Learning Time Series

Jens Christian Keil | j@polyhelion.com

London, 16.05.2025

A high-level technical overview of the current state of time series modelling in in general and in the financial services context in particular.

Linear Parametric Models

$$\Delta^d X_t = \sum_{i=1}^{p} \phi_i \, \Delta^d X_{t-i} + \epsilon_t + \sum_{j=1}^{q} \theta_j \, \epsilon_{t-j}$$

Linear combinations of autoregressive, moving average and/or latent (trend, seasonality, etc.) components. Common examples are ARIMA, VAR or ETS.

Hyperparameter selection
Optimizing information criteria, usually AIC, BIC or HQIC.

Parameter estimation
MLE, LS. Gradient-based optimizers for ETS-type models.

Linear Parametric Models

↑ High interpretability because of built-in temporal structure. No feature engineering required.

↑ Work with small data sets (< 10.000 effective samples).

↑ Representation as linear state space models allows for the use of Kalman filtering for hyperparameter selection (log-likelihood computation), parameter estimation (MLE computation), data imputation as well as online updating and smoothing after new data.

Linear Parametric Models

↑ Support statistical methodology like likelihood-based inference, confidence intervals, hypothesis tests.

↑ Inference computationally cheap (ARIMA(p, d, q) for example has inference cost around $\mathcal{O}(p + q)$ per time step). Results in very low latency. µs to single-digit ms on standard hardware.

Linear Parametric Models

↓ Capture only linear temporal patterns.

↓ Multivariate scenarios (apart from VAR) often cumbersome.

↓ Require stationarity or a preceding differencing procedure.

↓ Sensitive to missing values (imputation needed e.g. interpolation, forward fill, Kalman smoother) and outliers (removal or robust regression).

Linear Parametric Models

↓ Barely parallelizable (but see my talk on Deep SSMs).

↓ Require significant domain knowledge input.

↓ Inference : Poor Scalability for high-dimensional multivariate time series.

Linear Parametric Models

Applications in Finance

- ARIMA - Trading volumes (daily/weekly), commodity prices, interest rates (short-term yields), short-team macroec. indicators (quart. GDP, monthly unemployment, etc.)

- VAR - Macroec. indicators (GDP growth, inflation, interest rates), FX rates currency basket, equity index returns across countries

- ETS - Long-term macroec. indicators (consumption, inventory, GDP)

Linear Parametric Models

- Production grade libraries
  - Python : Nixtla.statsforecast, statsmodels, pmdarima (statsmodels ARIMA + Auto ARIMA functionality)
  - R : forecast

- Commercial providers
  - AWS Sagemaker AutoGluon
  - Azure ML AutoML
  - GCP Vertex AI AutoML

Nonlinear Parametric Models

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i \, \epsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \, \sigma_{t-j}^2$$

Mostly nonlinear autoregressive by nature. Capture complex dynamics like regime switching, volatility, smooth transitions. Common examples are GARCH, (S)TAR, NAR, Markov Switching, Damped Trend ETS.

Hyperparameter selection
Optimizing information criteria, usually AIC, AICc, BIC or HQIC.

Parameter estimation
(Q)MLE , CLS. MCMC, EM and SMC for more general nonlinear SSMs.

Nonlinear Parametric Models

↑ High interpretability because of built-in temporal structure. No feature engineering required.

↑ Work with small data sets.

↑ Support statistical methodology like likelihood-based inference, confidence intervals, hypothesis tests.

↑ Inference computationally cheap (usually around $\mathcal{O}(p \times k)$ per time step, k the number of nonlinear components). Results in low latency. ms range on standard hardware.

Nonlinear Parametric Models

↓ Multivariate scenarios are cumbersome.

↓ Require stationarity or a preceding differencing procedure.

↓ Sensitive to missing values (imputation needed e.g. interpolation, forward fill, Kalman smoother) and outliers (removal or robust regression).

↓ Barely parallelizable.

↓ Require significant domain knowledge input.

Nonlinear Parametric Models

Applications in Finance

- GARCH - High frequency asset return data (high kurtosis of ϵ_t , low autocorrelation and slow decay of $|\epsilon_t|^k$ and $\epsilon_t{}^{2k}$)
- (S)TAR - Interest and exchange rates, credit spreads (regime thresholds, different dynamics in low vs. high rate environments, volatile vs. calm markets)
- NAR - Trading volumes (heavy nonlinear dependencies on past values), implied volatilities (multiple maturities)

Nonlinear Parametric Models

Applications in Finance

- Markov Switching - Equity index levels (bull/bear phases), bond yields (shifts of monetary policy), commodities (supply disruptions)

- Damped Trend ETS - Long-term interest rates (historical trends, limits), corporate earnings/dividends (slowing growth), macroec. indicators (CPI, GDP, etc.)

Nonlinear Parametric Models

- Production grade libraries
  - Python : statsmodels, arch, Tensorflow Probability
  - R : rugarch, MSwM, tsDyn, pomp, KFAS

- Commercial providers
  - AWS Sagemaker : No built-in support. BYO with custom pipelines.
  - Azure ML : No built-in support. BYO with custom pipelines.
  - GCP Vertex AI : No built-in support. BYO with custom pipelines.

Gradient Boosting Machines

$$r_{i,m} = - \left[\frac{\partial \mathcal{L}(y_i, \hat{y}_{m-1}(x_i))}{\partial \hat{y}_{m-1}(x_i)} \right]$$

$$f_m(x) \longrightarrow r_{i,m} \quad (x \to x_i)$$

$$\eta_m = \arg\min_{\eta} \sum_i \mathcal{L}(y_i, \hat{y}_{m-1}(x_i) + \eta\, f_m(x_i))$$

$$\hat{y}_m(x) = \hat{y}_{m-1}(x) + \eta_m\, f_m(x)$$

Decision tree ensemble learning method. Weak learners are trained sequentially. Iterative error correction ('function space gradient descent') leads to bias reduction. Resulting model is an (implicitly) weighted sum of weak learners. (Friedman 1999, 2001) (Mason et al. 1999, 1999).

Hyperparameter selection

Bayesian optimization, Grid/random search + cross validation.

Gradient Boosting Machines

Configuration for time series learning

- Loss function : Regression (MSE/MAE/Huber) for forecasting, LogLoss/Softmax for classifying.

- Learning temporal order from feature engineering :
  Lag, rolling statistics (mean, standard deviation, moments), holidays, seasonal, etc.

Gradient Boosting Machines

↑ Capture short-term nonlinear and hierarchical dependencies.

↑ Multivariate scenario unproblematic.

↑ Missing values often dealt with natively,

↑ Work well with small and medium data sets (< 1.000.000 effective samples). Tend to saturate for larger data sets if more data does not result in more effective features.

↑ No precautions needed for nonstationarity, but models often profit from stationary or detrended features.

Gradient Boosting Machines

↑ Tree construction within single boosting iteration is effectively parallelizable, overall process by design is not.

↑ Inference computationally relatively cheap (around $\mathcal{O}(m \times d)$ for a prediction, m trees with depth d). Low to medium latency i.e. ms to 10s of ms range on standard hardware.

Gradient Boosting Machines

↓ Requires domain-informed feature engineering (lags, rolling averages, holidays, etc.),

↓ High inherent interpretability of single decision tree disappears in tree ensembles. Can be mitigated by feature importance, (Tree)SHAP.

↓ Notable sensitivity to outliers (mitigation through outlier removal and/ or usage of robust loss functions e.g. Huber, quantile loss).

Gradient Boosting Machines

Applications in Finance

- Asset price/return prediction (short-term, large feature sets with nonlinear relations, weak but consistent signals)
- Credit risk (complex cross section dependencies)
- Market microstructure/order book dynamics (large sets of derived features from noisy, high-dimensional data)

Gradient Boosting Machines

Applications in Finance

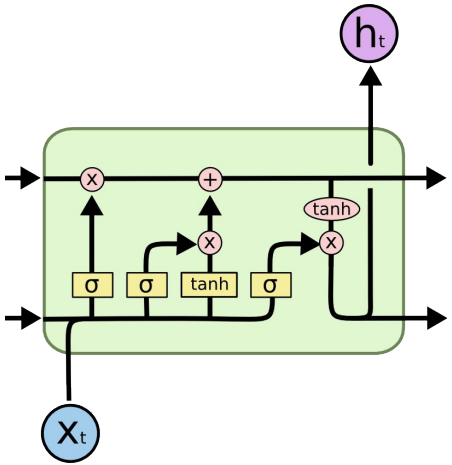
- Volatility forecasting (GARCH-derived features plus macroec., sentiment and order flow data)
- Macroec. prediction (high-dimensional sparse feature sets)

Gradient Boosting Machines

- Production grade libraries
  - Python : XGBoost, LightGBM (both can switch between Gradient Boosting and Bagging (Random Forest), Optuna, HyperOpt (Hyperparameter optimization)

- Commercial providers
  - AWS Sagemaker Autopilot : XGBoost, LightGBM with full custom tuning (also via AutoGluon)
  - Azure ML AutoML : XGBoost, LightGBM with full custom tuning
  - GCP Vertex AI AutoML Tabular : native Gradient Boosting with limited custom tuning; XGBoost, LightGBM via custom pipelines

Gated Recurrent Neural Networks



Olah 2015 'Understanding LSTM networks'

Gated recurrent neural networks (GRNNs) were invented to mitigate the notorious vanishing gradient problem of regular RNNs. The two most notable examples are LSTMs (long short-term memory networks) (Hochreiter and Schmidhuber 1997) and GRUs (gated recurrent units)(Cho et al. 2014).

Hyperparameter selection
Bayesian optimization, Grid/random search + cross validation.

Gated Recurrent Neural Networks

↑ Capture long-range and recurring (seasonal) temporal patterns.

↑ Multivariate scenario unproblematic (several approaches available).

↑ Little to none feature engineering required, minimal data preprocessing.

↑ Don’t expect stationarity.

↑ By definition sequential, can’t be parallelized across time. But profit significantly from (parallel) batch processing.

Gated Recurrent Neural Networks

↓ Hyperparameter tuning requires some effort.

↓ Low interpretability without additional techniques (integrated gradients).

↓ Sensitive to outliers and noise (mitigation through outlier removal) as well as to missing values (must be imputed).

↓ May fail to generalize for highly nonstationary data (long horizons).

↓ Larger datasets needed for proper generalization i.e. to avoid overfitting.
Larger means ~ 10^5 time steps to learn basic temporal patterns,
> 10^6 time steps for complex long-term dependencies.

Gated Recurrent Neural Networks

↓ Inference computationally relatively cheap (around $\mathcal{O}(h^2 + h \times d)$ per time-step for input dimension d and hidden dimension h). Results in medium latency, starting at 10s of ms on standard hardware.

Gated Recurrent Neural Networks

Applications in Finance

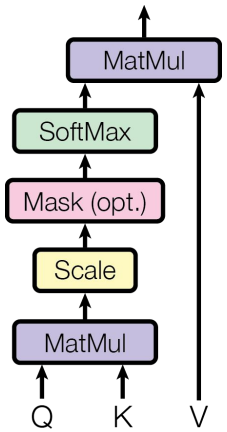
- Single asset short-term return (intra-/multi-day)
- Volatility (clustering, implied volatility)
- VaR/Expected Shortfall
- Intraday/Market Microstructure/Order book dynamics
- Macroec. indicators (GDP, CPI, unemployment, PMI, etc.)

Gated Recurrent Neural Networks

- Production grade libraries
  - Python : PyTorch nn.{LSTM, GRU} ,
    Tensorflow/Keras tf.keras.layers.{LSTM, GRU} ,
    NVIDIA cuDNN (includes C and C++ frontend)

- Commercial providers
  - AWS Sagemaker : DeepAR estimator, AutoGluon
  - Azure ML : native
  - GCP Vertex AI : native

Transformer Models



SDPA Vaswani et al. 2017

Sequential data processing neural networks with an underlying self-attention mechanism (specialized scaled dot-product attention (SDPA)) (Vaswani et al. 2017). Multi-head attention common. Encoder and/or decoder architecture.

Hyperparameter selection

Bayesian optimization, Population-based training, Grid/random search,

Transformer Models

↑ Capture long-range and recurring (seasonal) temporal patterns

↑ Handling of multivariate/multi-horizon time series unproblematic.

↑ Can learn high-dimensional data with complex correlated features.

↑ Massively parallelizable (all time steps in parallel).

↑ Handle multi-modal input signals from text, images or graphs.

↑ Zero-shot forecasting models don’t require retraining.

Transformer Models

- ↓ Hyperparameter selection usually a complex task. Proven defaults/ heuristics therefore play a big role. Underscores the black box character of this type of models.
- ↓ Large datasets needed. Small datasets result in overfitting. Problematic with proprietary, often small, datasets.
- ↓ Computationally expensive with high memory usage. Training places special demands on hardware (GPUs, high-bandwidth memory, etc.)
- ↓ No built-in temporal order. Needs to be entirely derived from positional encodings. (This is problematic with nonstationary temporal dynamics.)

Transformer Models

↓ Interpretability in theory given through attention mechanism, diminished through complexity of the models (multi-layer, multi-head). Techniques like Integrated Gradients and SHAP can improve the situation for time series transformers (see TimeGPT, Chronos).

↓ Sensitive to missing data. Larger amounts disrupt position encodings and can lead to sparse attention matrices (self-attention degeneracy). Mitigation through imputation, masking (use missing-value tokens), diffusion-based reconstruction.

Transformer Models

↓ High sensitivity to outliers and noise. Attention (softmax !) amplifies outliers, these get over-attended by the model. Distortion of embedding vectors. Common loss functions like MAE and MSE heavily affected by extreme values. Outliers can dominate gradient updates. Mitigation through smoothing and/or clipping.

↓ Inference computationally expensive (naive attention is $\mathcal{O}(L^2 \times d)$ with L the sequence length and d the embedding size). Depending on sequence length, medium to high latency. Scale very well for high-dimensional multivariate data. Transformer models designed for time series achieve better latency. Informer or Chronos (Bolt) (see next slides) reach $\mathcal{O}(L \times \log L)$ or even $\mathcal{O}(L)$.

Transformer Models

↓ AI laws in UK and Europe. Regulatory requirements may demand additional professional expertise.

Transformer Models

Transformer architectures adapted to time series

- Informer (Zhou et al. 2020) probabilistic sparse attention (reduce outlier sensitivity)
- Autoformer (Wu et al. 2021) (focus an trend and seasonality decomposition)
- ETSformer (Woo et al. 2022) exponential smoothing attention (reduce outlier sensitivity)

Transformer Models

Transformer architectures adapted to time series

- TimeGPT (Garza et al. 2023, Nixtla) foundation model (gen. transformer), pretrained on large amounts of time series data, zero-shot forecasting.

- Chronos (Ansari et al. 2024, AWS) foundation model (Google T5 -based autoregressive
decoding) , pretrained on large amounts of real and synthetic time series data, zero-shot forecasting.

- Chronos Bolt (2024, AWS) foundation model (Google T5 encoder-decoder) , pretrained on large amounts of real and synthetic time series data, zero-shot forecasting.

Transformer Models

Applications in Finance

- Mostly early stage/experimental.
- Known production deployments of TimeGPT and Chronos Bolt in anomaly detection and pattern-based volatility forecasting.

Transformer Models

- Production grade libraries
  - Python : Hugging Face Transformers,
    DeepSpeed (training and inference optimization),
    OpenAI Triton (optimized kernels),

- Commercial providers
  - AWS Sagemaker : Hugging Face DLCs, Chronos (Bolt) via AutoGluon
  - Azure ML : Azure Hugging Face SDK, Azure OpenAI
  - GCP Vertex AI : Model Garden, Custom Pipelines

Hybrid Models

Many production grade time series models are hybrids, built from two or more of the previous model types. Data with weak temporal patterns but many nonlinear relationships for example will often require a GBM step first to extract features. From these, sequences can be constructed which can be modelled by an LSTM.

Two important model classes are missing, CNNs and DNN Autoencoders. See my talk on order book dynamics.

Thank You for Your Attention !