Enhancing Robust Parameter Selection in Trading Strategy Optimization

Jeff Tuche :)

1 Understanding the Role of Robust Parameter Selection After Random Forest Learning

After training the Random Forest (RF) model, we now focus on extracting the most robust parameter sets. At first glance, it may seem like we are repeating the filtering process already handled by RF. However, this step provides additional refinement by explicitly analyzing the stability of the selected parameter ranges.

2 What Was Done in the Random Forest Step?

Previously, we trained an RF model to learn from a dataset where:

- Each row represented a full parameter set (moving averages, envelopes, position sizes).
- Each row included performance metrics (Sharpe ratios for train, test, and degradation).

2.1 What the RF Model Learned

- How input parameters influence performance on test Sharpe, train Sharpe, and Sharpe degradation.
- Which parameter values consistently lead to good performance.
- General patterns in parameter stability across different time windows.

2.2 What RF Did Not Do

- It did **not explicitly extract stable parameter ranges**; it only made predictions.
- It did **not quantify which parameters have high variance** across the top-performing sets.
- It did **not remove overfitted parameters**; it just predicted performance based on historical patterns.

3 What Does Robust Parameter Selection Add?

After the RF step, we use its predictions to extract the most stable and robust parameter ranges.

3.1 Key Difference Between RF and This Step

- RF trained on all data and made **predictions** for test Sharpe, train Sharpe, and Sharpe degradation.
- This function **uses those predictions to extract only the most robust and stable parameter values**.

3.2 How This Function Works

1. Load the exact same features used in RF training.
2. Use the trained RF model to predict test Sharpe ratios.
3. Select the top 20% of predicted Sharpe values (threshold-based filtering).
4. Analyze the **stability of parameters** inside that top 20% set:
    - Compute mean, standard deviation (std), coefficient of variation (CV), and range for each parameter.
    - CV helps identify parameters that fluctuate too much (unstable).

4 Link to RF and Improvements Needed

Currently, the function only filters on predicted test Sharpe, but we now have three predictions:

- **Train Sharpe (Predicted)**
- **Test Sharpe (Predicted)**
- **Sharpe Degradation (Predicted)**

4.1 What Needs Improvement?

Instead of filtering only on predicted test Sharpe, we should:

- Keep only parameter sets that have:
    - **High test Sharpe** (good performance on unseen data).
    - **Moderate to high train Sharpe** (to avoid overfit models).
    - **Low Sharpe degradation** (consistent performance between train & test).
- Use **multi-objective filtering** instead of just selecting the top 20% Sharpe.

5 Example of How This Works

5.1 Before Filtering (Random Forest Predicted Values)

Parameter Set	Pred. Train Sharpe	Pred. Test Sharpe	Pred. Sharpe Degradation
Set 1	2.5	2.6	-0.1
Set 2	3.1	2.1	1.0 (Degraded significantly)
Set 3	1.8	2.4	-0.6
Set 4	2.7	2.8	-0.1 (Stable and robust)
Set 5	1.2	2.9	-1.7 (Possible overfitting)

5.2 How We Improve Filtering

- **Set 2 is removed** because test Sharpe dropped a lot (unstable).
- **Set 5 is removed** because it had very low train Sharpe but high test Sharpe (possible overfitting).
- **Only stable sets like Set 1, Set 3, and Set 4 are kept**.

6 What We Learn from This Filtering

After selecting robust parameter sets, we compute stability metrics per parameter:

Parameter	Mean	Std Dev	CV (Lower = More Stable)	Range
btc_ma_window	11.65	4.42	0.379	14.00
btc_size	0.074	0.027	0.367	0.090
btc_envelope_1	0.071	0.012	0.166	0.040
btc_envelope_2	0.104	0.009	0.088	0.030

6.1 Key Insights from Stability Metrics

- **btc_ma_window has a high range (14) and CV = 0.379** $\rightarrow$ This means it fluctuates a lot across good models and might not be critical.
- **btc_envelope_2 has a very low CV (0.088)** $\rightarrow$ This suggests it is a highly stable parameter and should be prioritized.

7 Summary of the Updated Approach

What RF Did:

- Learned patterns from past backtests and predicted test, train, and degradation Sharpe.

What We Are Doing Now:

- Filtering parameter sets **not just by test Sharpe but also considering train Sharpe and degradation**.
- Extracting stable ranges for each parameter (mean, std, CV, and range).
- Ensuring we keep only parameter configurations that generalize well.

Next Steps:

- Use Gaussian Filtering to smooth out distributions and refine ranges.
- Use Bayesian Optimization within the stable range to fine-tune.

By incorporating train, test, and degradation Sharpe, we ensure our model doesn't just pick the best-performing parameters but selects the most stable ones for real-world deployment.