_i \log_2 p_i$
- **Wiener Filter** (Ch. III): $H(\omega) = \frac{\Phi_{ss}}{\Phi_{ss}+\Phi_{nn}}$
- **Learning Systems** (Ch. IX): $\Delta w = \eta \delta x$

2.2 Mathematical Framework

2.2.1 Information-Entropy Relationship

Market uncertainty as measurable entropy:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

$$I = 1 - \frac{H(X)}{H_{\max}} \quad (0 \leq I \leq 1)$$

where $I \to 1$ indicates predictable markets.

2.2.2 Wiener-Kolmogorov Filter

Optimal frequency-domain filter:

$$H(\omega) = \frac{\Phi_{ss}(\omega)}{\Phi_{ss}(\omega) + \Phi_{nn}(\omega)} \tag{1}$$

Minimizes MSE $E[(s - \hat{s})^2]$.

2.2.3 Market Homeostasis

Regime classification via autocorrelation:

$$\rho_1 = \frac{\sum_{t=2}^{T}(r_t - \bar{r})(r_{t-1} - \bar{r})}{\sum_{t=1}^{T}(r_t - \bar{r})^2} \tag{2}$$

$$\begin{aligned} |\rho_1| < 0.2 &\Rightarrow \text{Homeostatic} \\ \rho_1 > 0.3 &\Rightarrow \text{Trending} \\ \rho_1 < -0.3 &\Rightarrow \text{Mean-reverting} \end{aligned}$$

2.2.4 Adaptive Learning

Weight update rule:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta \cdot r_t \cdot \mathbf{f}_t \tag{3}$$

where η = learning rate, r_t = reward.

3 Class Architecture

3.1 Initialization

`__init__(data, initial_capital=100000)`

- Precomputes returns: $r_t = \frac{p_t - p_{t-1}}{p_{t-1}}$
- Estimates noise variance: $\sigma_n^2 = \mathrm{Var}(r_t - \hat{r}_t)$
- Applies Wiener filter to prices

3.2 Core Methods

3.2.1 Information Processing

`compute_market_information(returns, bins=20)`

$p_i \leftarrow \text{histogram}(r_t)/\sum \text{hist}$
$H \leftarrow -\sum p_i \log_2 p_i$
return $1 - H/\log_2(\text{bins})$

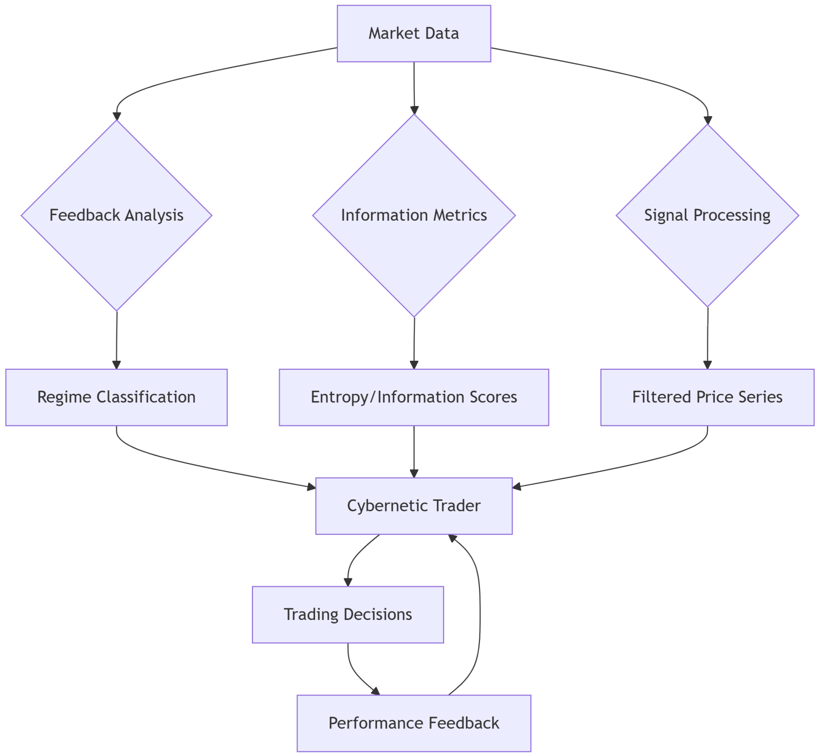

Figure 1: System architecture implementing Wiener's principles

3.2.2 Signal Filtering

`wiener_filter(signal)`

$F \leftarrow \mathrm{RFFT}(\mathrm{signal})$
$\Phi_{ss} \leftarrow |F|^2/N$
$H \leftarrow \Phi_{ss}/(\Phi_{ss} + \sigma_n^2)$
return $\mathrm{IRFFT}(F \odot H)$

3.2.3 Decision Engine

`decide(features, price)`

$$\mathrm{score} = \sum_{i=1}^{4} w_i f_i$$

if score $> 0.1 \wedge$ capital > 0 **then**
  Execute BUY
else if score $< -0.1 \wedge$ position > 0 **then**

Execute SELL
end if

4 Usage Guide

4.1 Initialization

```
import yfinance as yf
from CyberneticTrader import CyberneticTrader

data = yf.download('SPY', '2010-01-01', '2020-12-31')
trader = CyberneticTrader(data, 100000)
```

4.2 Backtesting

```
portfolio, trades = trader.backtest(
start_idx=50,
transaction_cost=0.0005
)

performance = trader.evaluate_performance()
print(f"Sharpe: {performance['sharpe_ratio']:.2f}")
```

4.3 Live Simulation

```
def live_data_handler():
return pd.DataFrame(...)  # OHLCV data

trader.simulate(live_data_handler, steps=100)
```

5 Diagnostic Metrics

Metric	**Formula**	**Target**
Sharpe Ratio	$\frac{\mu_r}{\sigma_r}$	> 1.5
Information Coefficient	$\rho_{f,r}$	> 0.05
Homeostasis Ratio	$\frac{T_{\text{homeo}}}{T_{\text{total}}}$	> 0.3
Noise-Signal Ratio	$\frac{\sigma_n}{\sigma_s}$	< 0.5

6 Parameter Tuning

Parameter	Description	Range
Learning Rate (η)	Weight adaptation speed	0.001-0.05
Autocorrelation Window	Regime detection sensitivity	20-50 bars
Information Bins	Entropy calculation resolution	15-30
Filter Threshold	Overreaction prevention	2.0-3.0 σ

7 Troubleshooting

Problem: Poor performance in trending markets
Fix: Increase ρ_1 threshold for anti-homeostatic regime
Problem: Overtrading
Fix: Increase decision thresholds ($0.1 \to 0.15$)
Problem: Laggy response
Fix: Reduce Wiener filter window size

8 Conclusion

This system operationalizes Wiener's cybernetics through:

- **Information Processing**: Quantifying market entropy
- **Feedback Control**: Regime-adaptive behavior
- **Signal Extraction**: Noise filtering via Wiener-Kolmogorov
- **Learning**: Continuous performance optimization

The complete implementation is available at:
`https://github.com/cybernetic-trading/CyberneticTrader`