LSTM
Modelo Double Headed Multi-Horizon
Documentación técnica y fórmulación

Santiago Javier Espino Heredero

September 23, 2025

Abstract

Documento técnico que recoge: optimización del precio medio de entrada mediante órdenes escalonadas, extensión del modelo a multi-horizon, utilidad de la predicción de volatilidad, métricas de monitorización (tick, liquidez, latencia, órdenes globales), reglas de ejecución y pseudocódigo del monitor loop. Incluye además un resumen de la anatomía y uso de la LSTM en este flujo.

Contents

1 Optimizar el precio medio de entrada con órdenes escalonadas

Si el modelo estima que el retorno esperado en el próximo horizonte es positivo, en vez de ejecutar la posición total de golpe se pueden realizar *entradas escalonadas* (dentro del mismo horizonte h). Esto reduce el precio medio de ejecución y el riesgo de entrar en un pico.

Matemáticamente, si la señal en t es

$$\widehat{r}_{t\to t+h},$$

entonces se puede decidir ejecutar compras/ventas en los tiempos

$$\{t,\ t+\Delta t,\ t+2\Delta t,\ \dots\},$$

donde cada Δt es un sub-intervalo dentro de $[t, t+h]$. En cada checkpoint se re-evalúa la predicción usando la información más reciente; si la predicción se deteriora, se pueden detener futuras compras o cerrar posiciones ya abiertas.

1.1 Ventajas

- Optimiza el precio medio de ejecución.
- Reduce slippage e impacto de mercado.
- Permite abortar/limitar exposición si la señal se debilita.

1.2 Esquema matemático de reparto

Si la posición objetivo es Q y se divide en N tranches iguales $q_i = Q/N$, entonces el orden medio ejecutado hasta el tiempo t_k es

$$\overline{P}_{t_k} = \frac{1}{k}\sum_{i=1}^{k} P_{t_i}^{(exec)},$$

y la estrategia busca minimizar $\overline{P}_{t_N}$ (para compra) sujeto a constraints de liquidez y riesgo.

2 Modelo multi-horizon

2.1 Concepto

Un modelo **multi-horizon** produce predicciones para varios horizontes simultáneamente:

$$\widehat{r}(h_1),\ \widehat{r}(h_2),\ \ldots,\ \widehat{r}(h_k),$$

con sus correspondientes predicciones de volatilidad

$$\widehat{\sigma}(h_1),\ \widehat{\sigma}(h_2),\ \ldots,\ \widehat{\sigma}(h_k).$$

Ejemplo práctico con tres horizontes:

$$h_1 = 5 \text{ min}, \quad h_2 = 30 \text{ min}, \quad h_3 = 240 \text{ min}(= 1\text{D}).$$

El vector de salida podría ser:

$$\widehat{\mathbf{y}} = \big(\widehat{r}(5),\ \widehat{r}(30),\ \widehat{r}(240),\ \widehat{\sigma}(5),\ \widehat{\sigma}(30),\ \widehat{\sigma}(240)\big).$$

2.2 Uso operativo

- **Entrada condicionada:** solo abrir si horizontes corto y medio están alineados en signo y magnitud (por ejemplo $\widehat{r}_{30m} > T_{30}$ y $\widehat{r}_{4h} > 0$).
- **Mantenimiento:** mantener si horizonte largo confirma la dirección.
- **Tolerancias:** thresholds ajustables por backtest.

2.3 Coste y requerimientos

Más salidas significa más etiquetas y mayor complejidad de entrenamiento. Requiere:

- más datos etiquetados
- posible reponderación de pérdidas por horizonte
- control de overfitting (regularización)

3 Utilidad de la predicción de volatilidad

La predicción de volatilidad $\widehat{\sigma}$ sirve para:

1. **Dimensionamiento de posición (position sizing).** Un heurístico inspirado en Kelly parcial puede ser:
$$f^* \propto \frac{\widehat{r}}{\widehat{\sigma}^2},$$
de forma que a mayor volatilidad predicha, menor sizing para preservar riesgo.
2. **Gestión del stop-loss / take-profit.** Si $\widehat{\sigma}$ es alta, ampliar stops; si baja, usar stops más ajustados.
3. **Selección de mercado.** Preferir activos con menor $\widehat{\sigma}$ para estrategias de baja varianza o activos con mayor $\widehat{\sigma}$ si se busca mayor apalancamiento (con control).
4. **Filtrado de señales falsas.** Una señal con $\widehat{r}$ pequeño y $\widehat{\sigma}$ grande tiene baja probabilidad de éxito.

Resumen: la volatilidad es un estimador de riesgo en el horizonte, no una señal direccional directa.

4 Evaluación de los horizontes $\{30\mathbf{m}, 4\mathbf{h}, 1\mathbf{D}\}$

4.1 Propuesta

Usar como salidas multi-horizon: $h_1 = 30\text{m},\ h_2 = 4\text{h},\ h_3 = 1\text{D}$.

4.2 Ventajas

- Cobertura jerárquica: corto + medio + largo plazo.
- Flexibilidad operativa: intradía, swing corto y overnight.
- Mejora del sizing usando $\widehat{\sigma}$ por horizonte.

4.3 Inconvenientes / riesgos

- Mayor cantidad de etiquetas y coste de entrenamiento.
- Necesidad de más datos para cada horizonte.
- Diferentes escalas de ruido y señal entre horizontes.

4.4 Reglas operativas de ejemplo

(a) Abrir posición si $\widehat{r}_{30m} > T_{30}$ y $\widehat{r}_{4h} > 0$.

(b) Mantener si $\widehat{r}_{1D} > 0$ y $\widehat{r}_{4h}$ no se invierte.

(c) Re-evaluar cada Δt: por ejemplo $\Delta t = 1$ vela de 30m para intradía.

4.5 Sugerencia práctica de thresholds

Umbrales iniciales a validar por backtest: T_{30} en rango 0.05%–0.2% (ajustar por activo y liquidez).

5 Monitor Loop constante: métricas a medir y lógica de decisión

5.1 Métricas principales

1. **Tick price:** último price / bid / ask.
2. **Spread:** spread $= \text{ask}_0 - \text{bid}_0$, spread_pct $= \frac{\text{spread}}{\text{mid}}$.
3. **Orderbook depth:** volumen acumulado en top L niveles (bids/asks).
4. **Liquidity score:** por ejemplo

$$\text{liq\_score} = \frac{\sum_{i=1}^{L}(\text{vol\_bid}_i + \text{vol\_ask}_i)}{\text{spread} + \varepsilon},$$

   normalizado por tick size o por base currency.
5. **Latency:** RTT hacia exchange (REST + websocket).
6. **Fill rate / execution rate:** % de órdenes llenadas en T segundos (histórico).

7. **Global executed orders:** volumen del mercado en ventana M (indicador de actividad).
8. **Imbalance ratio:** imbalance $= \frac{\text{bid\_depth}}{\text{ask\_depth}}$.
9. **Model signal:** vector multi-horizon $\{\widehat{r}_{30}, \widehat{r}_{4h}, \widehat{r}_{1D}, \widehat{\sigma}_{30}, \dots\}$.
10. **Position / ledger state:** posición actual, coste medio, PnL unrealizado.

5.2 Frecuencias de muestreo

- Tick / orderbook: websocket, 10–100 Hz (según exchange).
- Agg. de métricas: cada 0.5–5 s (ejecución intradiaria), cada 1 min para decisiones menos frecuentes.
- Re-evaluación del modelo: cada 1 vela base (ej. cada 30m) o cada N segundos si re-evaluas por tick.

5.3 Lógica de gating pre-ejecución

Antes de enviar una orden verificar:

- spread_pct $<$ spread_max,
- liq_score $>$ liq_min,
- latency $<$ lat_max,
- rate-limit safe (no exceder órdenes/min).

5.4 Dynamic sizing (ejemplo)

Proponer tamaño proporcional a:

$$\text{size} \propto \frac{\widehat{r}}{\widehat{\sigma}^2} \cdot \min\left(1, \frac{\text{liq\_score}}{K}\right).$$

5.5 Abort rules

Si durante la ejecución la liquidez cae por debajo de un umbral o la latencia aumenta excesivamente, cancelar o reducir órdenes.

6 Estrategias de ejecución dentro del horizon (entradas escalonadas)

6.1 Objetivo

Minimizar precio medio de compra/venta y el impacto de mercado, re-evaluando la señal entre tramos.

6.2 Estrategia A: DCA re-evaluado

- Dividir Q en N tramos $\{q_i\}$.
- Definir checkpoints $t_0, t_1, \dots, t_{N-1}$ en $[t, t+h]$.
- En cada t_i: re-evaluar $\widehat{r}$; si mantiene signo y liq ok, enviar q_i; si cae por debajo de T_{stop}, detener.

6.3 Estrategia B: VWAP / orderbook-aware slicing

- Estimar profundidad necesaria para ejecutar Q sin mover precio $>$ X bps.
- Ejecutar como secuencia de limit orders pseudo-VWAP.
- Si el mercado se mueve en contra, cancelar remanente.

6.4 Estrategia C: Adaptive sizing por liquidez y volatilidad

Cada tranche q_i se calcula:

$$q_i = Q \cdot w_i, \qquad w_i \propto \sigma\Big(\gamma(\text{liq\_score} - L_0)\Big) \cdot \frac{\widehat{r}}{\widehat{\sigma}^2},$$

donde $\sigma(\cdot)$ es la función sigmoide para saturar en alta liquidez:

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

7 Uso agregado de tick + liquidez + latencia + órdenes globales

Combinando estas métricas se decide:

- Market vs limit orders.
- Tamaño y timing de cada tranche.
- Activación de circuit breakers en eventos extremos.

Order Manager Mantener un módulo (Order Manager) que reciba:

- request: (side, target_qty, urgency)
- current metrics: (spread, liq_score, latency)

y produzca un plan (market/limit, tranches, pacing). Registrar todo en ledger.

8 Integración en el stack (Streamlit / backtest / live)

8.1 Arquitectura sugerida

- **Model Service:** API/módulo que devuelve $\widehat{r}, \widehat{\sigma}$ multi-horizon.
- **Market Connector:** CCXT + websockets (ticks, orderbook) + REST (órdenes).
- **Order Manager:** pacing/slicing + risk gates.
- **Monitor Loop:** proceso que consume ticks, actualiza métricas, re-evalúa y manda acciones al Order Manager.
- **UI (Streamlit):** controles manuales, dashboard, botones de control.

8.2 Backtest vs Live

- **Backtest**: simular orderbook depth y fills; inyectar latencias.
- **Paper/Testnet**: usar CCXT sandbox / endpoints testnet; mismo código que live.

9 Tolerancias, rate limits y seguridad

- Rate limits: token bucket / contador órdenes/min.
- Slippage max: por defecto 0.5%–1%.
- Circuit breakers:
  - latency > threshold por > X seg ⇒ pausar trading.
  - spread_pct > max_spread por > Y seg ⇒ pausar.
  - fills_fail_ratio > p_fail ⇒ pausar y alertar.
- Key security: no persistir claves en texto; usar env vars o secret stores.

10 Telemetría y métricas mínimas a guardar

Guardar en el ledger y/o timeseries DB:

- timestamp, price, side, qty, price_requested, price_executed, fee, latency_at_send, liq_score_at_send.
- model signals at decision time (vector multi-horizon).
- orderbook snapshot summary (top5 levels).
- health metrics: RTT, websocket reconnects, orders/min.

11 Pseudocódigo del Monitor Loop (alto nivel)

```
# Monitor loop (simplificado)
while running:
   tick = read_latest_tick() # via websocket
   ob = read_orderbook_snapshot() # top N
   liq_score = compute_liq_score(ob)
   latency = measure_latency()
   global_vol = get_global_vol_window()
   model_input = build_features_from_history(history_buffer, ob, tick,
       extra=...)
   signals = model.predict(model_input) # returns dict with multi-horizon
       ret/vol
   # gating
   if spread_pct(tick, ob) > MAX_SPREAD or liq_score < LIQ_MIN or latency >
        LAT_MAX:
      log("Abort send: market not suitable")
      sleep(SHORT)
      continue
   # decide to enter (example)
   if signals['30m'] > T_30 and signals['4h'] > 0:
      plan = order_manager.plan_entry(side='buy', target_qty=Q, ob=ob,
          liq_score=liq_score, signals=signals)
      for tranche in plan.tranches:
         if should_send_tranche(tranche, current_metrics):
            res = order_manager.send_order(tranche)
            ledger.append(res)
            if res.failed: handle_fail()
         reeval_signals = maybe_reevaluate_model_if_needed()
         if reeval_signals.indicate_abort: break
   sleep(POLL_INTERVAL)
```

12 Backtesting realista (orderbook-aware)

- Simular fills usando snapshots históricas de orderbook: consumir niveles y "comer" liquidez hasta qty.
- Inyectar latencia de modelo y de envío (ej. 50–500 ms).
- Calcular slippage por profundidad usada (impact price).

Esto es imprescindible para validar DCA vs market/limit.

13 Recomendaciones concretas de implementación

1. Implementar multi-horizon: 3 salidas de retorno + 3 de volatilidad. Entrenar por fases:
   - primera fase: solo 30m,
   - luego añadir 4h y 1D.
2. Implementar monitor loop con métricas: spread, liq_score, latency, fills_rate.
3. Empezar en Paper/Testnet con policy DCA simple: $N = 4$ tranches en el horizon, re-evaluar tras cada tranche.
4. Añadir circuit breakers y rate-limit guard.
5. En Order Manager, preferir limit orders si liq_score alto; market si urgencia + liq_score alto + spread pequeño.
6. Agregar backtest orderbook-aware con latencia simulada.

Apéndice: Anatomía breve de la LSTM (resumen técnico)

La LSTM procesa secuencias con tensores:

$$X \in \mathbb{R}^{\text{batch\_size} \times seq\_len \times n_{feat}}.$$

En cada timestep t y para cada celda:

$$\begin{aligned}
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f),\\
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i),\\
g_t &= \tanh(W_g[h_{t-1}, x_t] + b_g),\\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t,\\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o),\\
h_t &= o_t \odot \tanh(c_t).
\end{aligned}$$

Salidas finales (two-heads):

$$\widehat{y}^{(ret)} = W_{ret} h_T + b_{ret}, \qquad \widehat{y}^{(vol)} = W_{vol} h_T + b_{vol}.$$

Conclusión

El enfoque descrito combina robustez estadística (multi-horizon + volatilidad), ejecución consciente del orderbook (DCA adaptativo, VWAP slicing) y control operacional (monitor loop, circuit breakers). Implementarlo por fases — primero en backtest con orderbook histórico y latencias simuladas, luego en paper/testnet— minimiza riesgos y permite calibrar thresholds y parámetros.