Heterocedasticidad en LSTM para series financieras: Guía Completa

Santiago Javier Espino Heredero

2025-10-14

Resumen

Esta guía técnica presenta fundamentos, arquitectura e implementaciones prácticas para modelar heterocedasticidad en redes recurrentes (LSTM) aplicadas a trading. Se detallan las motivaciones financieras, la parametrización numéricamente estable (log-variance), la arquitectura propuesta (cabezas heterocedásticas), funciones de pérdida, procedimientos de entrenamiento y evaluación, así como recomendaciones de despliegue e integración con proyectos existentes.

Índice

1. Fundamentos Matemáticos

1.1. ¿Qué es la Heterocedasticidad?

En regresión clásica se asume ruido i.i.d. con varianza constante (homocedasticidad):

$$y_t = f(x_t) + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, \sigma^2), \quad \sigma^2 \text{ constante.}$$

La heterocedasticidad indica que la varianza está condicionada a la información disponible:

$$\varepsilon_t \sim \mathcal{N}(0, \sigma_t^2), \quad \sigma_t^2 = g(x_t) \text{ o bien } \sigma_t^2 = \text{función de la historia.}$$

Es decir, el ruido es condicional a la información disponible y no constante en el tiempo.

1.2. Parametrización Numéricamente Estable

Para garantizar positividad de la varianza y estabilidad numérica se modela la *log-variance*:

$$s_t \equiv \log(\sigma_t^2) \in \mathbb{R},$$

la verosimilitud gaussiana por muestra t puede escribirse (negativa del log-likelihood):

$$\ell_t = -\log p(y_t \mid x_t) = \tfrac{1}{2}\left[\exp(-s_t)(y_t - \mu_t)^2 + s_t + \log(2\pi)\right].$$

Ventajas:

- s_t es real sin restricciones; $\exp(s_t) = \sigma_t^2$ garantiza positividad.
- Se evitan divisiones por cero directas y se obtienen gradientes más estables.

Forma simplificada para optimización Se usa la pérdida por muestra (sin constantes):

$$L_{\text{NLL}}(\mu, s, y) = \tfrac{1}{2}\left[\exp(-s)(y - \mu)^2 + s\right].$$

Los dos términos juegan papeles complementarios: el primero penaliza el error ponderado por confianza ($1/\sigma^2$) y el segundo penaliza varianzas excesivas.

2. Por Qué Importa en Finanzas

2.1. Volatility Clustering

Los mercados exhiben "volatility clustering": períodos de alta volatilidad que se suceden y períodos de calma. Modelos clásicos como GARCH o modelos de volatilidad estocástica (SV) son ejemplos de varianzas condicionales.

2.2. Limitaciones de MSE

El uso de MSE (suponiendo varianza constante) implica:

1. Ignorar información de incertidumbre por muestra.
2. No disponer de medidas para decisiones 'risk-aware'.
3. Penalizar outliers de la misma forma que errores en condiciones de baja volatilidad.
4. Tomar decisiones subóptimas en mercados con regímenes de volatilidad.

2.3. Beneficios de modelar σ_t

Modelar la incertidumbre por muestra permite:

- Obtener intervalos de confianza $\mu \pm k\sigma$.
- Aplicar reglas de trading condicionadas al SNR $= |\mu|/\sigma$.
- Mejor scoring mediante NLL que penaliza adecuadamente heterocedasticidad.
- Dimensionamiento de posición adaptativo (inversamente proporcional a σ_t).
- Detección de cambios de régimen mediante aumentos de σ_t.

3. Arquitectura LSTM Heterocedástica

3.1. Baseline: LSTM2Head

Resumen de la arquitectura existente (extracto conceptual): una LSTM seguida de dos cabezas que predicen retornos y volatilidad (punto) usando activaciones convencionales y pérdida MSE.

3.2. HeteroHead: cabeza para μ y $s = \log \sigma^2$

Proponemos una cabeza que prediga simultáneamente la media μ y la log-var s. Se aplican técnicas de regularización (dropout) y un clamp sobre s para estabilidad numérica.

```
class HeteroHead(nn.Module):
    def __init__(self, hidden_size):
        mid = max(8, hidden_size // 2)
        self.mu = nn.Sequential(...)
        self.log_var = nn.Sequential(...)
    def forward(self, x):
        mu = self.mu(x).squeeze(-1)
        s = self.log_var(x).squeeze(-1)
        s = torch.clamp(s, min=-12.0, max=8.0)
        return mu, s
```

El clamp recomendado $s \in [-12, 8]$ corresponde a $\sigma^2 \in [\exp(-12), \exp(8)] \approx [6 \times 10^{-6}, 3 \times 10^3]$.

3.3. Modelo completo: `LSTM4HeadsHetero`

Se propone una LSTM compartida y cuatro cabezas heterocedásticas (ret, high, low, vol), cada una produciendo (μ, s). Esto permite incertidumbre específica por target manteniendo eficiencia computacional.

4. Implementación PyTorch

4.1. Función de pérdida: Gaussian NLL

```
def hetero_nll_loss(mu, log_var, y, reduction='mean'):
    inv_var = torch.exp(-log_var)
    mse_term = inv_var * (y - mu) ** 2
    log_term = log_var
    loss = 0.5 * (mse_term + log_term)
    if reduction == 'mean':
        return loss.mean()
    elif reduction == 'sum':
        return loss.sum()
    else:
        return loss
```

4.2. Pérdida combinada para 4 cabezas

Se calcula NLL por cabeza y se combinan con pesos y una penalización opcional sobre $\log \sigma$ (anti-inflación):

```
# combined_hetero_loss(outputs, targets, weights, penalty_sigma)
# retorna total_loss y componentes para logging
```

Pesos recomendados: $w_{ret} = 1{,}0$, $w_{high} = w_{low} = 0{,}8$, $w_{vol} = 0{,}5$, penalty_sigma $= 1 \times 10^{-3}$.

4.3. Bucle de entrenamiento y evaluación

Se proporciona un `train_epoch_hetero(...)` y un `eval_epoch_hetero(...)` incluyendo opciones de mixed precision, gradient clipping, acumulación de métricas y manejo de tensores en GPU.

5. Inicialización y Estabilidad

5.1. Problema

Inicializaciones arbitrarias de la cabeza de log-var pueden producir predicciones extremas de s (colapso o inflación), llevando a NaNs o gradientes inestables.

5.2. Solución: inicializar bias con varianza empírica

Se propone fijar el bias de la última capa del branch `log_var` a $\log(\widehat{\mathrm{Var}}_{train})$ para cada target:

```
def initialize_hetero_heads(model, y_train_dict):
    for name, y_data in y_train_dict.items():
        emp_var = np.var(y_data) + 1e-8
        init_log_var = np.log(emp_var)
        last_layer = head.log_var[-1]
        last_layer.bias.fill_(init_log_var)
```

5.3. Gradient clipping y LR schedule

Gradient clipping (norma máxima recomendada 1.0) y scheduler ReduceLROnPlateau son medidas esenciales para la estabilidad. Recomendación de LR inicial: 1×10^{-4} con weight decay 1×10^{-5}.

6. Calibración de Incertidumbre

6.1. Concepto

La calibración evalúa si las σ predichas reflejan el error observado; idealmente aproximadamente 68 % de errores dentro de $\pm 1\sigma$, 95 % dentro de $\pm 2\sigma$, etc.

6.2. Función de calibración

Se presenta una función que calcula fracciones observadas dentro de $k\sigma$ y compara con los valores teóricos gaussianos. Se define tolerancia: diferencia $< 5\,\%$ buena, $< 10\,\%$ aceptable.

6.3. Acciones ante mal calibrado

- Si σ subestimado: aumentar `penalty_sigma` o revisar inicialización.
- Si σ sobreestimado: reducir `penalty_sigma` o añadir regularización.
- Si asimetría: considerar likelihood Student-t.

7. Uso en Trading (Inference)

7.1. SNR y regla de decisión

Se define SNR como:

$$\text{SNR} = \frac{|\mu|}{\sigma}.$$

Regla de decisión simple: operar sólo si SNR >umbral (recomendado 1.5–2.0, calibrable vía backtests).

7.2. Función de predicción con incertidumbre

Se describe una función que devuelve μ, σ, SNR e intervalos de confianza por cada target.

7.3. Lógica de trading risk-aware

Se propone una estrategia de entradas (entry1, entry2), take-profit y stop-loss basados en predicciones de high/low y tamaño de posición inversamente proporcional a σ:

$$\text{position\_size} = \frac{\text{risk\_per\_trade}}{1 + 10 \cdot \sigma_{ret}}.$$

8. Problemas Comunes y Soluciones

Se enumeran problemas típicos (sigma inflado, colapso, gradientes inestables, calibración pobre, overfitting) y soluciones concretas: penalización sobre `log_var`, clamp en s, gradient clipping, inicialización con varianza empírica, regularización y early stopping.

9. Comparación: MSE vs Hetero NLL

Tabla 1: Comparativa resumida entre MSE (baseline) y Hetero NLL

Aspecto	MSE (LSTM2Head)	Hetero NLL (LSTM4HeadsHetero)
Outputs por head	1 (solo μ)	2 (μ y $s = \log \sigma^2$)
Varianza	Constante	Condicional (dependiente de x_t)
Incertidumbre	No disponible	por muestra
Intervalos	No	Sí ($\mu \pm k\sigma$)
SNR	No	Sí (filtrado risk-aware)
Volatility clustering	Ignorado	Capturado
Position sizing	Fijo	Adaptativo (1/)
Complejidad	Baja	Media (+50 % parámetros)
Estabilidad training	Alta	Media (requiere init y clipping)
Costo computacional	1x	1.3x

10. Integración con Proyecto Actual

Se presentan fases de migración backward-compatible: añadir clases y funciones heterocedásticas en `fiboevo.py`, extender `create_sequences_from_df` con `include_high_low`, actualizar `prepare_dataset.py` con flags `-use-hetero` y adaptar el trading daemon para consumir salidas con incertidumbre.

10.1. Puntos clave de compatibilidad

- `include_high_low=False` por defecto para no romper pipelines existentes.
- Si `use_hetero=True`, los DataLoaders y TensorDatasets deben contener 5 tensores (X, y_ret, y_vol, y_high, y_low).
- El daemon de trading debe verificar `use_hetero` y usar SNR filtering para decisiones.

11. Configuración Recomendada

Se listan configuraciones sugeridas para distintos entornos (GTX 1070 — 8GB, GPU potente — 16GB+, CPU-only). Ejemplo (GTX 1070):

```
config_hetero = {
  'input_size': 35,
  'hidden_size': 128,
  'num_layers': 2,
  'dropout': 0.15,
  'batch_size': 64,
  'seq_len': 32,
  'horizon': 10,
  'epochs': 50,
  'learning_rate': 1e-4,
  'weight_decay': 1e-5,
  'grad_clip': 1.0,
  'use_amp': True,
```

```
  'w_ret': 1.0,
  'w_high': 0.8,
  'w_low': 0.8,
  'w_vol': 0.5,
  'penalty_sigma': 1e-3,
}
```

12. Referencias y Recursos

Se enumeran trabajos clave sobre incertidumbre en redes neuronales, modelos clásicos de heterocedasticidad y herramientas de referencia (PyTorch GaussianNLLLoss, TensorFlow Probability, Pyro) así como libros y tutoriales relevantes.

Apéndice A: Glosario de Términos

Término	Definición
Heterocedasticidad	Varianza no constante (condicional a x_t)
Homocedasticidad	Varianza constante (σ^2 fija)
NLL	Negative Log-Likelihood
log_var	$\log(\sigma^2)$, parametrización estable de varianza
SNR	Signal-to-Noise Ratio = $\lvert\mu\rvert/\sigma$
Calibración	Qué tan bien σ predicho refleja el error real
Aleatoric uncertainty	Incertidumbre en los datos (ruido)
Epistemic uncertainty	Incertidumbre del modelo (falta de datos)
Volatility clustering	Alta volatilidad seguida de alta volatilidad
Mixed precision (AMP)	Entrenamiento FP16 + FP32

Apéndice B: Checklist de Implementación

- ☐ ‘high‘ y ‘low‘ en df_ohlcv (disponibles)
- ☐ ‘create_sequences_from_df‘ devuelve 5 arrays cuando ‘include_high_low=True‘
- ☐ ‘DataLoader‘ con 5 tensores (xb, yret, yvol, yhigh, ylow)
- ☐ Modelo ‘LSTM4HeadsHetero‘ instanciado correctamente
- ☐ ‘initialize_hetero_heads()‘ llamado con ‘y_train_dict‘
- ☐ Learning rate moderado (1e-4)
- ☐ Gradient clipping activado (1.0)
- ☐ Mixed precision habilitado si GPU lo soporta

Apéndice C: Troubleshooting Quick Reference

Síntoma	Causa probable	Solución rápida
Loss → NaN	σ colapsa	Añadir ‘clamp(s,-12,8)‘
Loss no converge	LR muy alto	Reducir a 1×10^{-4} o 1×10^{-5}
Gradientes explotan	Sin clipping	‘clip_grad_norm$_(params, 1,0)$‘
σ muy grande	Modelo "hace trampa"	Aumentar ‘penalty$_sigma$‘$a5e-3$
Calibration pobre	Init incorrecto	Llamar ‘initialize$_hetero_heads()$‘
OOM en GTX 1070	Batch/hidden grande	Reducir batch=32, hidden=96

Documento creado: 2025-10-14
Tema: Análisis de heterocedasticidad para LSTM de series financieras
Versión: 1.0
Estado: Documentación completa lista para implementación