DDBI Lecture 5: Advanced Methods

MLP, Gradient Boosting, Kalman, LSTM

Dr. Stavros K. Stavroglou

December 23, 2024

Outline

Lecture Objectives

In this lecture, you will:

- Explore **advanced forecasting methods** for **multivariate time series** using a climate dataset (AAO, AO, NAO, PNA, ENSO).
- Understand each method's **theoretical foundation**, as well as their strengths and weaknesses:
  - MLP (Multilayer Perceptron)
  - Gradient Boosting
  - Kalman Filter (State-Space)
  - LSTM (Long Short-Term Memory)
- Compare **simple out-of-sample forecasts** vs. **rolling forecasts**.
- Prepare for the **coding session** that will follow (using the provided Python script).

Dataset Overview

Dataset: `combined_climate_indices_2024.csv`

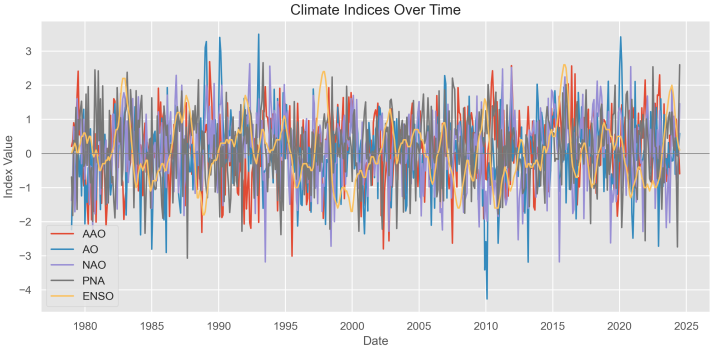
- Columns: `Date, AAO, AO, NAO, PNA, ENSO` (monthly from 1979+).
- We **forecast** the ENSO index using all 5 columns (AAO, AO, NAO, PNA, ENSO) as inputs.
- **Train/Test Split:** 80% train, 20% test (temporal split).

Important Considerations:

- Missing data handled by **linear interpolation**.
- **Possible scaling** for some models (e.g. neural nets).
- Evaluate both **simple out-of-sample** and **rolling** forecasts.

Dataset Visualization (Placeholder)

- Below is a placeholder for a time-series plot of the dataset.
- The Python script will generate a figure showing all indices over time.

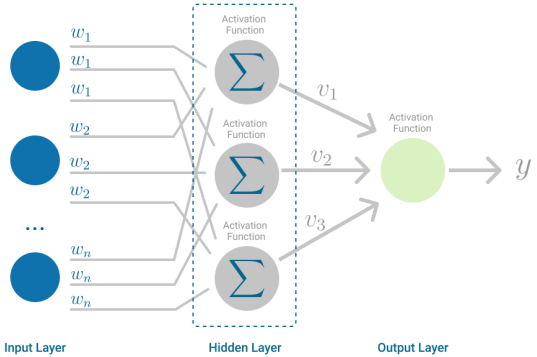


(Combined climate indices plot.)

MLP: Intuitive Explanation

- **Multiple layers of neurons** learn hierarchical features from input data.
- **Non-linear activations** help capture complex relationships.
- **Backpropagation** adjusts weights to minimize forecast error over iterations.

MLP Architecture



(Schematic of a feed-forward MLP with one hidden layer.)

MLP (Multilayer Perceptron)

Pros:

- Can capture **complex, non-linear** relationships.
- Flexible architecture (layers, neurons, activation).

Cons:

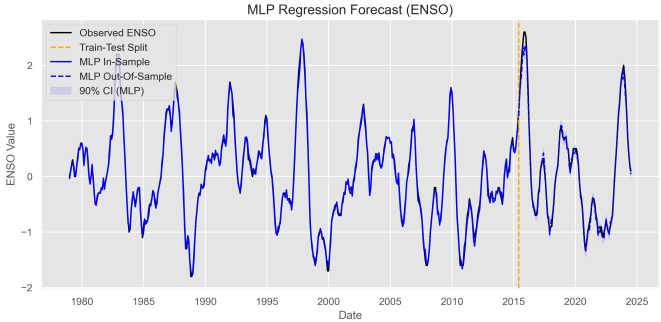
- **Prone to overfitting** without regularization/early stopping.
- Requires more **data** and careful hyperparameter tuning.

Tips:

- Often helps to **scale** data (e.g. StandardScaler).
- Use **validation set** or cross-validation for tuning layer sizes, learning rates.

MLP Forecast Example (Placeholder)

- Plot of actual ENSO (train + test) vs. MLP's fitted/forecasted values.
- A dashed orange line typically shows the train-test split.
- **Confidence intervals** or residual-based bands may appear as shaded areas.

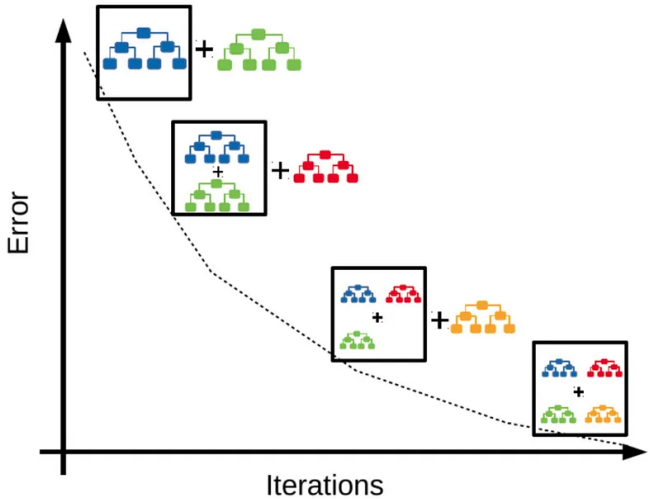


(MLP forecast visualization.)

Gradient Boosting: Intuitive Explanation

- **Starts with a simple model** (often a small tree) and fits to the data.
- **Sequentially adds new trees** that correct residual errors from previous steps.
- **Learning rate** scales the contribution of each new tree for controlled updates.

Gradient Boosting Diagram



(Ensemble of decision trees added iteratively.)

Gradient Boosting Basics

Pros:

- Often high accuracy "out of the box."
- Can model non-linearities; relatively robust to outliers.

Cons:

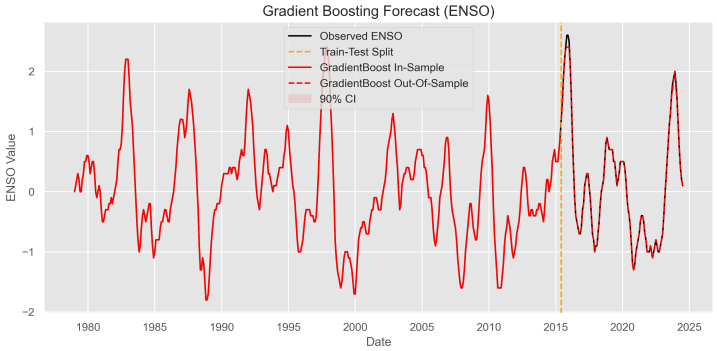
- Many **hyperparameters** (learning rate, n_estimators, max_depth, etc.).
- Can **overfit** if not tuned properly.

$$F_m(x) = F_{m-1}(x) + \nu \cdot h_m(x),$$

where h_m fits the current residuals, and ν is the **learning rate**.

Gradient Boosting Forecast (Placeholder)

- Placeholder for a plot showing the GB predicted vs. actual ENSO.
- Expect a **smooth** forecast that refines errors each iteration.

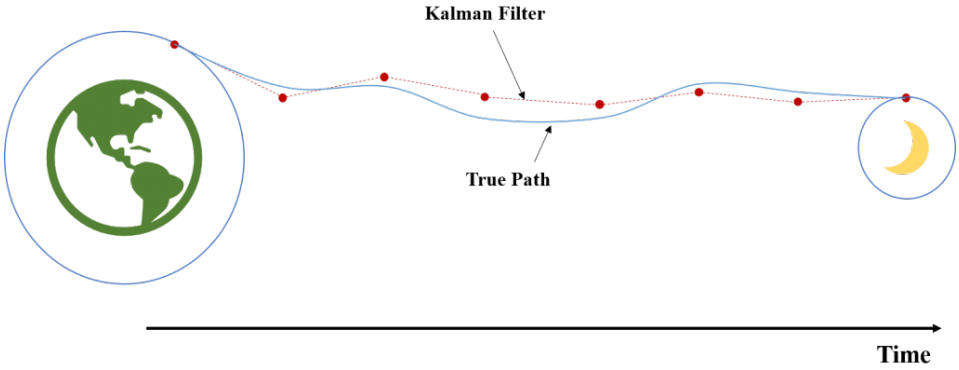


(Gradient Boosting forecast visualization.)

Kalman Filter: Intuitive Explanation

- **Views time series as hidden states** evolving with some dynamics.
- **Updates beliefs** about these states using noisy observations at each step.
- **Combines prior predictions and new data** to get a refined state estimate.

Kalman Filter Illustration



(A conceptual depiction of sequential updating, e.g. Earth-to-Moon orbit.)

Kalman Filter: State-Space Modeling

Pros:

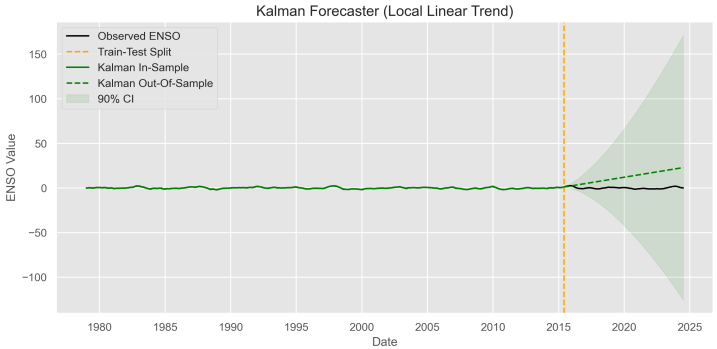
- Handles noisy time series well (blends prior + new data).
- Explains system with interpretable **state parameters**.

Cons:

- Assumes **linear-Gaussian** transitions (for basic version).
- Need to carefully define the **state-space structure**.

Kalman Filter Forecast (Placeholder)

- A plot of Kalman in-sample fit + out-of-sample forecast for ENSO.
- The orange vertical line: train-test split.
- Possibly 90% CI from the model's state-space conf. intervals.

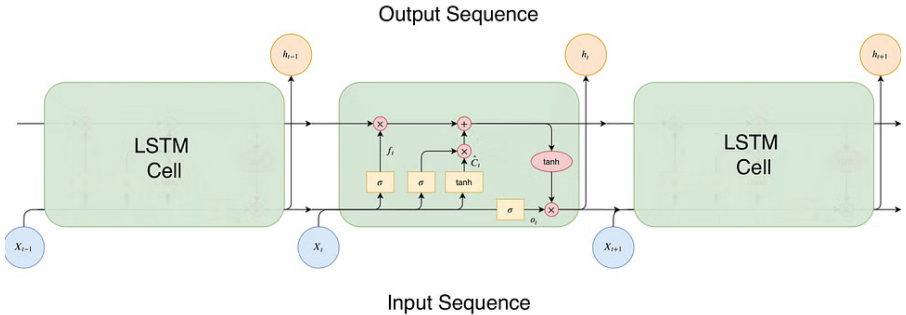


(Kalman filter forecast visualization.)

LSTM: Intuitive Explanation

- **An RNN variant** designed to mitigate vanishing/exploding gradients.
- **Uses gating mechanisms** (input, forget, output) to control memory flow.
- **Maintains long-term dependencies** useful for sequential/temporal data.

LSTM Cell Diagram



(LSTM cell with input, forget, and output gates.)

LSTM (Long Short-Term Memory)

Pros:

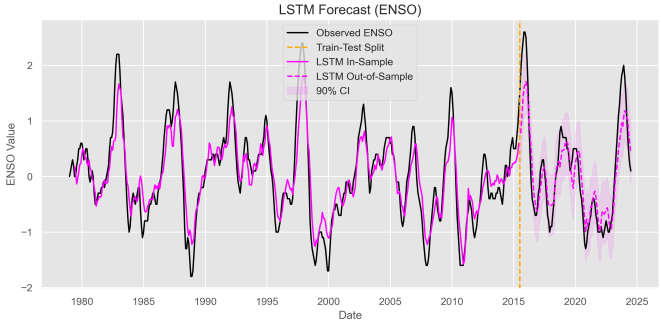
- Great for sequential data (language, finance, climate).
- Can **capture complex temporal** patterns better than simple MLP.

Cons:

- Needs **more data** and computing resources.
- Hyperparameter tuning (sequence length, units, dropout, etc.) can be intensive.

LSTM Forecast (Placeholder)

- A sequence-based model: **sliding window** or **sequence length** approach.
- Plot typically starts later (since the first seq_length points are used for initialization).
- 90% CI from residual bootstrapping or advanced methods.



(Placeholder: LSTM forecast visualization.)

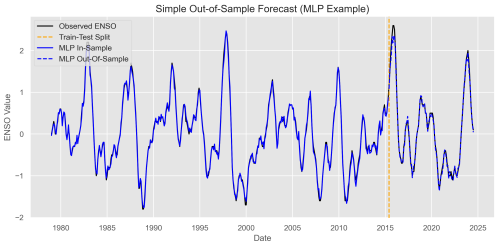
Forecasting: Simple vs. Rolling

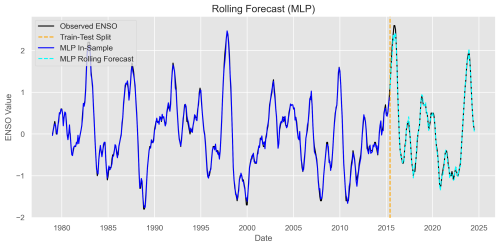
Simple Out-of-Sample:

- Train on all training data, predict the entire test set "at once."
- **Fast**, but might be **less realistic** for real-time updates.

Rolling Forecast:

- Predict 1 step (or a short horizon), then **update** the model with the new actual.
- **More realistic** but **computationally heavier**.





Method Pros & Cons Recap (1/2)

MLP

- **Pros**: Flexible, handles non-linearities well.
- **Cons**: Overfitting possible, might need lots of data/tuning.

Gradient Boosting

- **Pros**: Often top performance, can capture complex patterns.
- **Cons**: Many hyperparameters, can overfit if not tuned carefully.

Method Pros & Cons Recap (2/2)

Kalman Filter (Local Linear Trend / State-Space)

- **Pros**: Good for noise, interpretable states, easy real-time updating.
- **Cons**: Typically linear-Gaussian assumption, need domain knowledge to set states.

LSTM

- **Pros**: Great for long-range dependencies, good at capturing non-linear temporal patterns.
- **Cons**: Data-hungry, heavy compute, hyperparameters can be extensive.

Recommendation: Start with simpler models, then escalate to more complex if needed.

Coding Session Preview

We will:

- Load the `combined_climate_indices_2024.csv` dataset in Python.
- Implement each method (MLP, Gradient Boost, Kalman, LSTM) to forecast ENSO.
- Compare performance on **simple out-of-sample** forecasts.
- Attempt a **rolling** approach for some models to see real-time updates.

Key Steps in Script:

1. Data prep, interpolation, train/test split.
2. Model fitting + forecast + plotting with confidence intervals.
3. Side-by-side performance comparison (e.g. with MSE, MAE, etc. if desired).

Hands-On Considerations

- **Feature Engineering:**
  - Possible lags of ENSO or rolling means.
  - Consider **scaling** for MLP, LSTM.
- **Hyperparameter Tuning**:
  - MLP: hidden layers, neurons, learning rate.
  - Gradient Boost: n_estimators, max_depth, learning rate.
  - LSTM: sequence length, hidden units, epochs, batch size, dropout.
- **Validation Approach**:
  - Time series split, rolling windows.
  - Evaluate both simple and rolling scenarios for a broader view.

Key Takeaways

- **Advanced methods** each bring distinct assumptions, pros, and cons.
- Evaluate performance with **multiple scenarios** (simple vs. rolling).
- **Hyperparameter tuning** is critical—no single setting fits all data.
- Domain knowledge often helps in **feature engineering** and picking suitable methods.

Next Steps & Lab

- **Lab Session**:
  - Implement each model in Python (`scikit-learn`, `statsmodels`, `tensorflow`).
  - Compare forecasts on the climate dataset.
  - Experiment with small hyperparameter changes to see the effect.
- **Further Explorations**:
  - Incorporate domain knowledge about climate cycles.
  - Explore more advanced neural network architectures or **Transformers** for time series.
  - Delve into robust **state-space** or **Gaussian Process** regressions.

Get ready to experiment and code!

Thank you! Questions?