Time Series Modeling using Neural Networks

Dušan Fedorčák

11/2021 – RBI

Machine Learning College

Background

- Ph.D. in computer science at VŠB-TU Ostrava
  - Neural networks & unsupervised self-organization
- Experienced in simulations
  - flood prediction system for MSK
  - traffic monitoring & prediction systems
- Experienced in computer graphics & scientific visualization
  - GIS related real-time 3D visualizations
- 5+ years in applied ML and artificial intelligence
  - Lead researcher in GoodAI – general artificial intelligence
  - CTO in Neuron Soundware – sound processing via Deep Learning
  - Lead ML in Merlon Intelligence Inc. – natural language processing

Content

DAY 1

- **Classical time series analysis**
  - *Decomposition of time series*
  - *ARIMA models family*
  - *State space models generalization*
- **Theoretical window**
  - *Neural Networks & Recurrent NNs*
  - *Time series specifics*
- **Practical examples**
  - *Simple regression – toy example*
  - *Rainfall-runoff simulation – regression*

– lunch break –

- **Practical examples**
  - *Trampoline jumping – classification*
  - *Local Weather Forecast – regression*

DAY 2

- **Product Design & ML**
  - *Integration of ML models into products*
  - *Tips & tricks for debuging NNs*
- **Practical Examples** (in random order)
  - *Exoplanets Hunting*
  - *Mobile Motion Sensing*

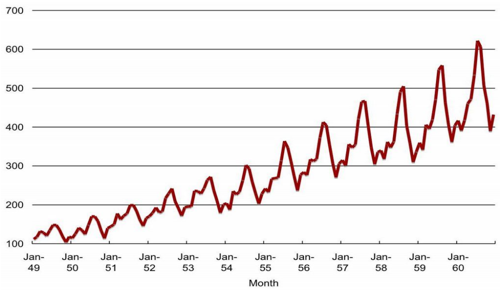
– lunch break –

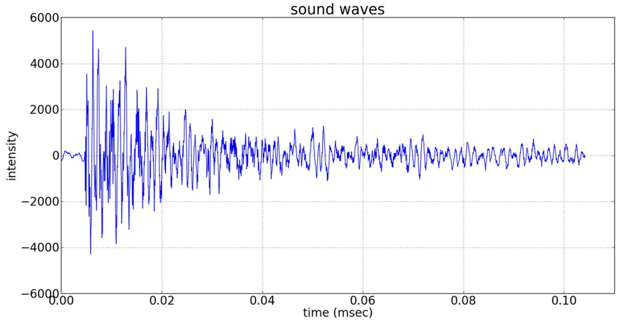
  - *Manufacturing Process Modeling*
  - *Financial distress prediction*

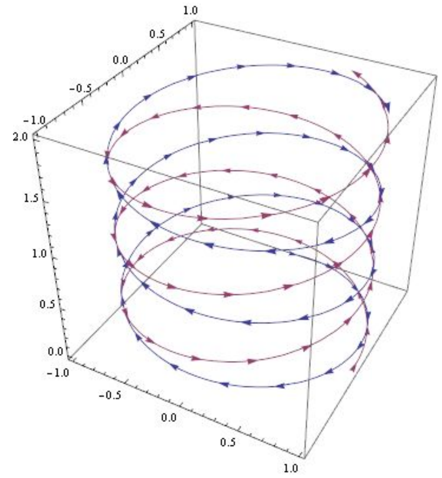
- *Google Drive Folder with data*
- *GitHub repository with example sources*

Machine Learning College

Time series – example data

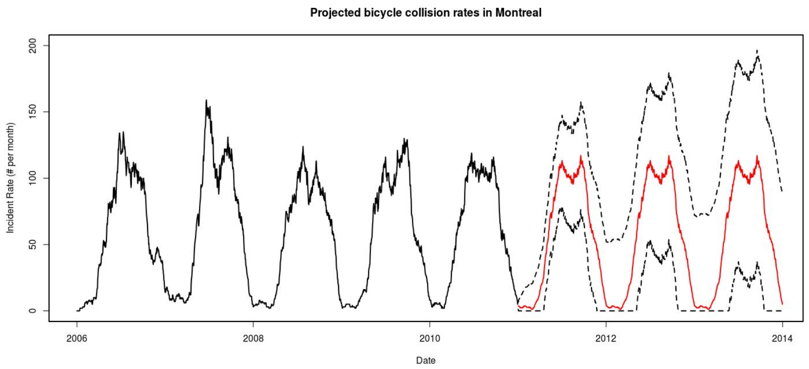


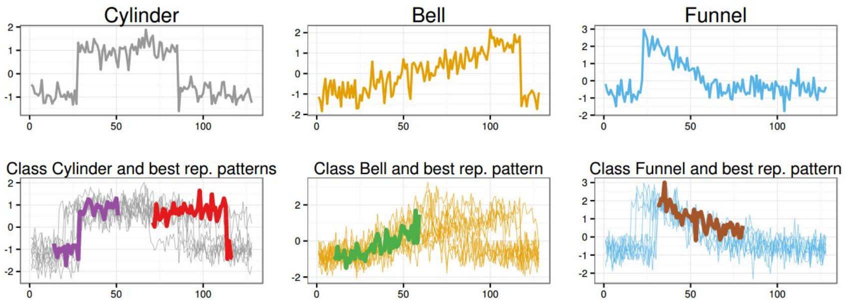


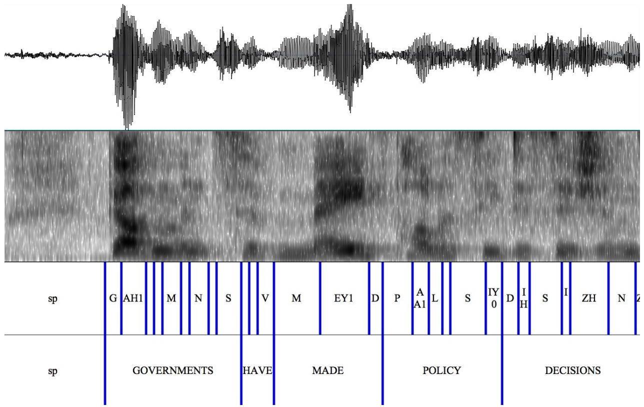


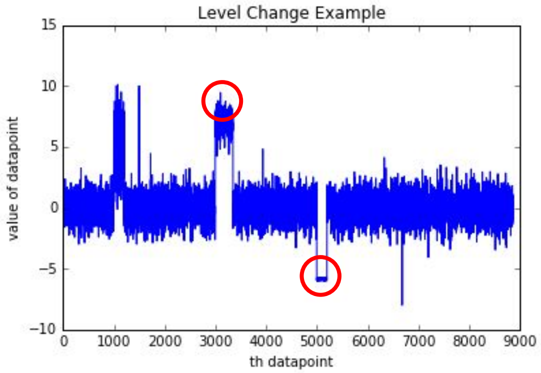
Machine Learning College

Time series – example tasks





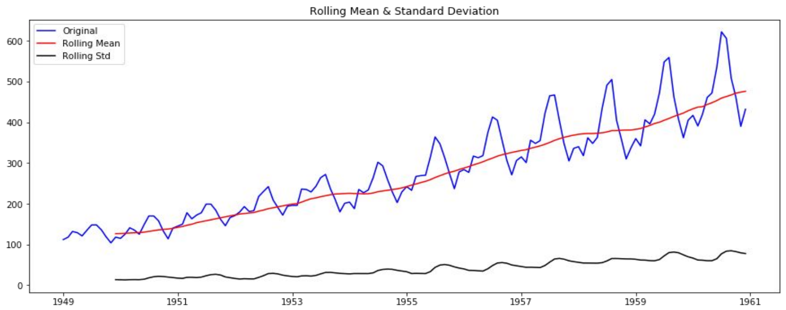


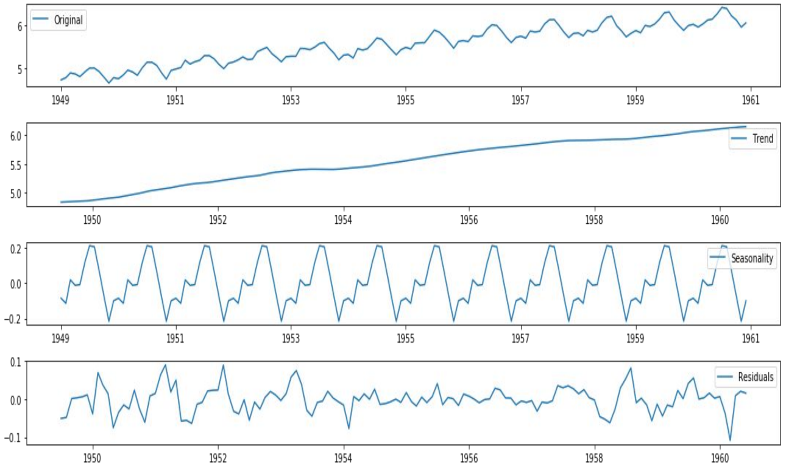


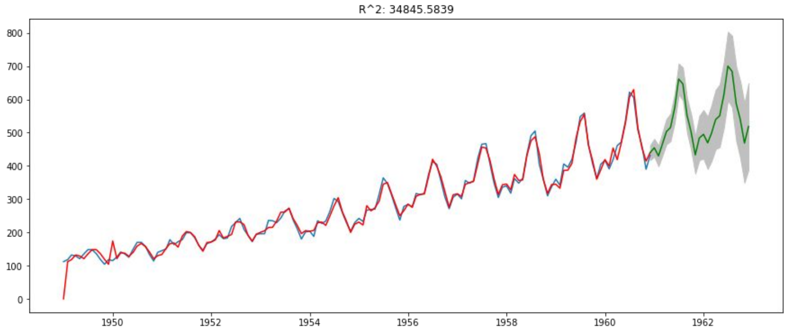
Machine Learning College

Time Series – classical analysis & modeling

- Time Series Decomposition
  - Inflation, trend, seasonality, differencing
- ARIMA models
  - http://people.duke.edu/~rnau/411home.htm







Machine
Learning
College

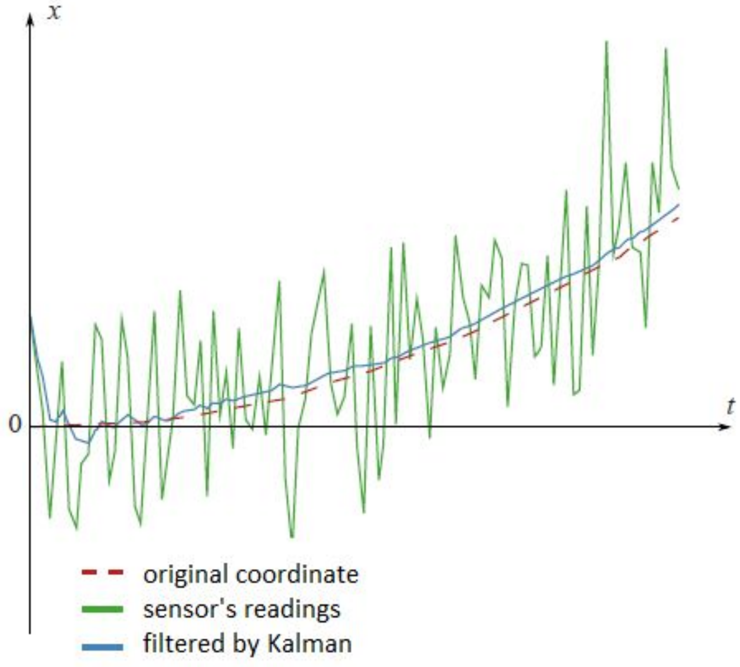
State Space Models

- State Space Models
  - A dynamic system that **evolves over time**
  - Knowing **the current state of the model is enough** to predict the future
  - The true state of the system might **not** be **directly observable**
- Model Description
  - State
    - $\boldsymbol{x}_t \sim N(\boldsymbol{x}_t, \boldsymbol{P}_t)$
  - State Equation
    - $\boldsymbol{x}_t = \boldsymbol{F}\boldsymbol{x}_{t-1} + N(0, \boldsymbol{Q})$ – *sometimes without noise*
  - Observation Equation
    - $\boldsymbol{y}_t = \boldsymbol{H}\boldsymbol{x}_t + N(0, \boldsymbol{R})$

Machine Learning College

Kalman Filter

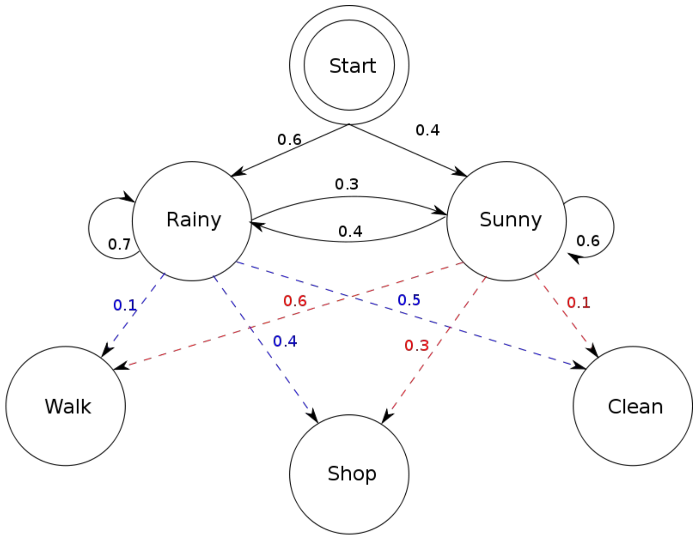
- Evolve state
  - $x'_t = Fx_{t-1}$
  - $P'_t = F\,P_{t-1}F^T + Q$
- Integrate observation
  - $x_t = x'_t + K_t(y_t - Hx'_t)$
  - $P_t = (I - K_tH)\,P'_t$
- **Kalman Gain**
  - $K_t = P'_t\,H^T(HP'_tH^T + R)^{-1}$
- ARIMA and Kalman Filter
  - ARIMA can be viewed as a state space model
  - ARIMA can be fitted with MLE via Kalman Filter
  - https://bookdown.org/rdpeng/timeseriesbook/maximum-likelihood-with-the-kalman-filter.html
  - https://towardsdatascience.com/the-kalman-filter-and-maximum-likelihood-9861666f6742



Machine Learning College

Hidden Markov Model

- Model Description
  - HMM ($\boldsymbol{\lambda}$) can be viewed as a **state space model**
  - Finite set of hidden states
    - $\mathbf{Q} = \{q_1, q_2, \dots, q_n\}$, $\boldsymbol{\pi} = \{\pi_1, \pi_2, \dots, \pi_n\}$ – init
    - $\boldsymbol{n}$ – number of states (hyperparameter)
  - Set of observations
    - $\mathbf{O}_i = (o^1, o^2, o^3, \dots, o^T)$
  - Transition probability matrix & emissions
    - $\boldsymbol{A} = (a_{00}, \dots, a_{nn})$, $\mathbf{B} = q_i \to o$

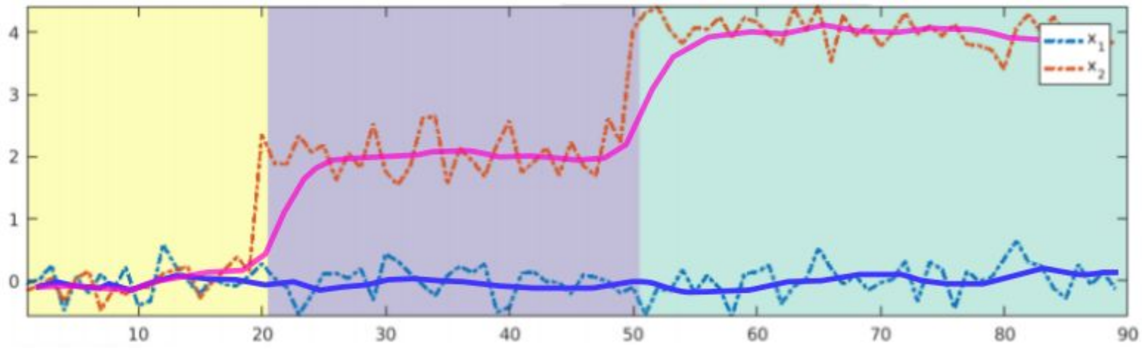


- Model Capabilities
  - $P(\mathbf{O}|\boldsymbol{\lambda})$ – Give prob. of **O** being produced by $\boldsymbol{\lambda}$ – *forward-backward alg.*
  - $P(\mathbf{q}_1, \dots, \mathbf{q}_t|\mathbf{O}, \boldsymbol{\lambda})$ – Give most likely sequence of states for given **O** – *Viterbi alg.*
  - $\mathbf{O} \Rightarrow \boldsymbol{\lambda}$ – Model must be trainable with **O** – *Baum-Welch alg.*

Machine Learning College

Kalman Filter vs. Hidden Markov Model

- **Kalman Filter**
  - Continuous state
  - Generic state & observation equation
  - Linear dynamic system
  - *Fusion of sensor readings and controls*
  - *ARMA models implementation*
- **Hidden Markov Model**
  - Discrete set of states
  - N-states hyperparameter
  - Emission & Transition tables
  - *Speech recognition*
  - *Time series segmentation*



http://lasa.epfl.ch/teaching/lectures/ML_MSc_Advanced/Slides/Lec_XI_HMM_extensions.pdf

Machine Learning College

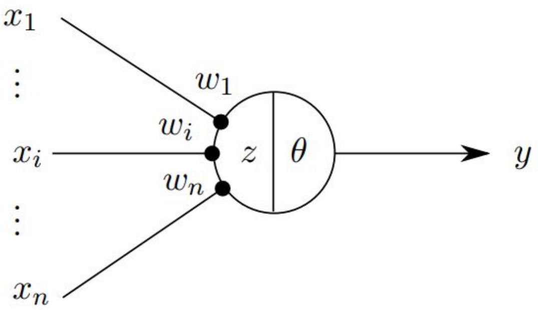
Time Series – goal in classical terminology

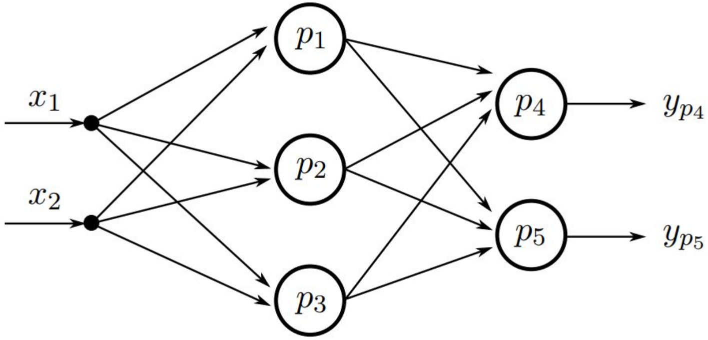
- **Forecasting**
  - Given the past and the present observation, what will the future look like?
- **Time scale analysis**
  - Given the observations, what time scales dominate when observing temporal variation in the data
- **Filtering**
  - Given the past and the present observation, how should I update my estimate of the true state of nature?
- **Smoothing**
  - Given a complete dataset, what can I infer about the true state of nature in the past?
- **Regression**
  - Given a time series of two phenomena, what is the association between them?

https://bookdown.org/rdpeng/timeseriesbook/

Machine Learning College

Neural networks

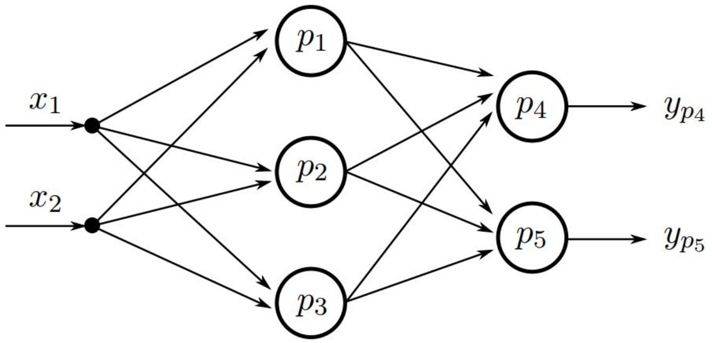


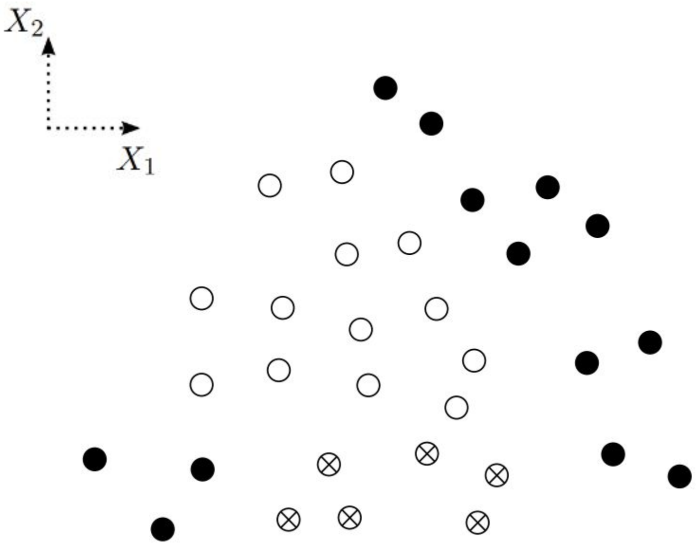


- Artificial Neural Cells
  - Linear combination of inputs
  - Non-linear activation function
- Connected Neurons
  - Directed graph
  - Layered structure
    - Dense connections
    - Convolutions & pooling
    - Recurrency, signal gates
    - Masking & attention heads
- Universal function approximator
  - Trainable with data
  - Backpropagation
  - Deep vs. shallow architecture

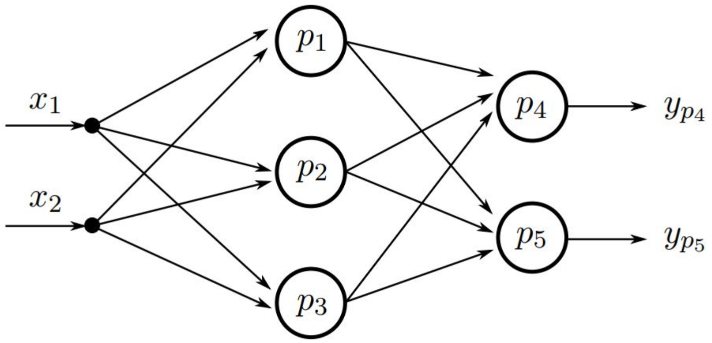
Machine Learning College

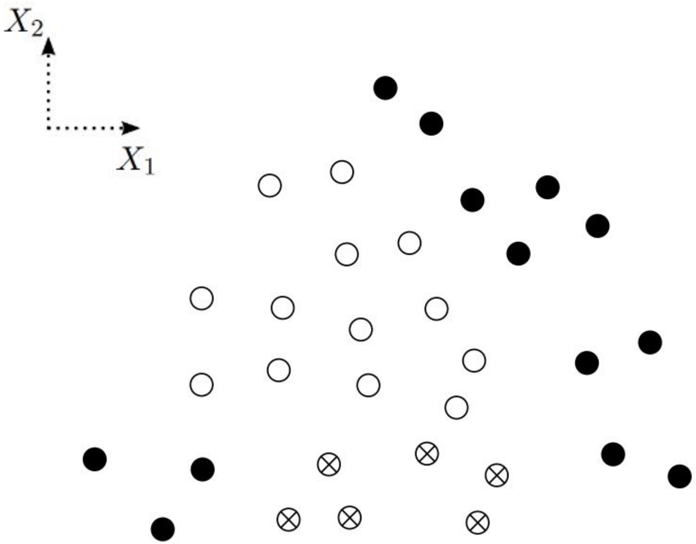
Neural networks – Input space separation





Neural networks – Input space separation

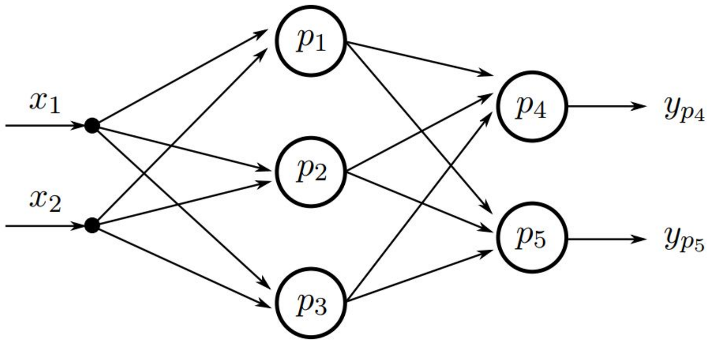


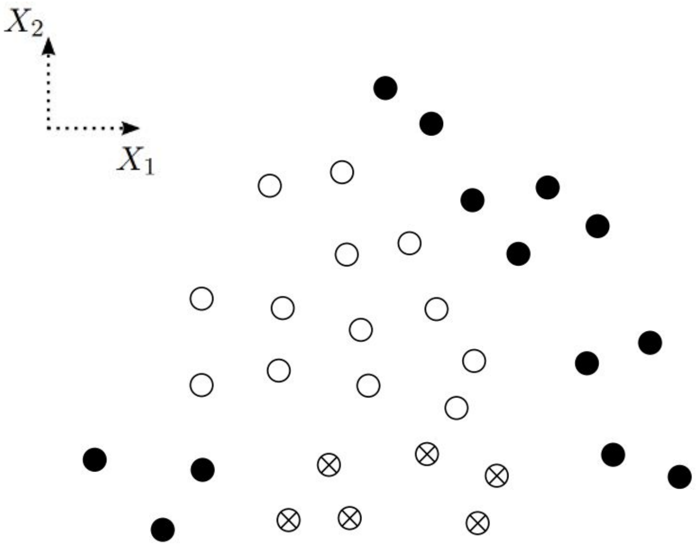


$$y = s(\Sigma\, w_i x_i - \theta)$$

Machine Learning College

Neural networks – Input space separation

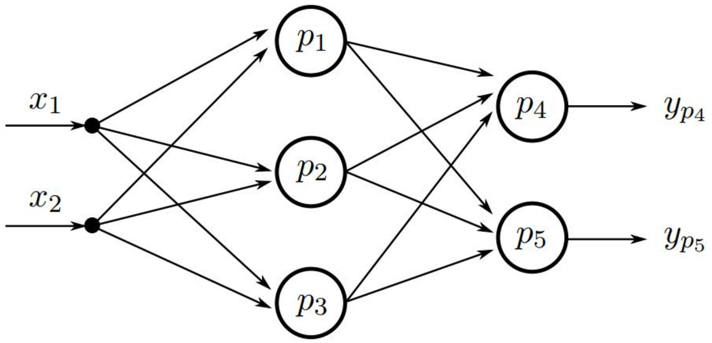


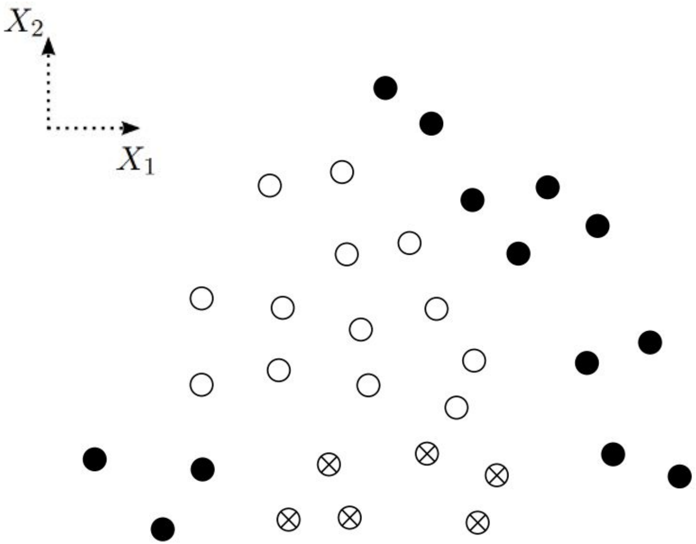


$$y = s(\Sigma\, w_i x_i - \theta) = s(w_1 x_1 + w_2 x_2 - \theta)$$

Machine Learning College

Neural networks – Input space separation



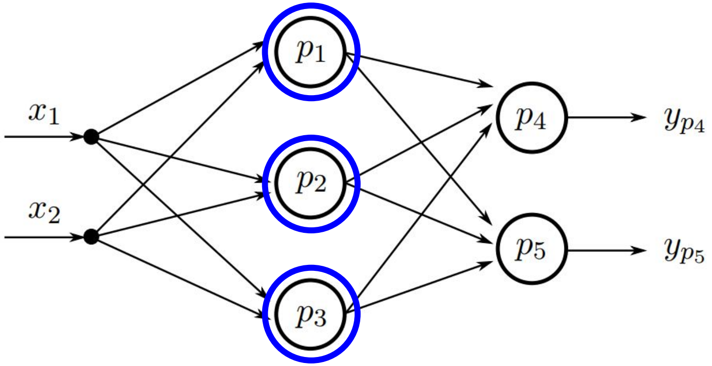


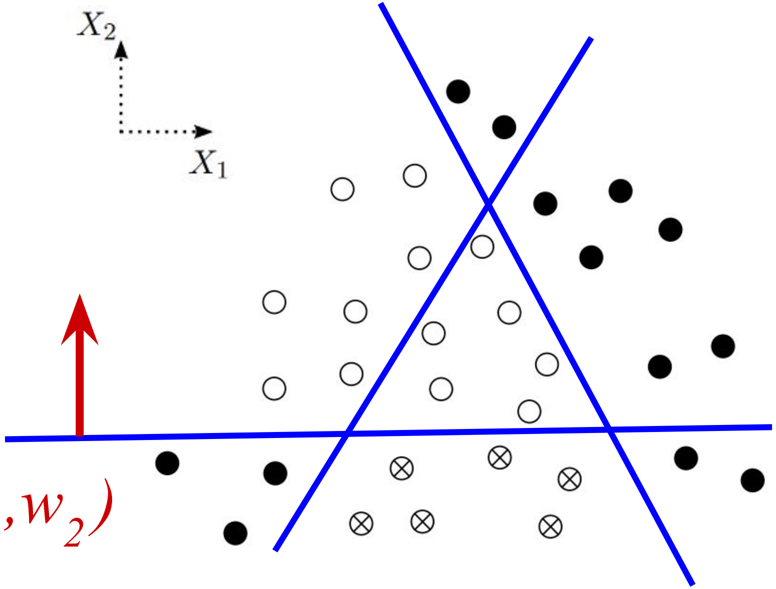
$$ax + by + c = 0$$

$$y = s(\Sigma w_i x_i - \theta) = s(w_1 x_1 + w_2 x_2 - \theta)$$

Machine Learning College

Neural networks – Input space separation



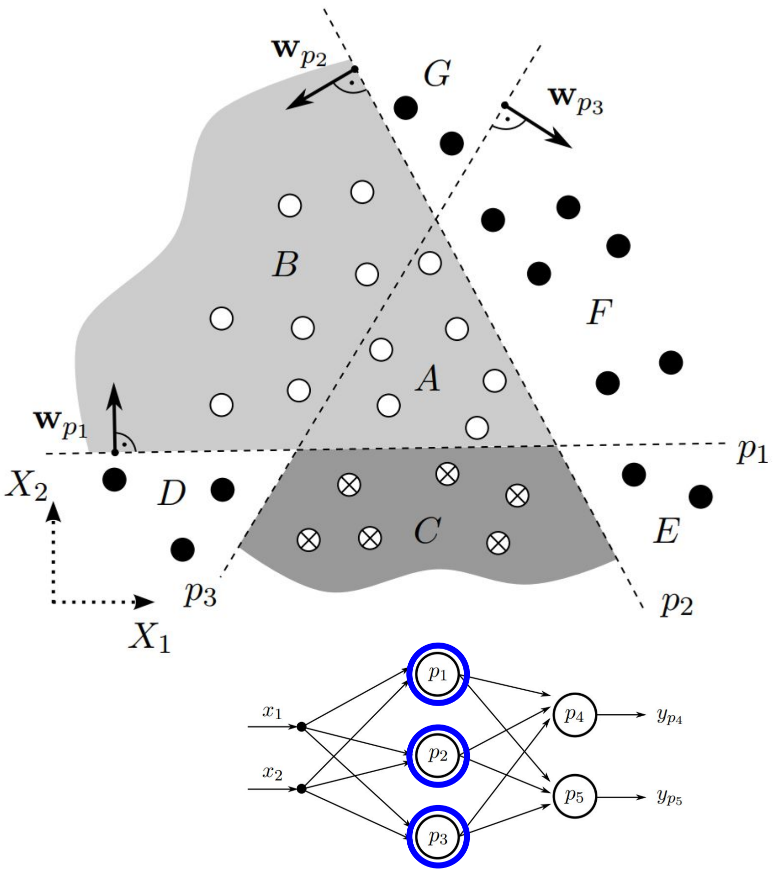


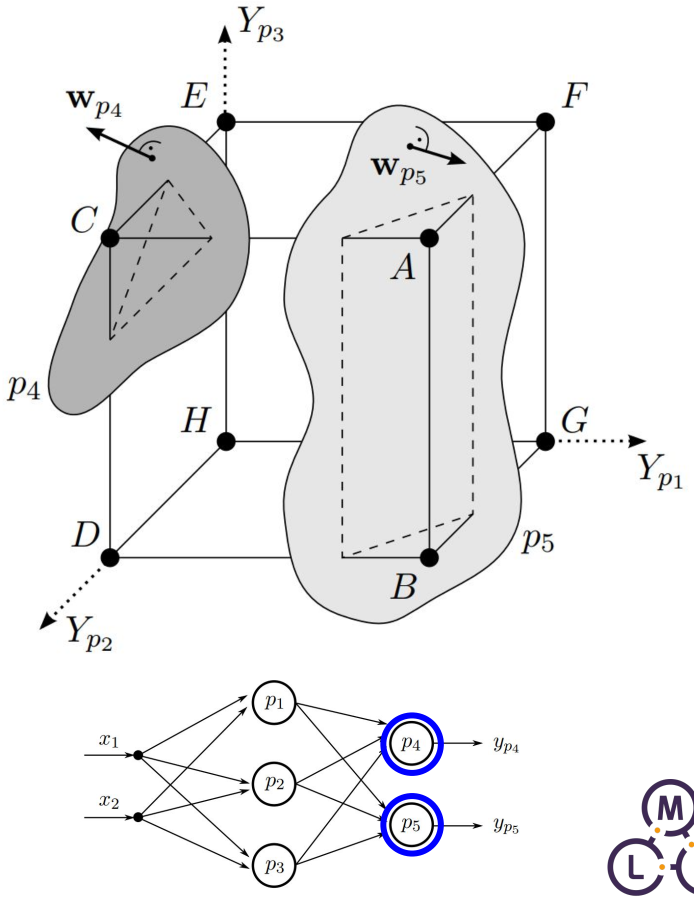
$$(a,b) = (w_1, w_2)$$

$$ax + by + c = 0$$

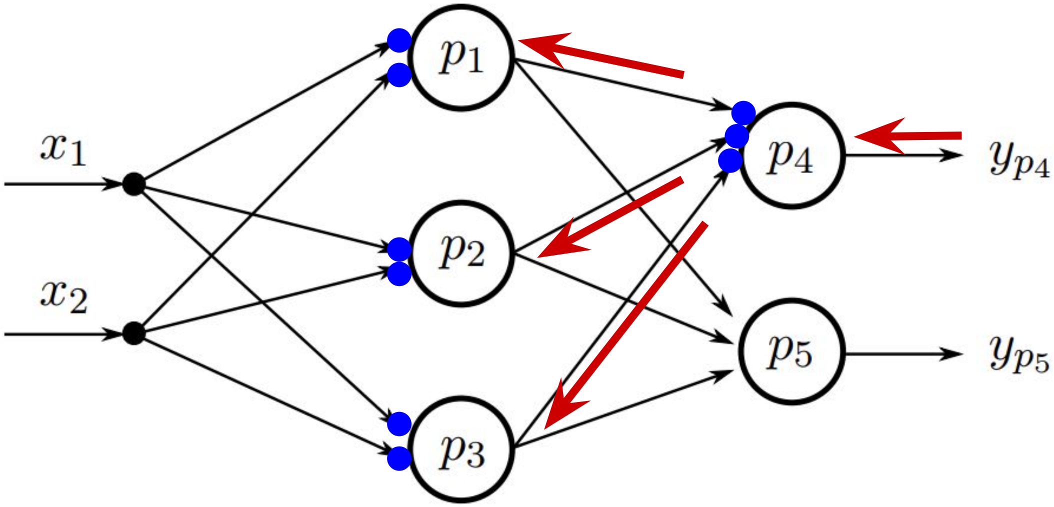
$$y = s(\Sigma\, w_i x_i - \theta) = s(w_1 x_1 + w_2 x_2 - \theta)$$

Neural networks – Input space separation

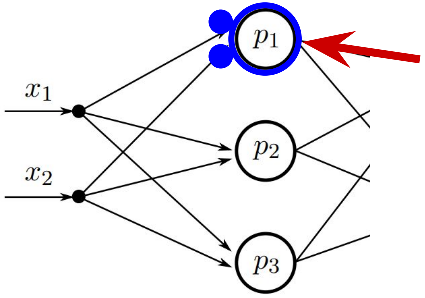


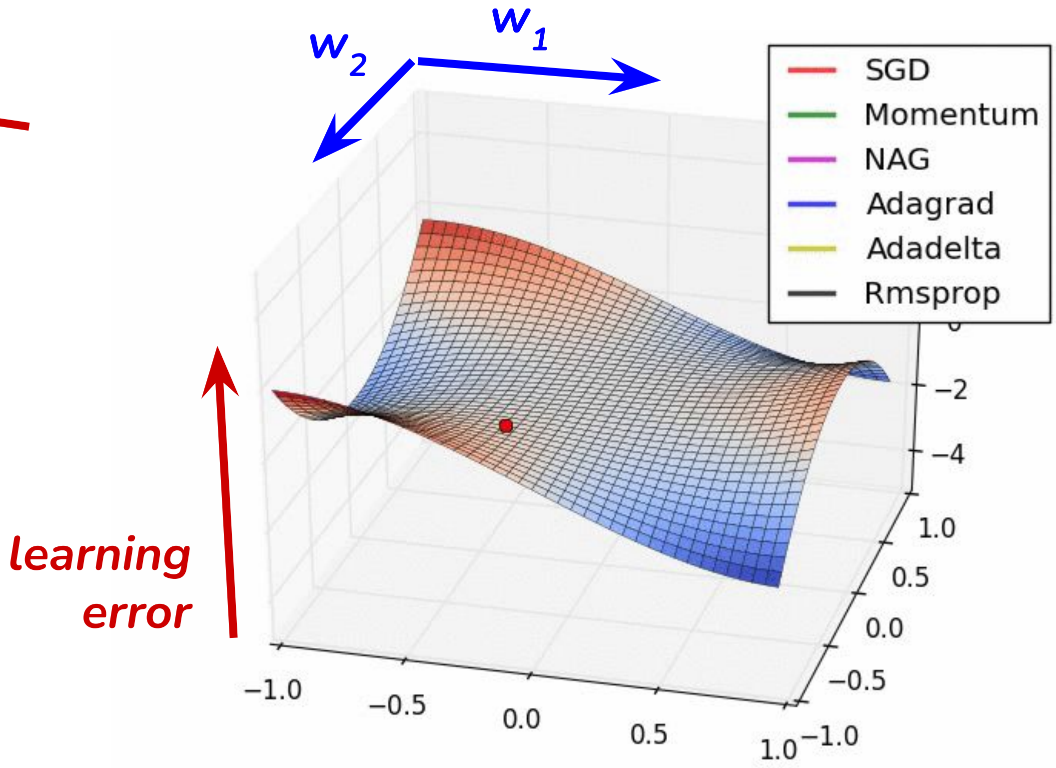


Neural networks – Backpropagation



Neural networks - Backpropagation

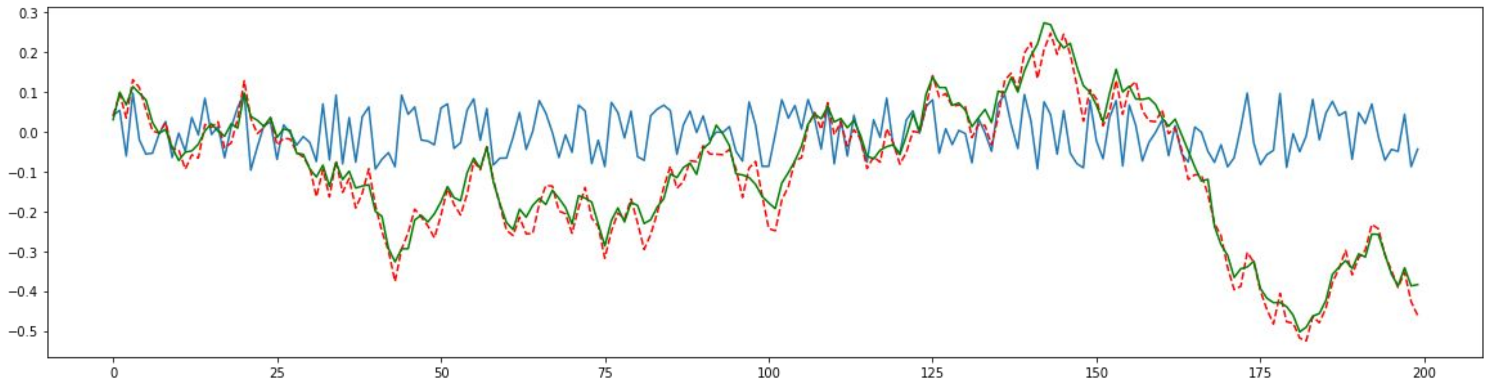




http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html

Machine Learning College

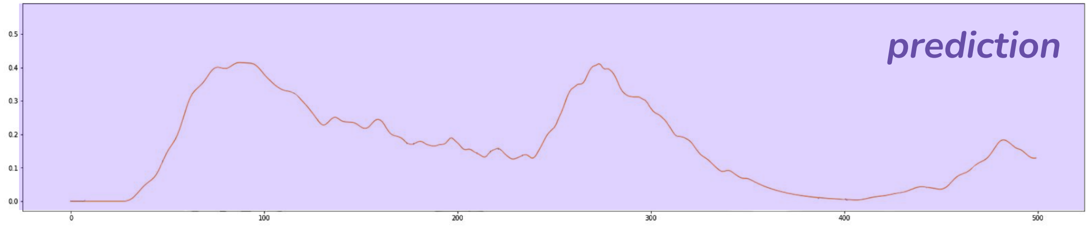
Time Series with Neural Networks

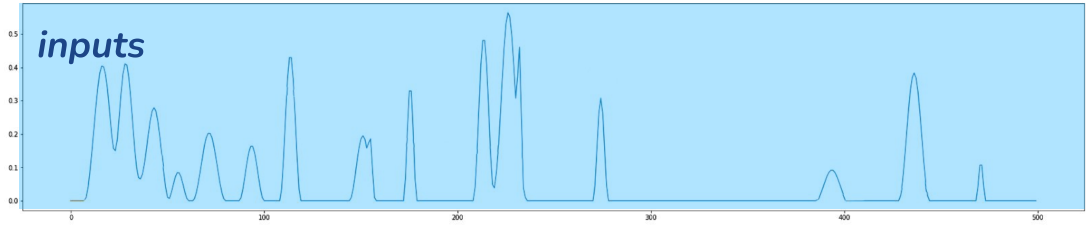


- Neural Networks
  - How to **express time domain**
  - How to **prepare training data**
  - How to **design the model**
  - How to **train & test the model**

Machine Learning College

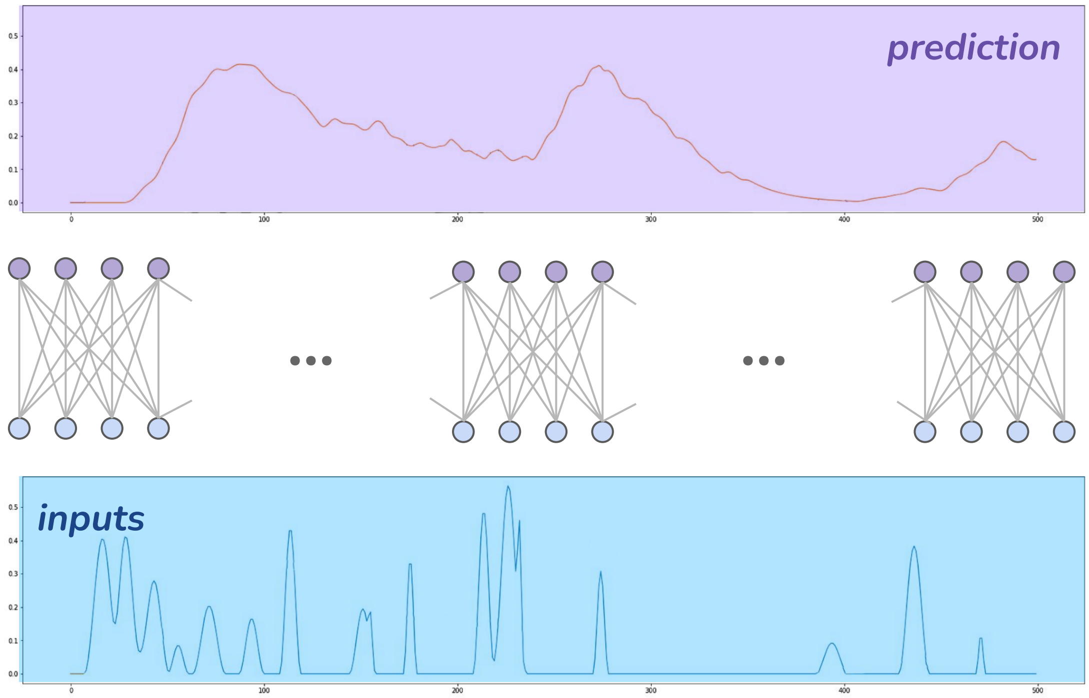
How neural network fits?



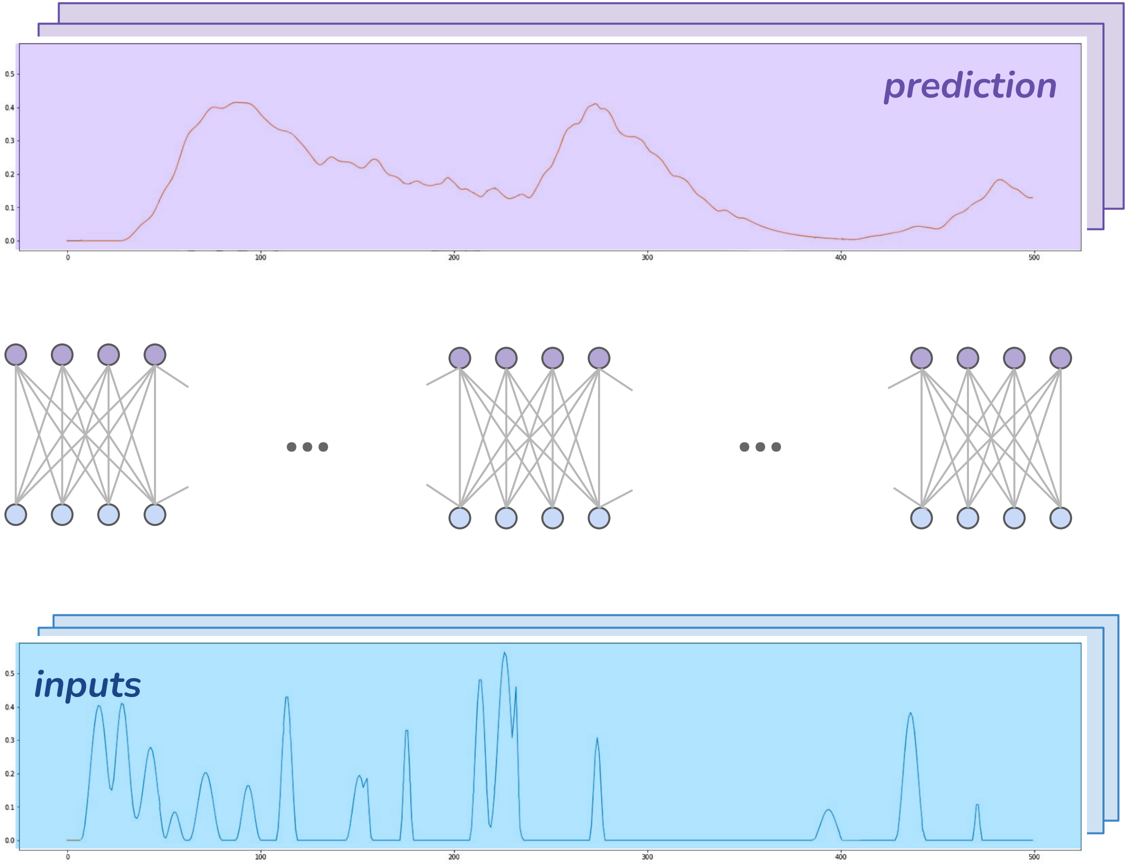


Machine Learning College

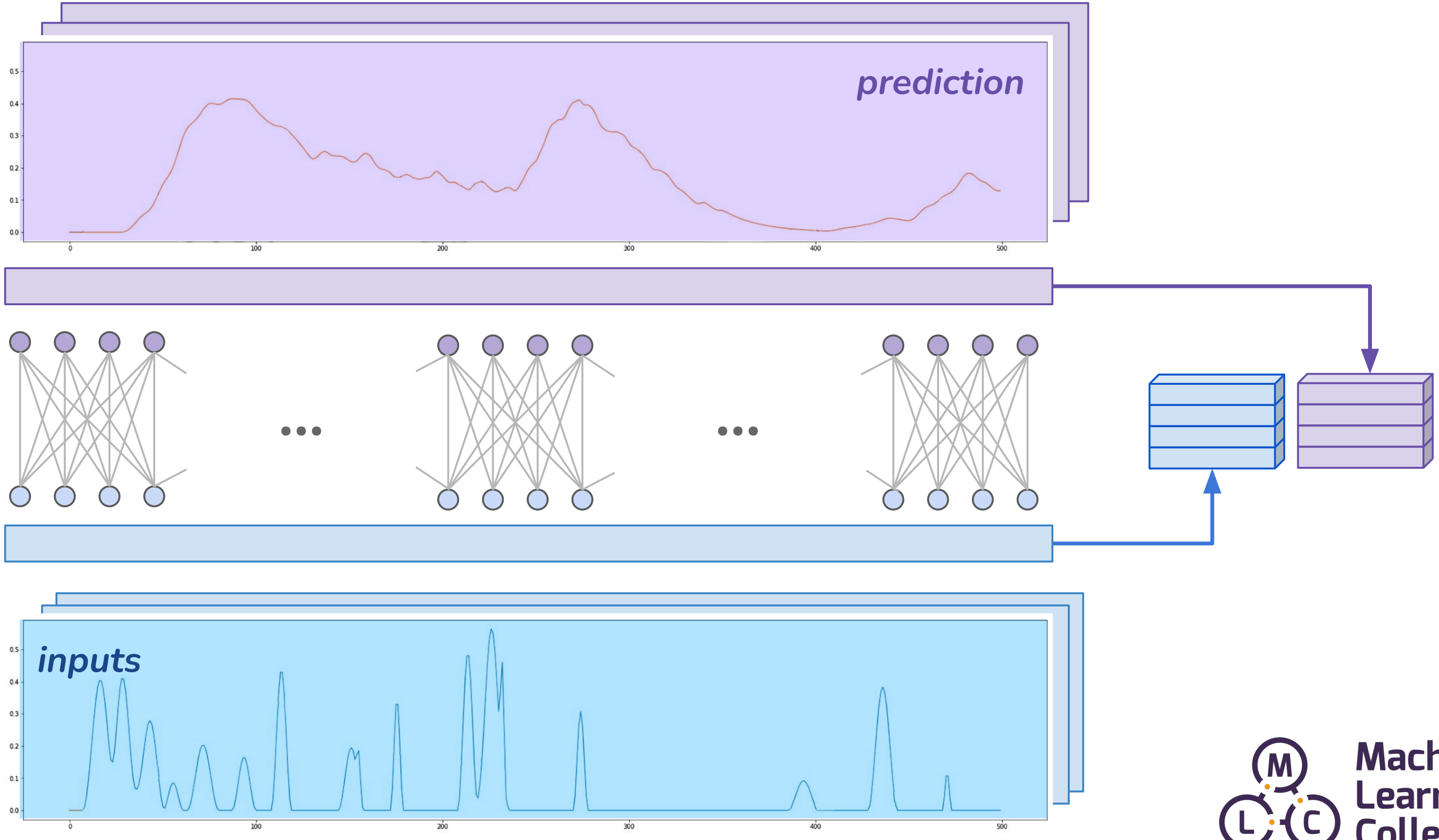
Simple feed-forward network



Simple feed-forward network

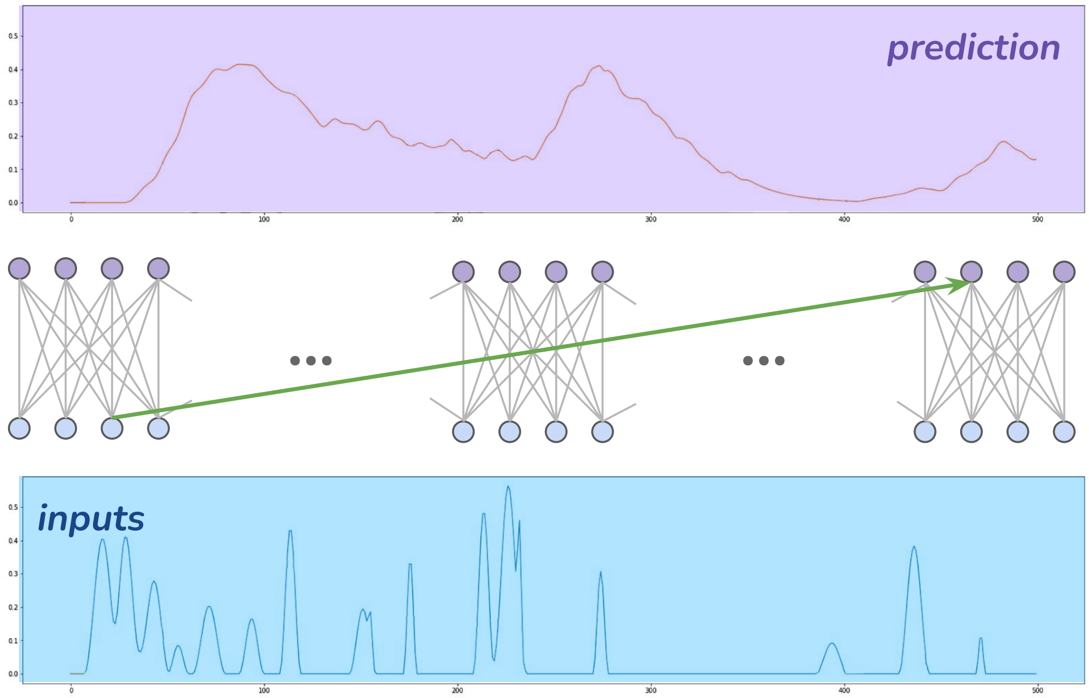


Simple feed-forward network

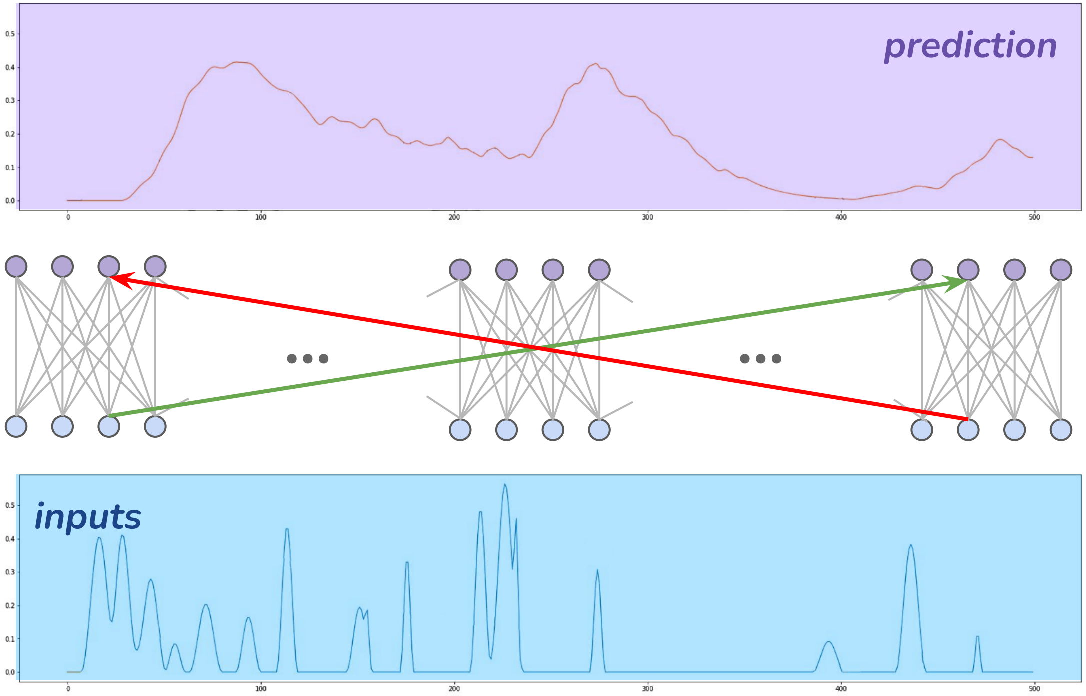


Machine Learning College

Simple feed-forward network



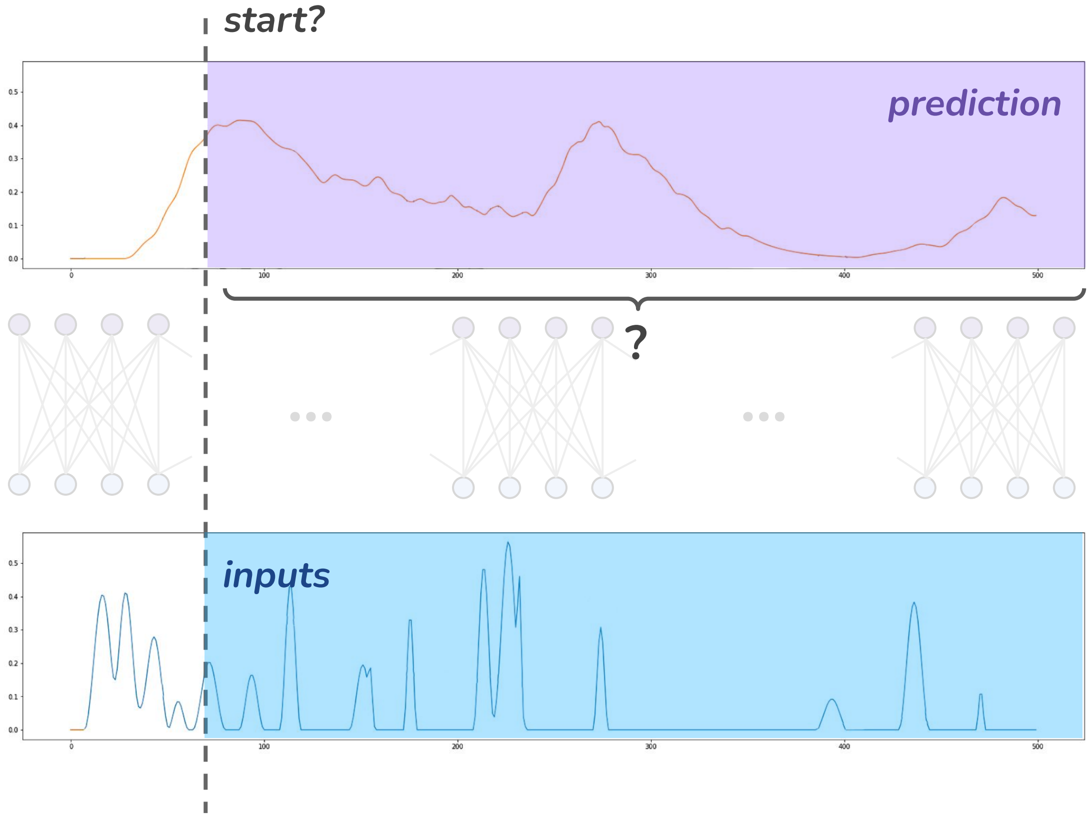
Simple feed-forward network



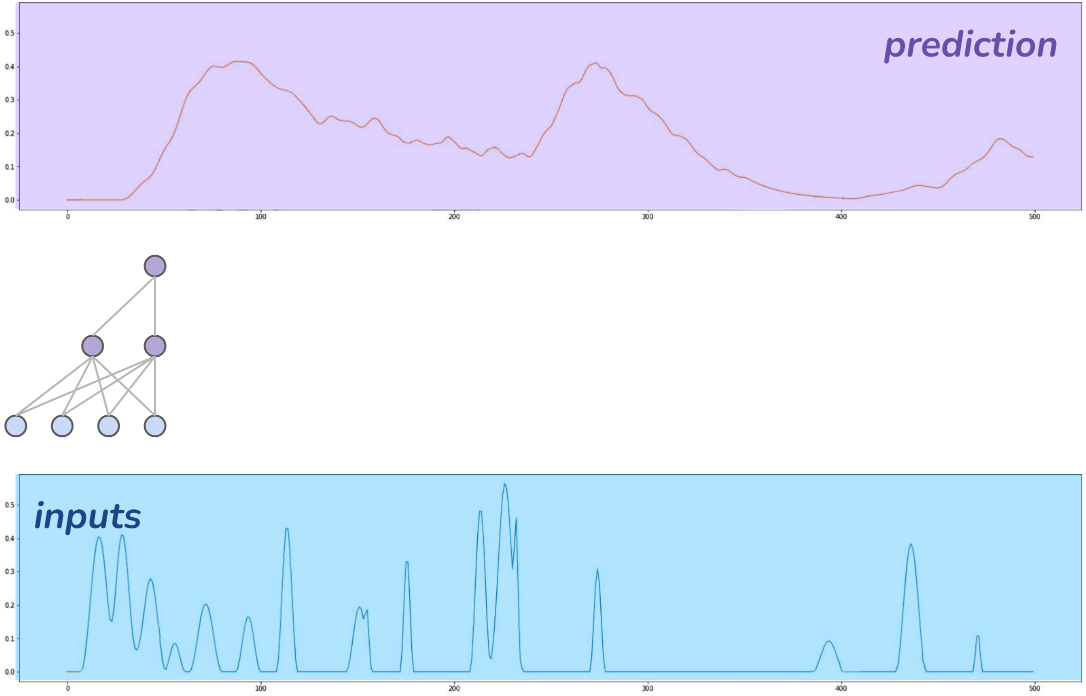
predicting
from future ?

Machine
Learning
College

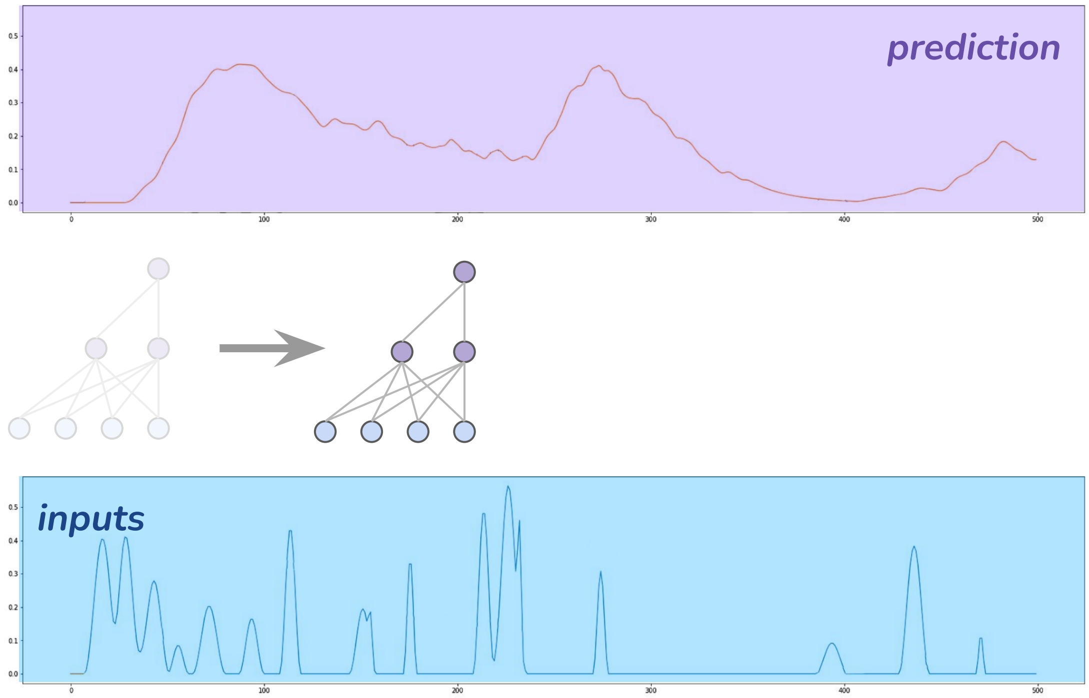
Simple feed-forward network



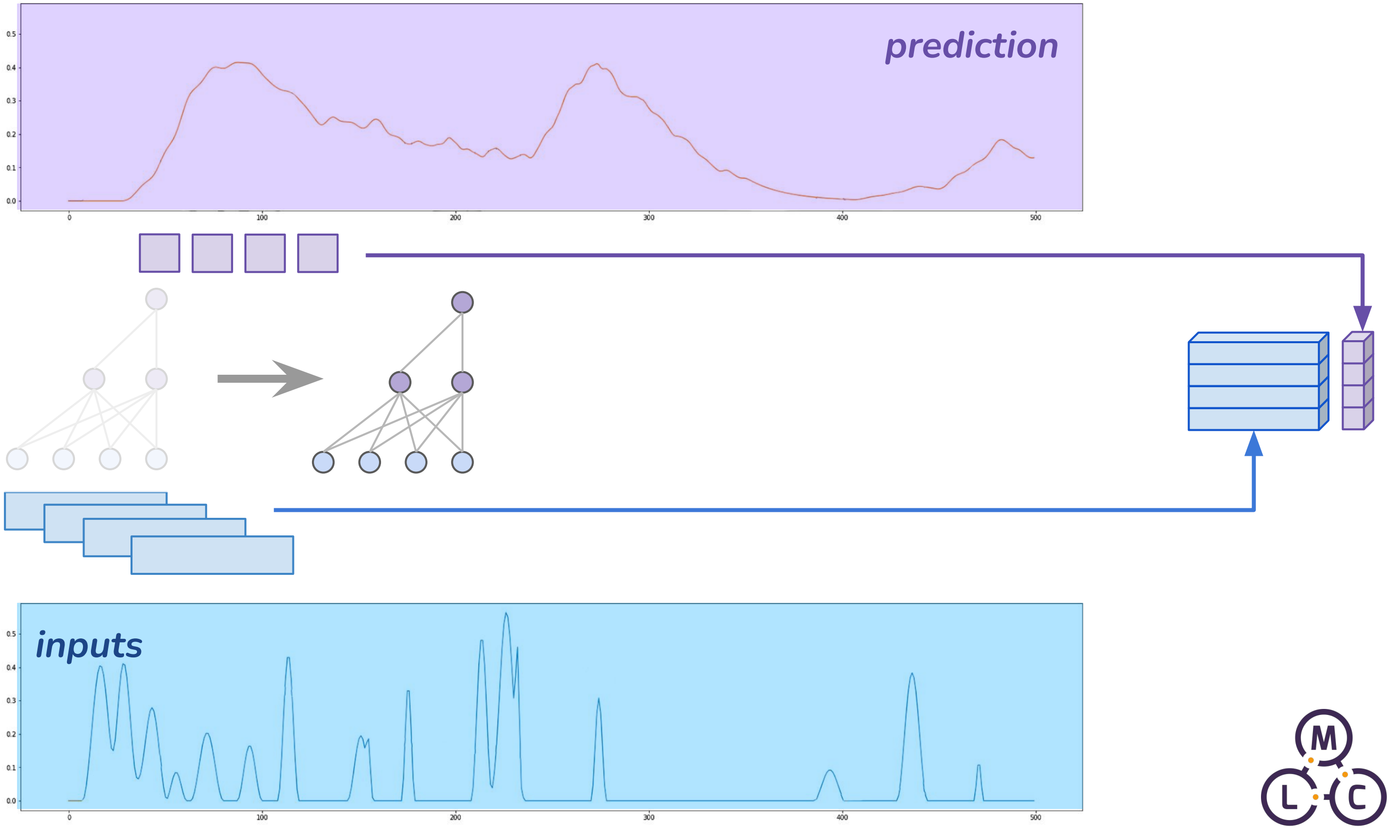
Sliding feed-forward network



Sliding feed-forward network

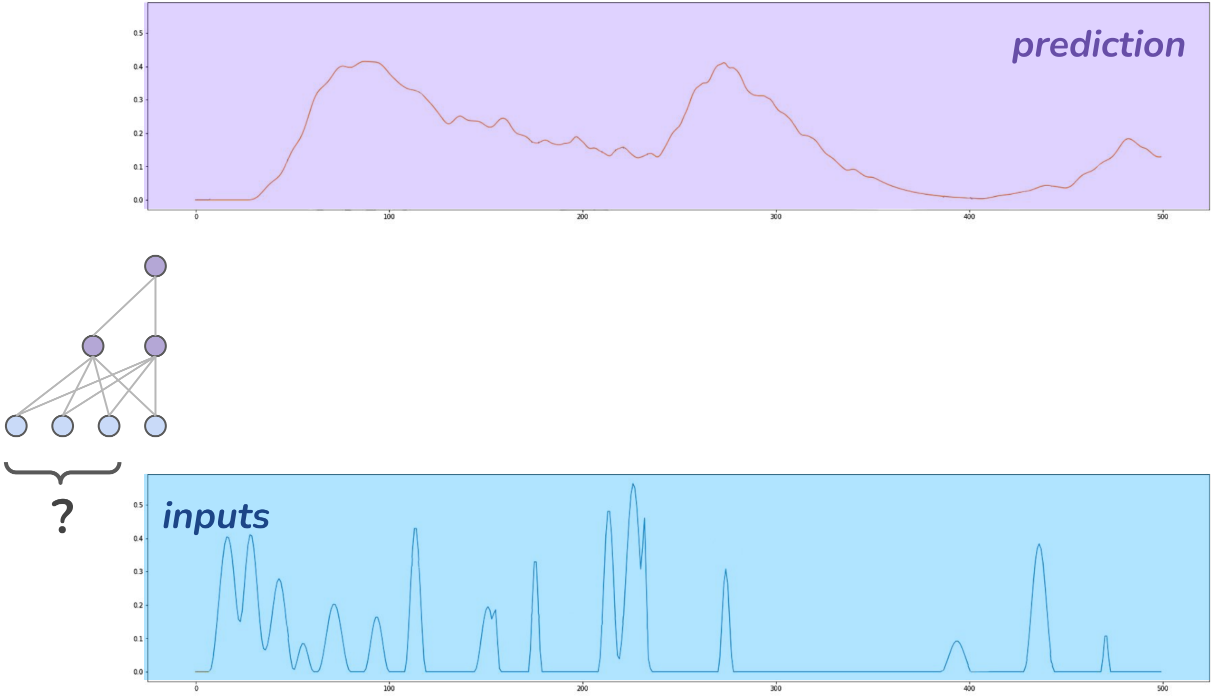


Sliding feed-forward network



Machine Learning College

Sliding feed-forward network

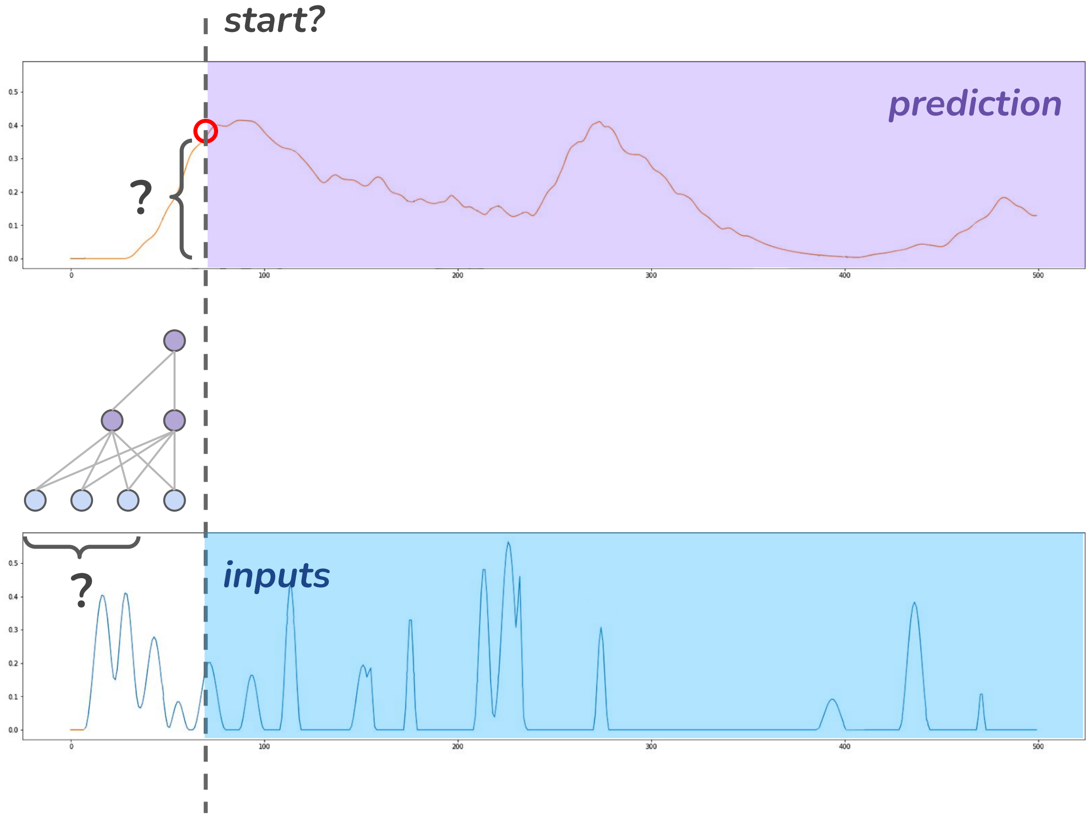


Machine Learning College

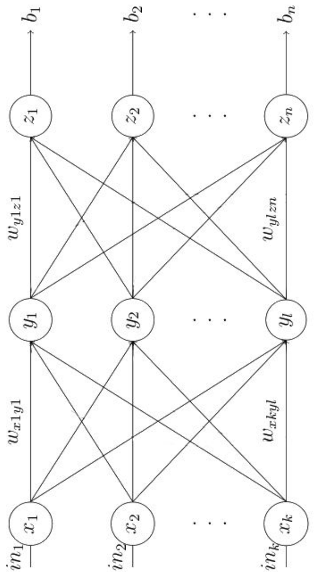
Sliding feed-forward network

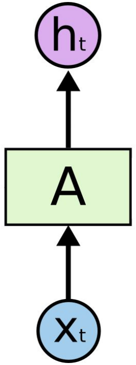
Machine Learning College

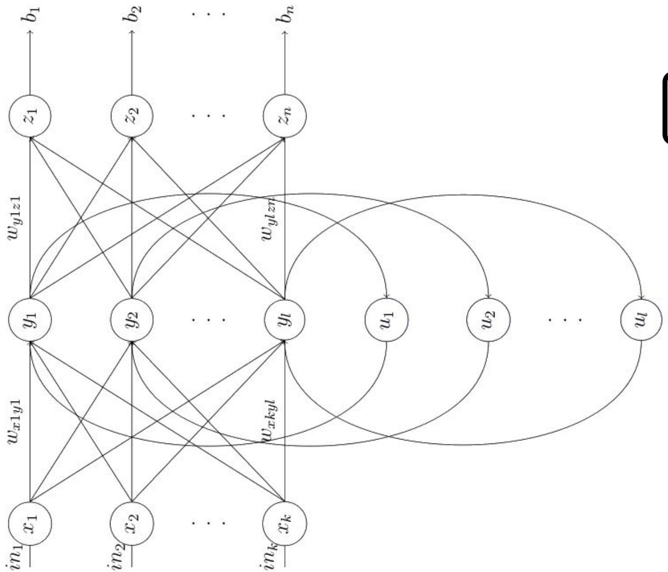
Sliding feed-forward network

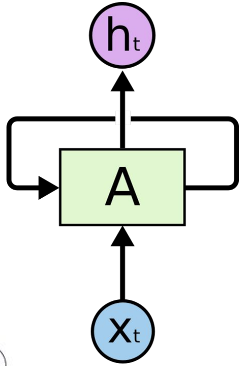


Recurrent Neural Networks



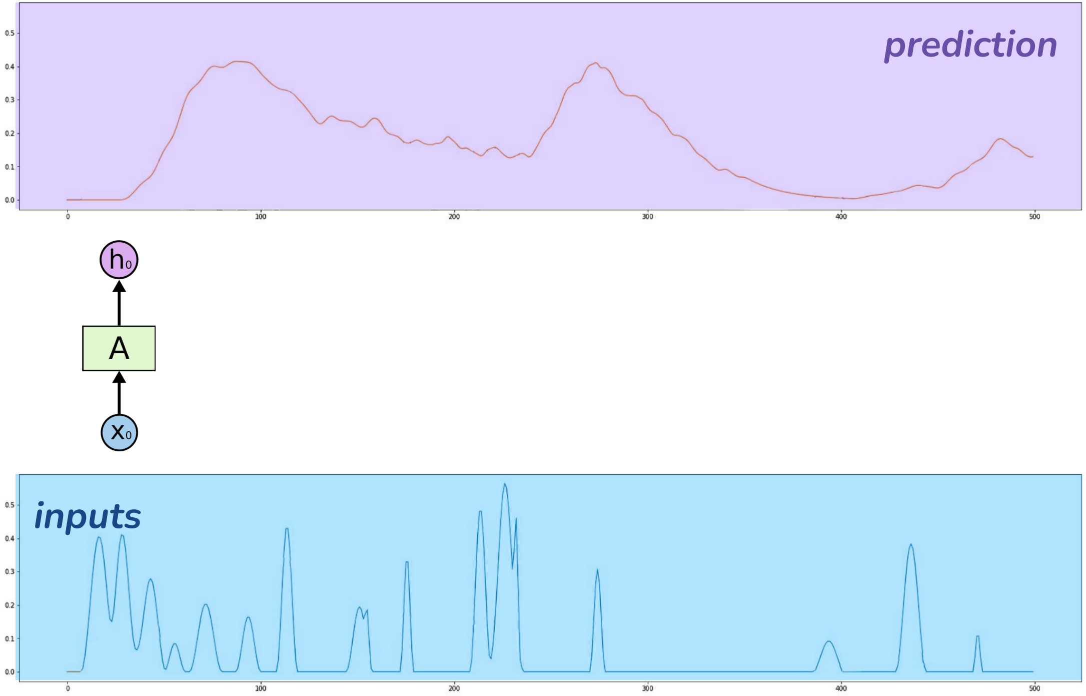






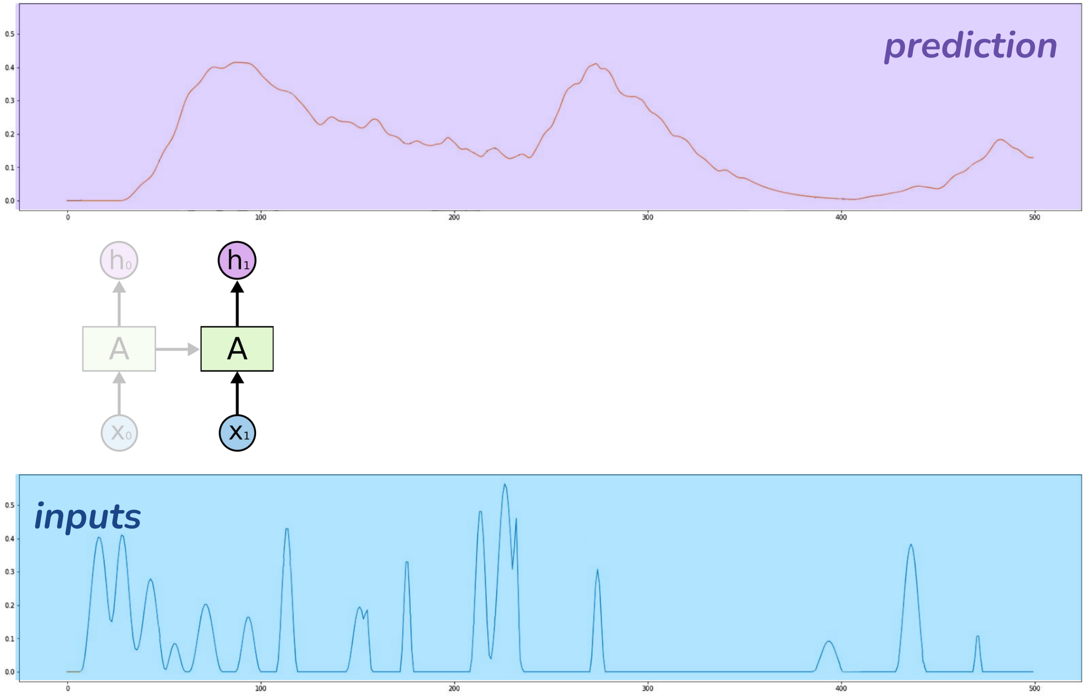
*http://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNN for time series prediction



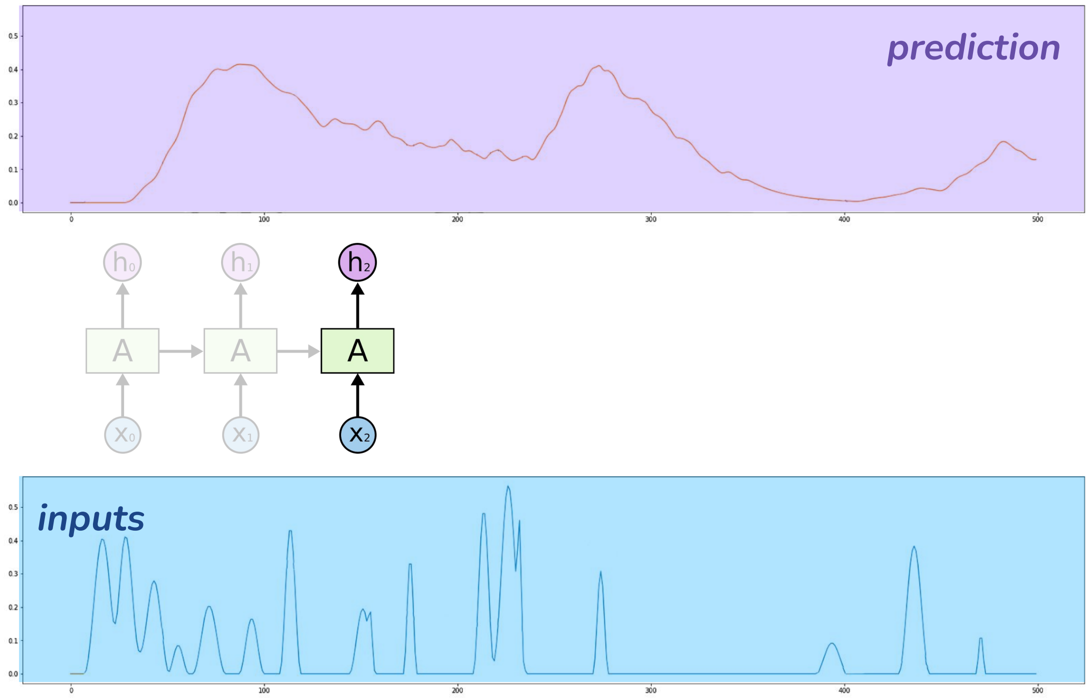
Machine Learning College

RNN for time series prediction



Machine Learning College

RNN for time series prediction

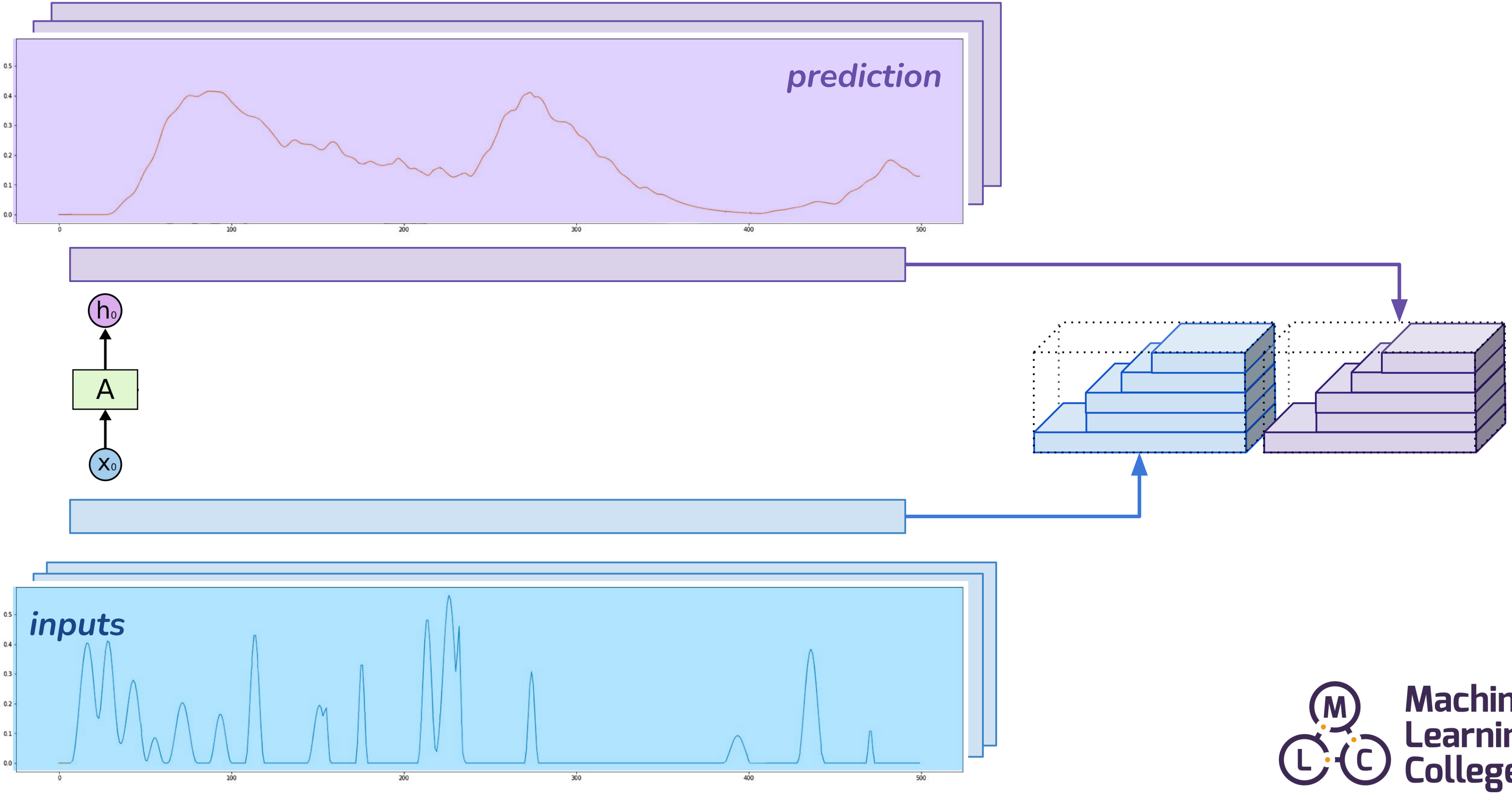


Machine Learning College

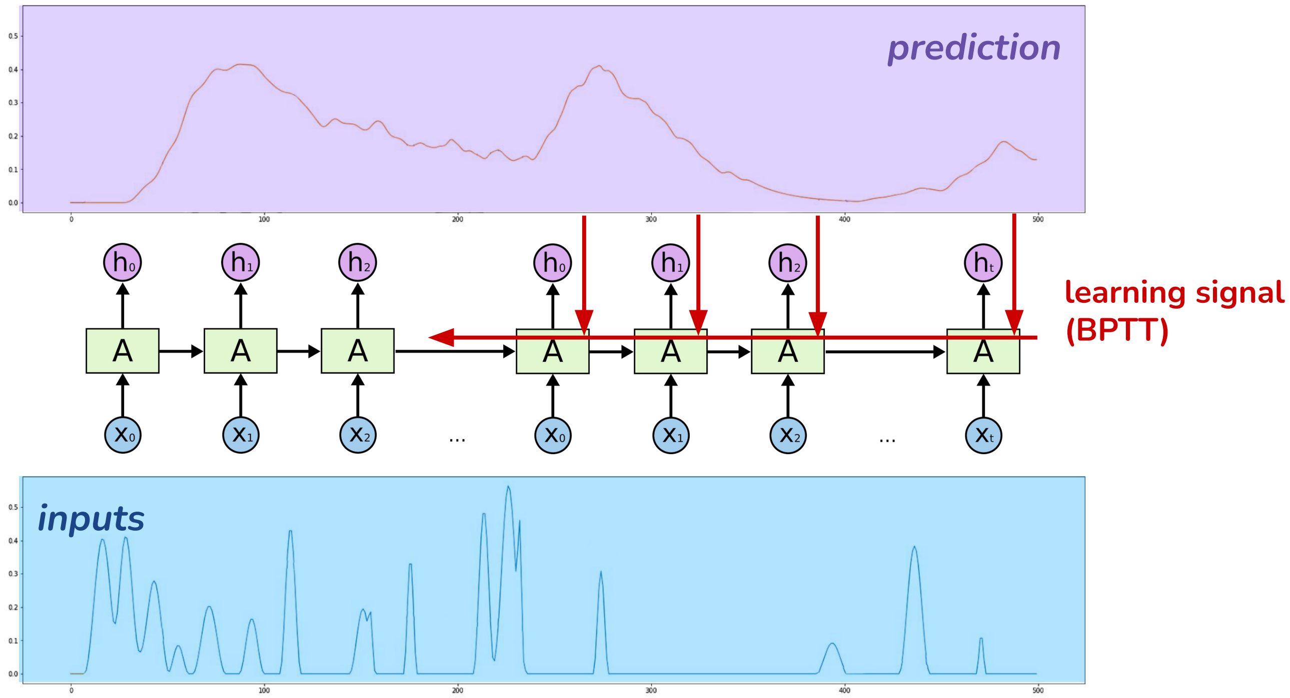
RNN for time series prediction

Machine Learning College

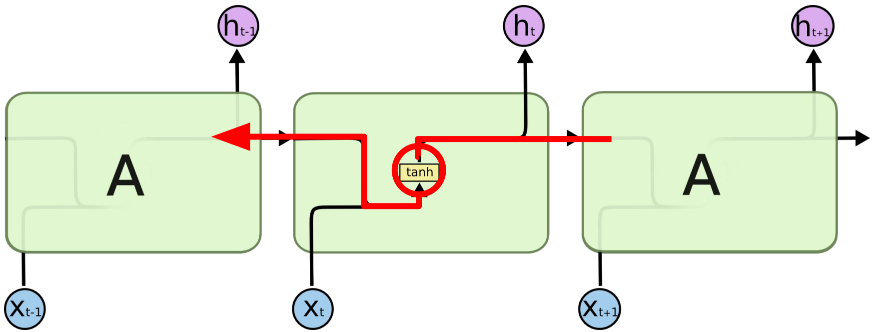
RNN for time series prediction

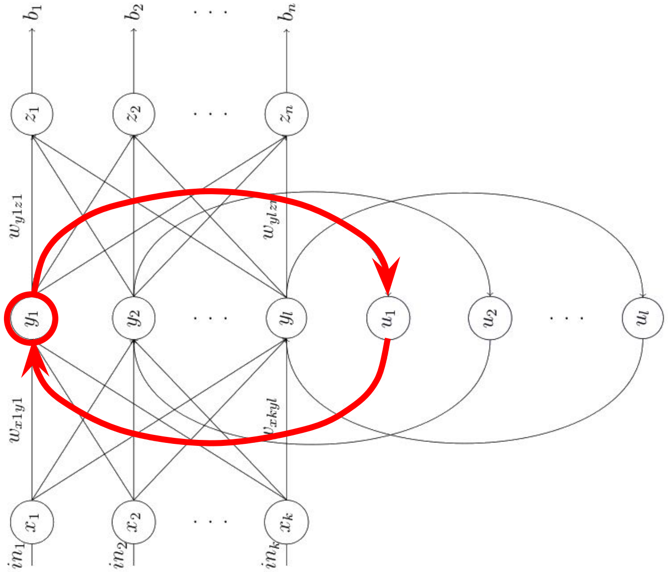


RNN for time series prediction



RNN – Vanishing gradients

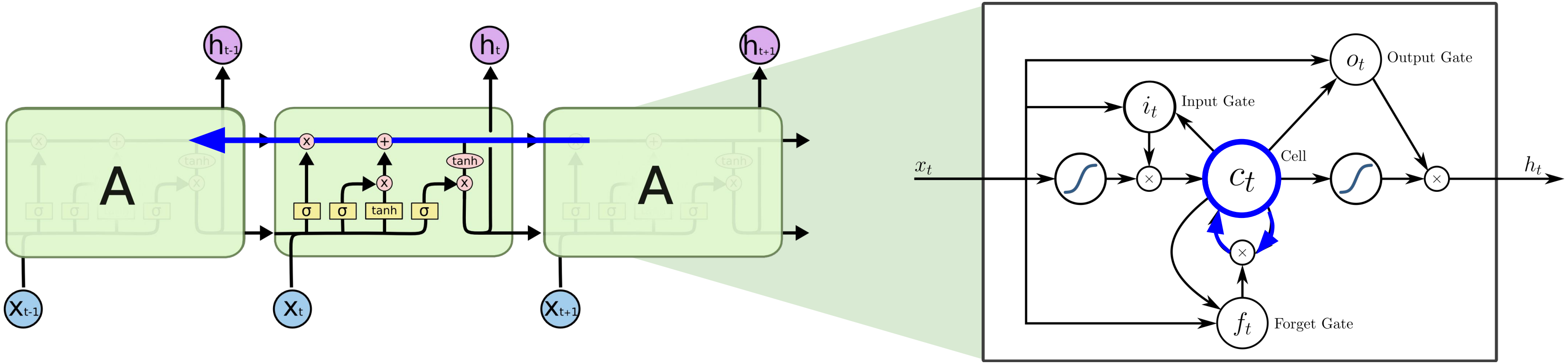




*http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Machine Learning College

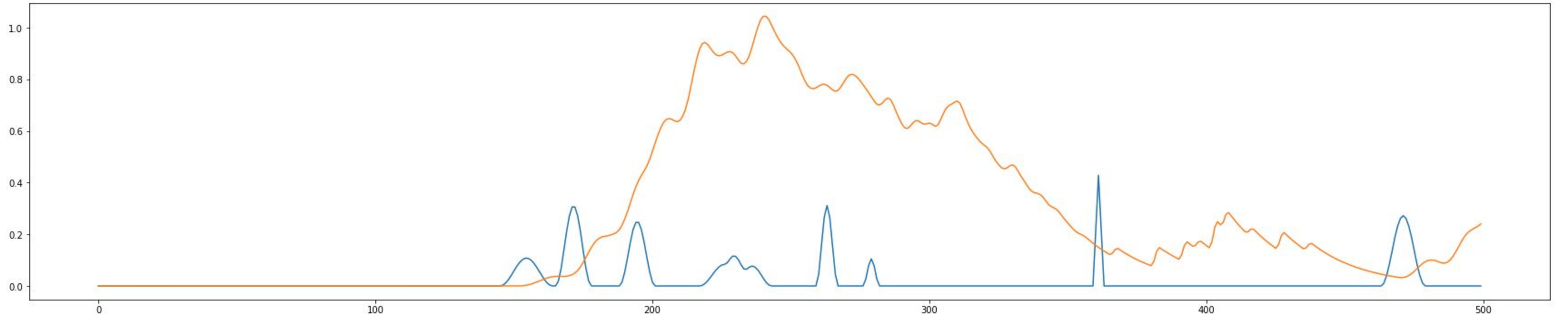
Long short-term memory – LSTM



*http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Rainfall-runoff example

rainfall_runoff.ipynb

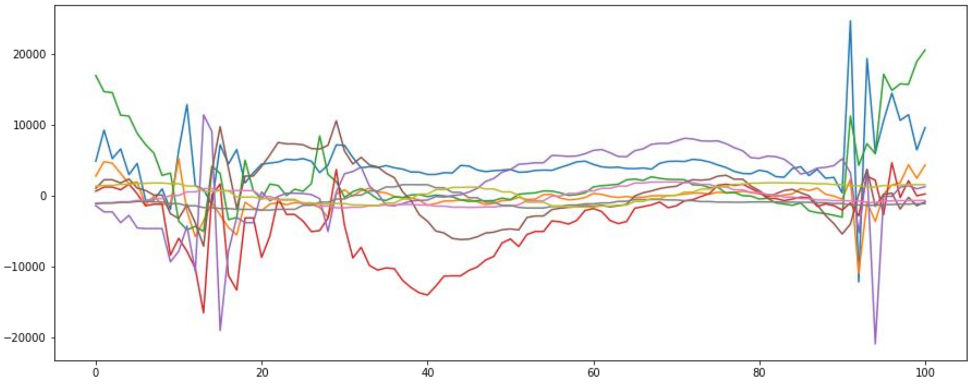


- Test out neural networks on simple **generated rainfall-runoff dataset**
  - Simulated **long-time dependencies** in data
  - Test various neural network architectures
    - Flat feed-forward network
    - LSTM

Machine Learning College

Trampoline jumping example

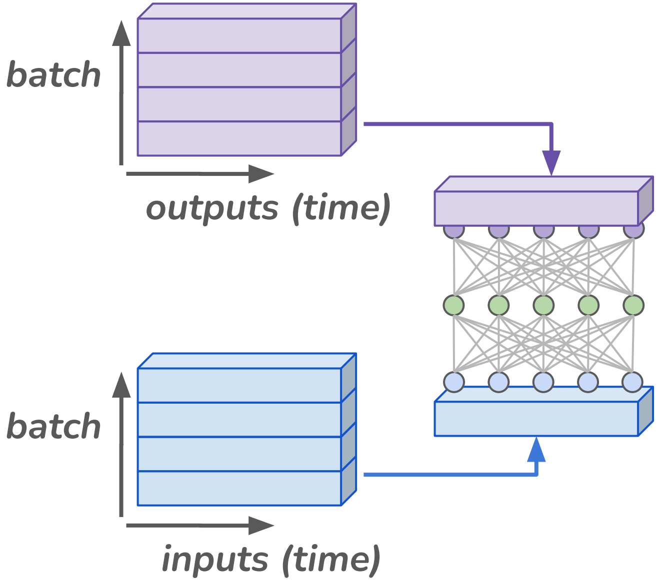
- Data preparation
  - Dataset normalization
  - Sequence padding
- Binary classification task
  - Target values & dimensions
  - Loss functions
- Training & evaluation
  - Inference visualization
  - Evaluation metrics



Machine Learning College

Tensors & dimentions – univariate regression

Flat NN = 2D training data

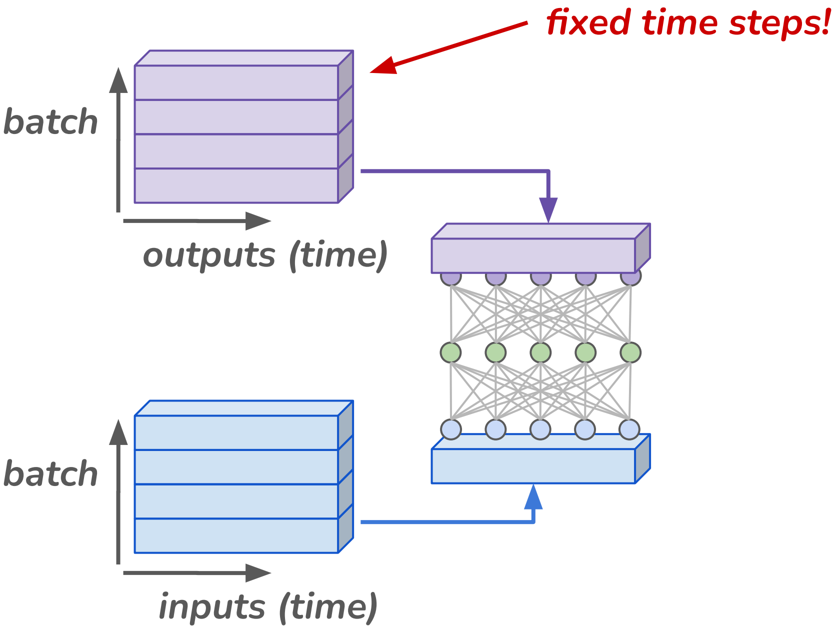


(B, T)⟶(B, T)

Machine Learning College

Tensors & dimentions – univariate regression

Flat NN = 2D training data

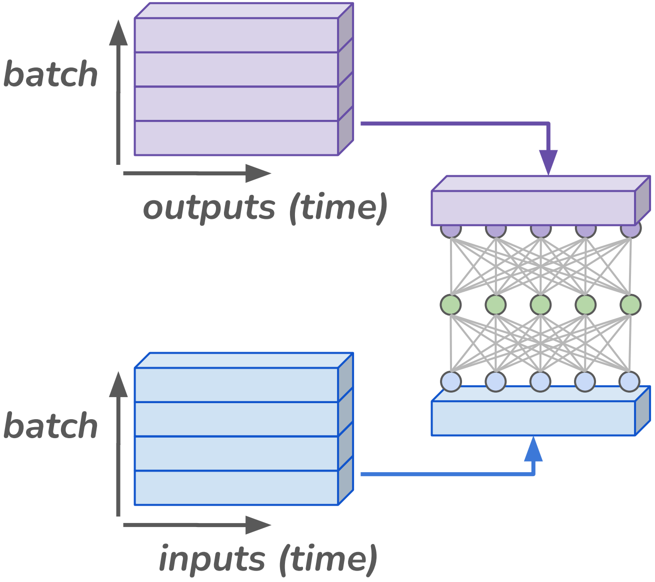


(B, T)⟶(B, T)

Machine Learning College

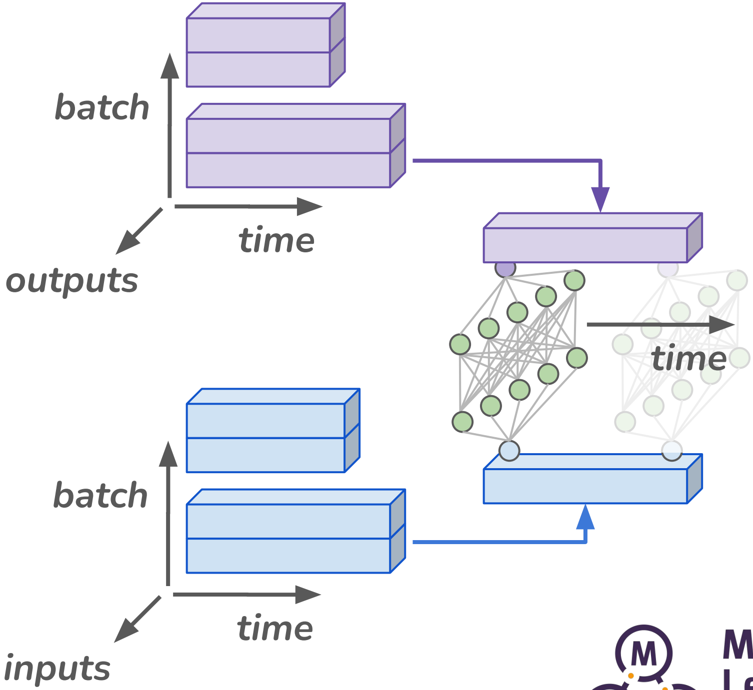
Tensors & dimentions – univariate regression

Flat NN = 2D training data



$(B, T) \longrightarrow (B, T)$

Recurrent NN = 3D training data

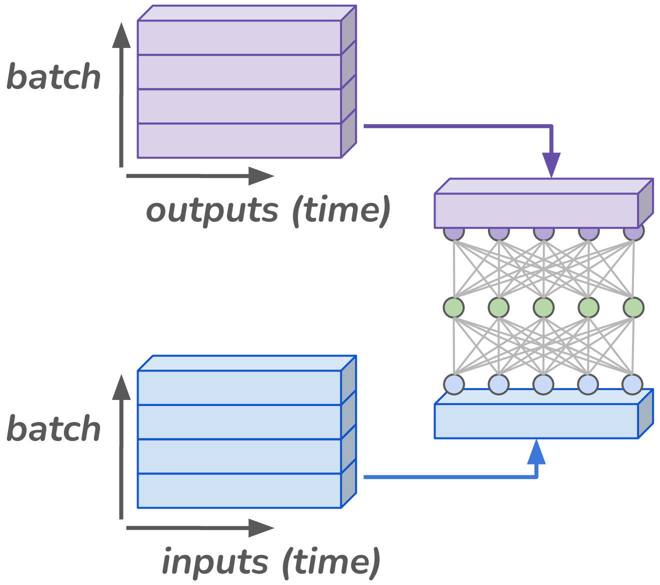


$(B, T, 1) \longrightarrow (B, T, 1)$

Machine Learning College

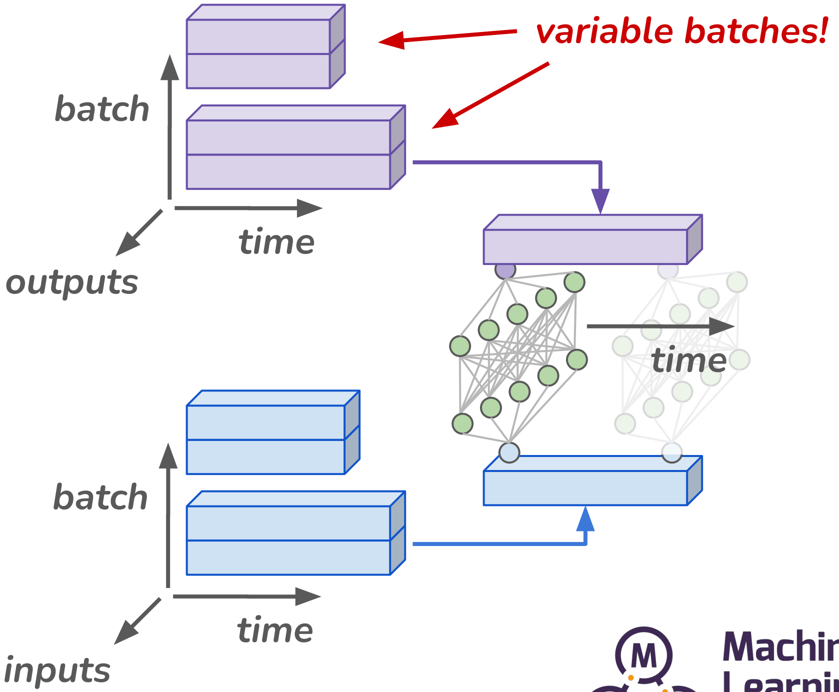
Tensors & dimentions – univariate regression

Flat NN = 2D training data



(B, T) ⟶ (B, T)

Recurrent NN = 3D training data

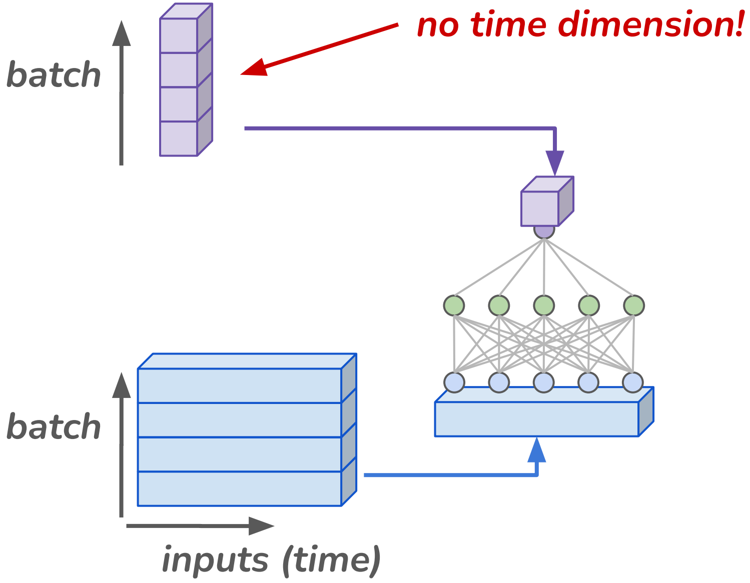


(B, T, 1) ⟶ (B, T, 1)

Machine Learning College

Tensors & dimentions – binary classification

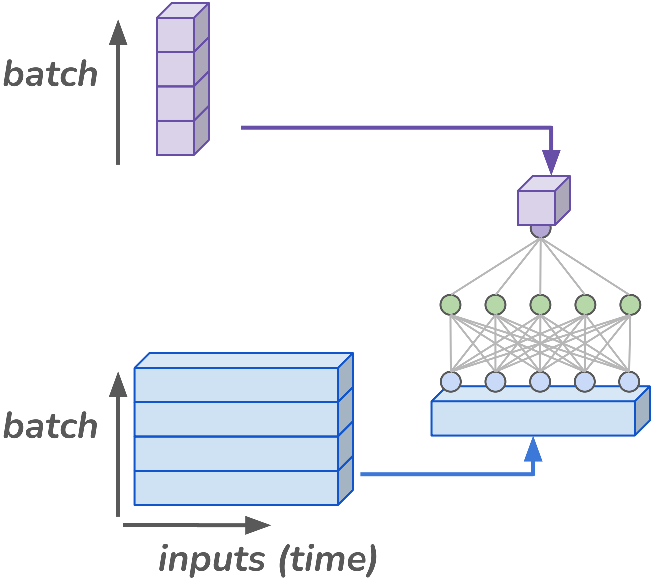
Flat NN = 2D training data



(B, T) ⟶ (B, 1)

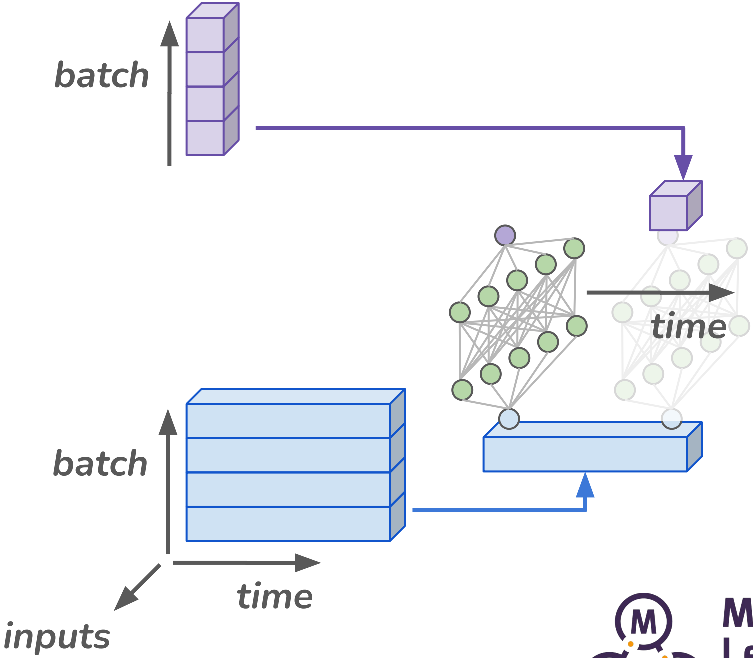
Tensors & dimentions – binary classification

Flat NN = 2D training data



(B, T) ⟶ (B, 1)

Recurrent NN = 3D training data

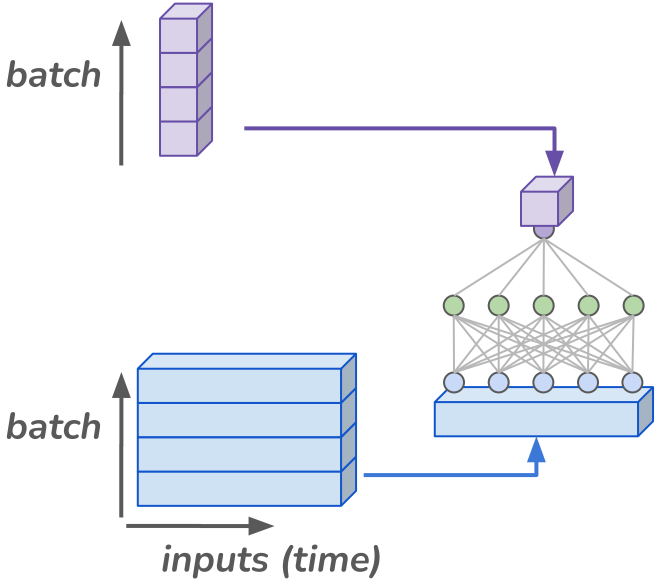


(B, T, 1) ⟶ (B, 1)

Machine Learning College

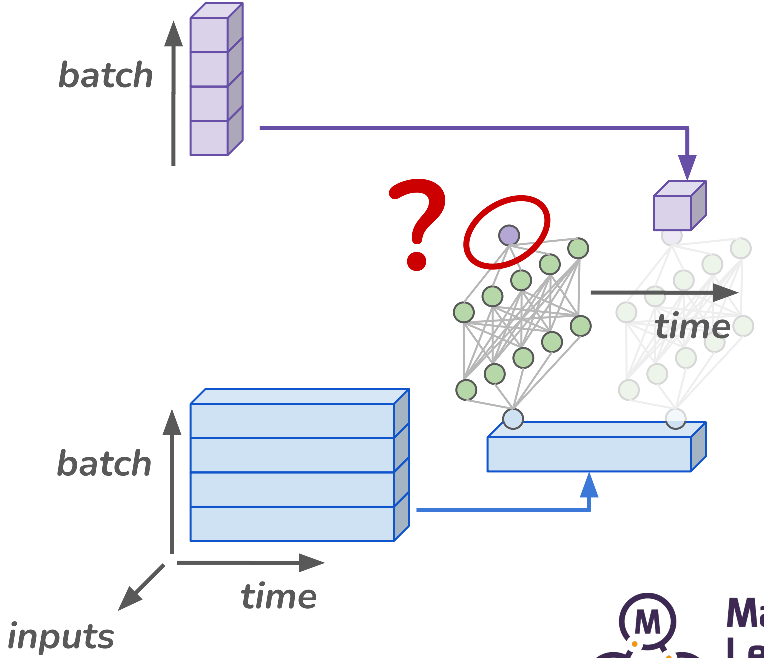
Tensors & dimentions – binary classification

Flat NN = 2D training data



(B, T)⟶(B, 1)

Recurrent NN = 3D training data

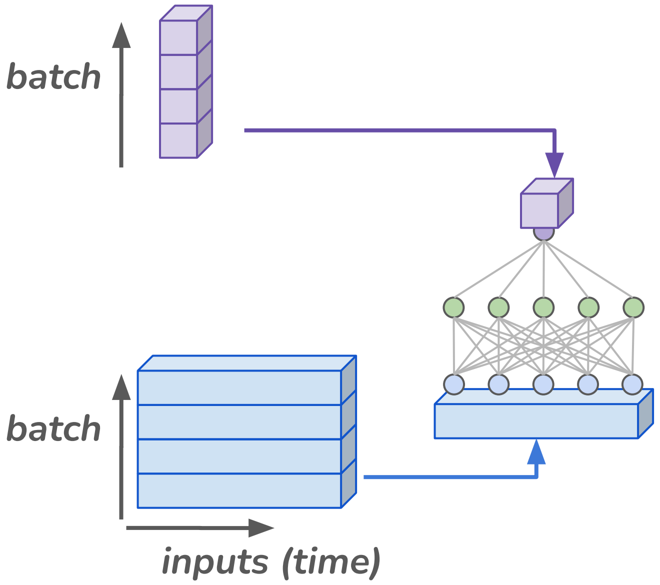


(B, T, 1)⟶(B, 1)

Machine Learning College

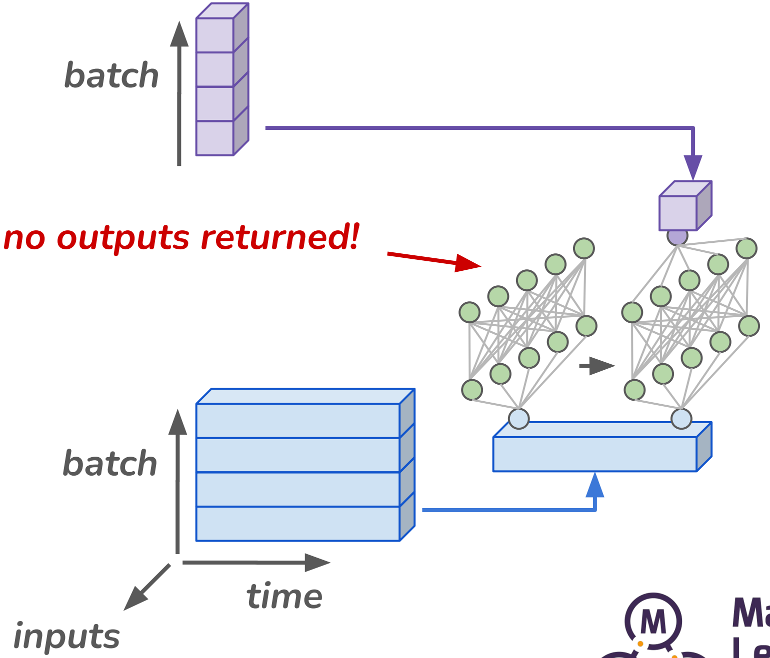
Tensors & dimentions – binary classification

Flat NN = 2D training data



(B, T)⟶(B, 1)

Recurrent NN = 3D training data



(B, T, 1)⟶(B, 1)

Machine Learning College

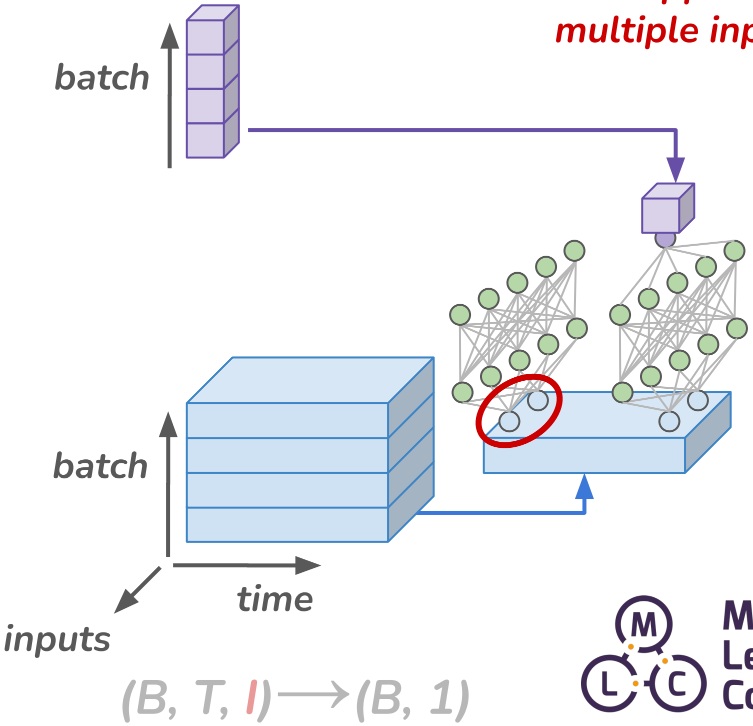
Tensors & dimentions – multivariate b. classification

Flat NN = 2D training data

?

Recurrent NN = 3D training data

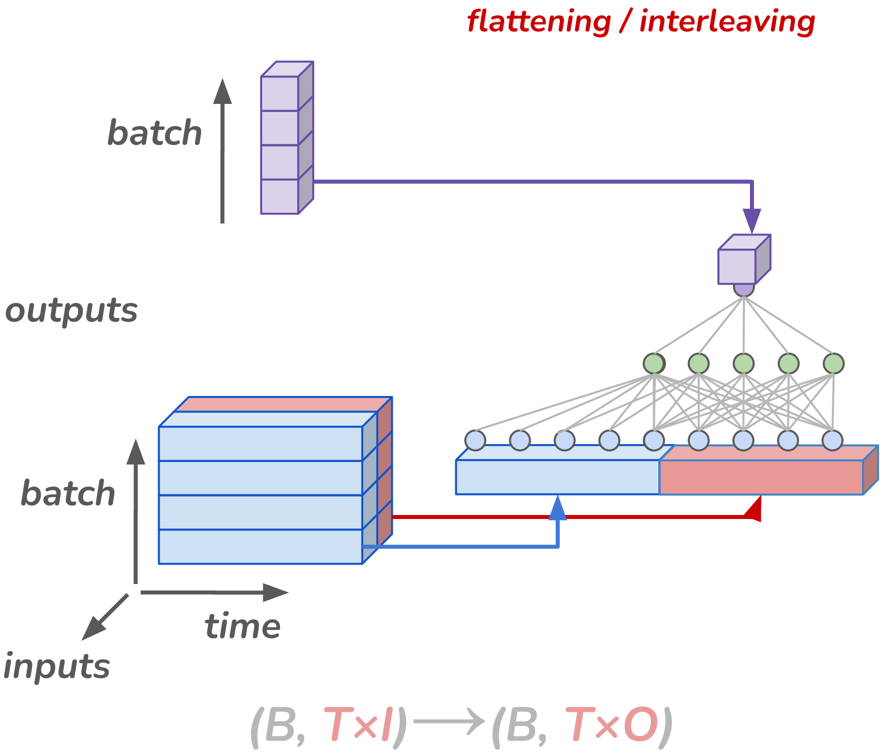
natural support for multiple inputs



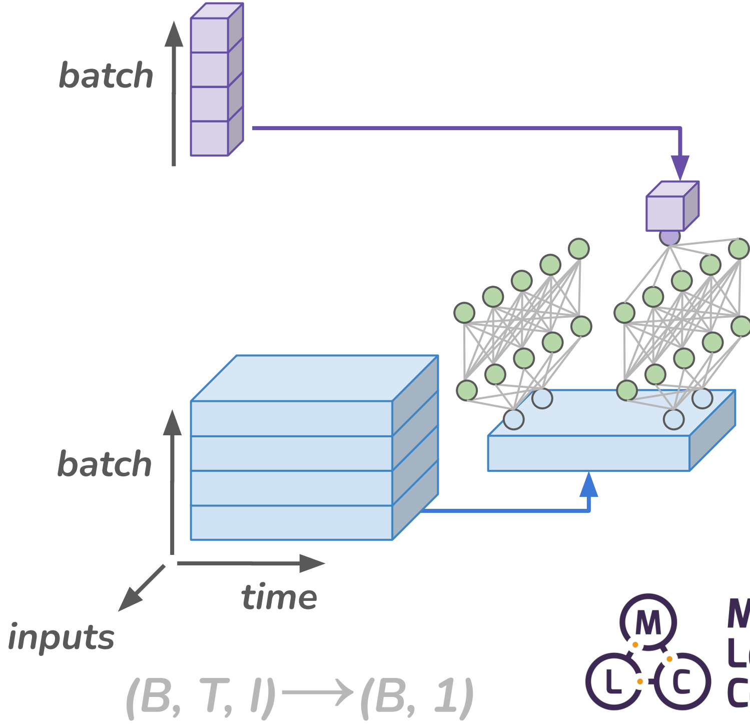
Machine Learning College

Tensors & dimentions – multivariate b. classification

Flat NN = 3D⟶2D training data



Recurrent NN = 3D training data

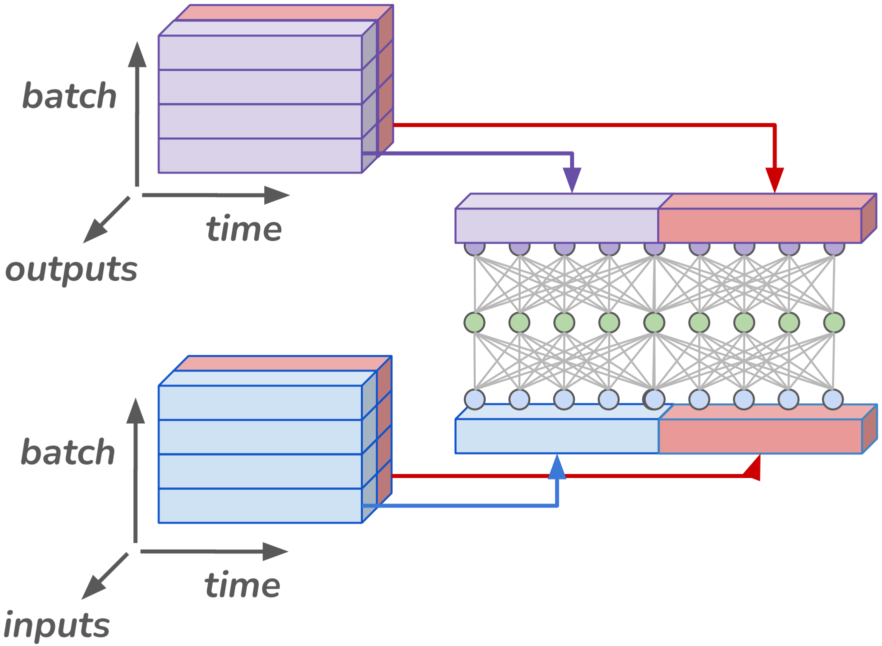


Machine Learning College

Tensors & dimentions – multivariate regression

Flat NN = 3D⟶2D training data

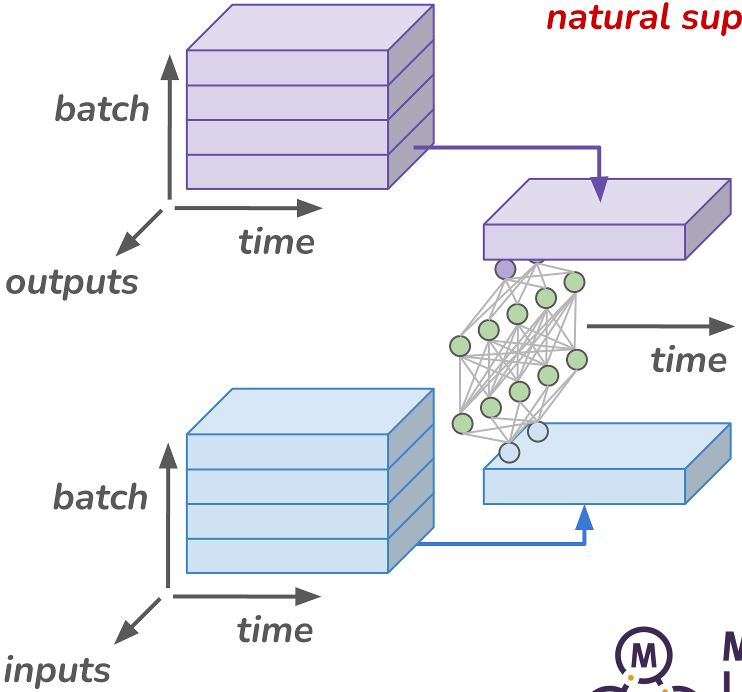
flattening / interleaving



$(B,\ T\times I) \longrightarrow (B,\ T\times O)$

Recurrent NN = 3D training data

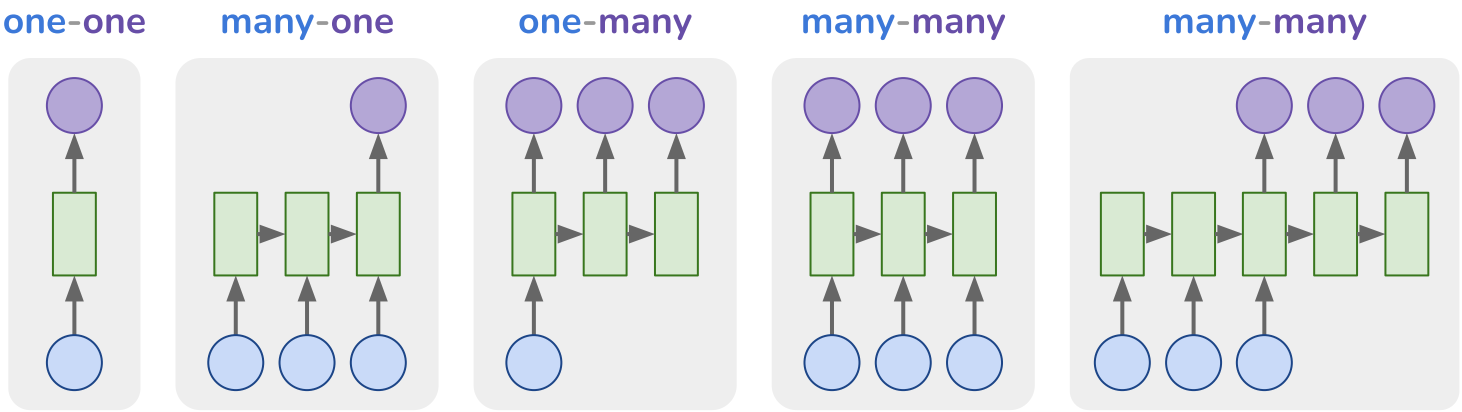
natural support



$(B,\ T,\ I) \longrightarrow (B,\ T,\ O)$

Machine Learning College

RNN and sequence data



sequence classification

image captioning

time series prediction

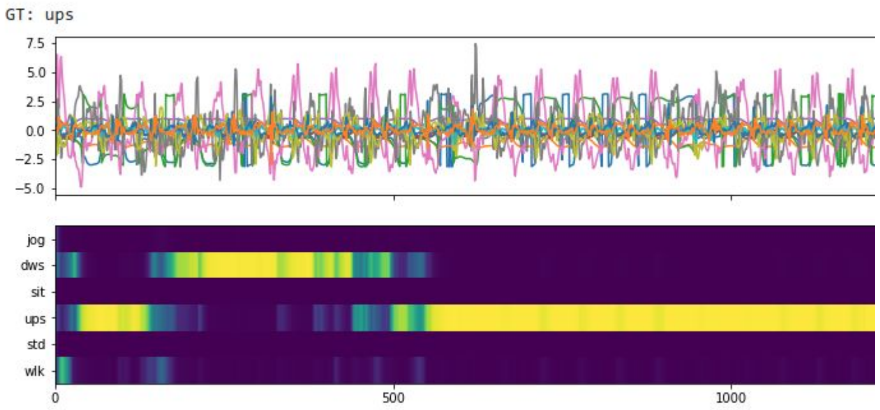
machine translation

*http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Machine Learning College

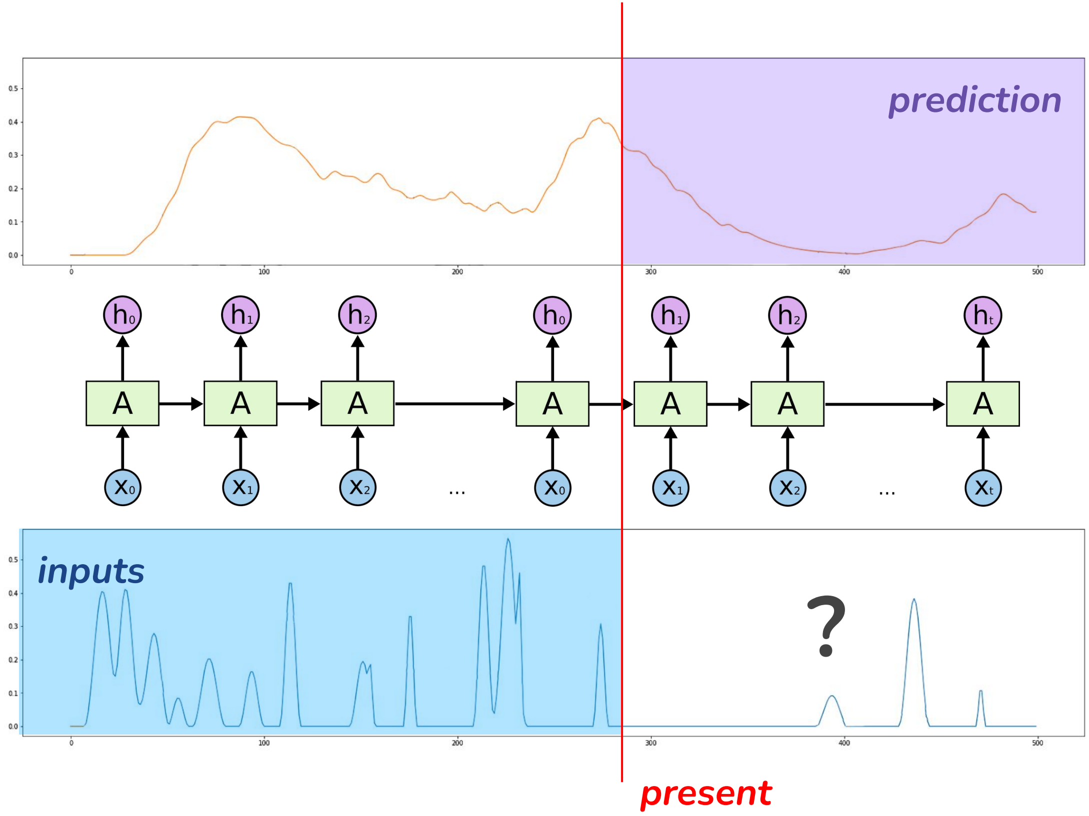
Motion sensing example

- Data preparation
  - Dataset normalization
  - Slicing long sequences
- Categorical classification task
  - Predict activity type
  - Use correct activation & loss function
- Training & evaluation
  - Try different architectures
  - Evaluate resuslt with standard metrics
- Secondary task
  - Subject identification



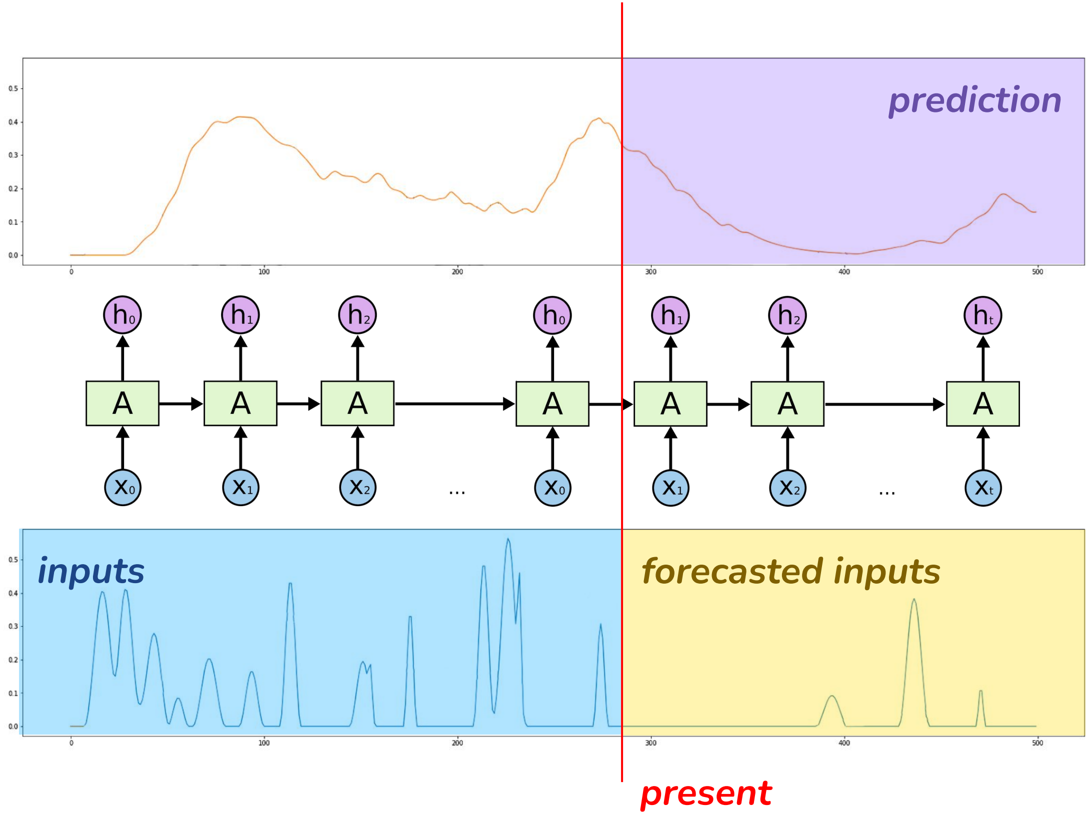
Machine Learning College

Forecasting



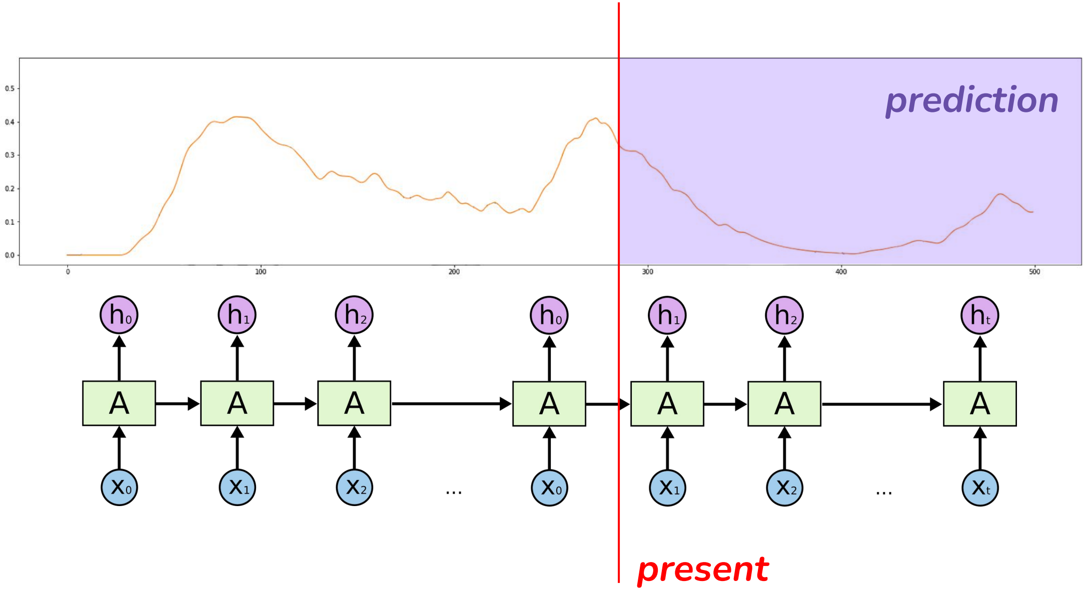
Machine Learning College

Forecasting

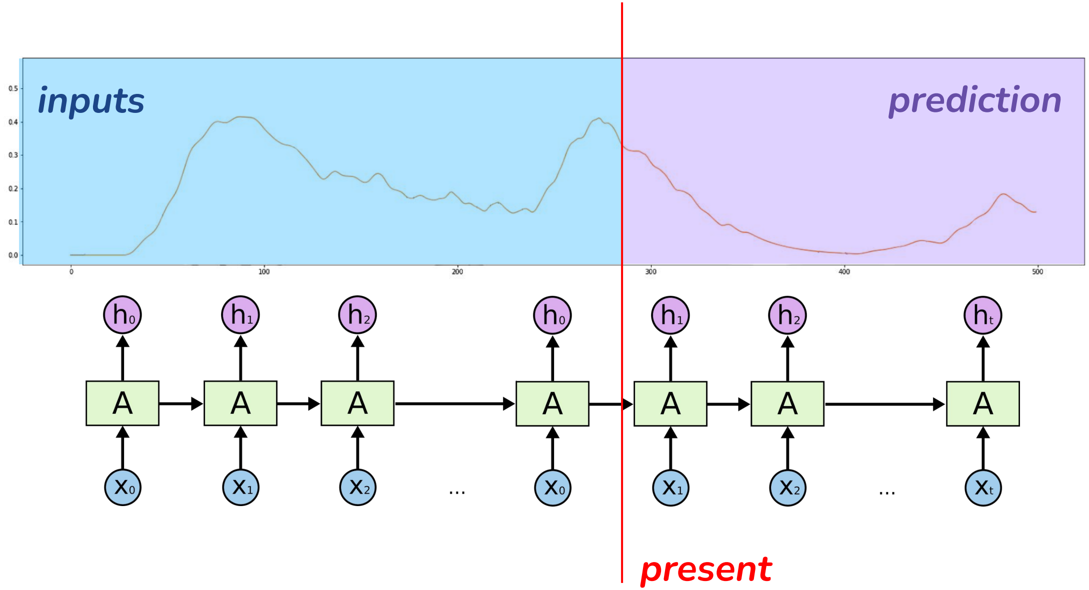


Machine Learning College

Forecasting

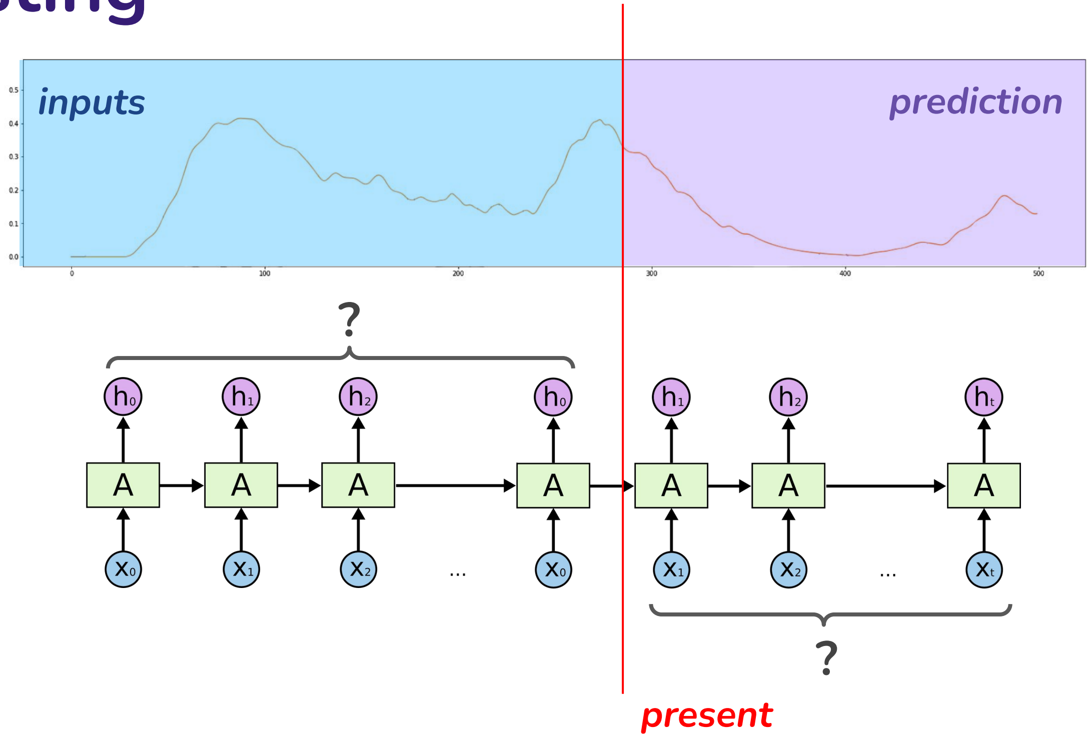


Forecasting



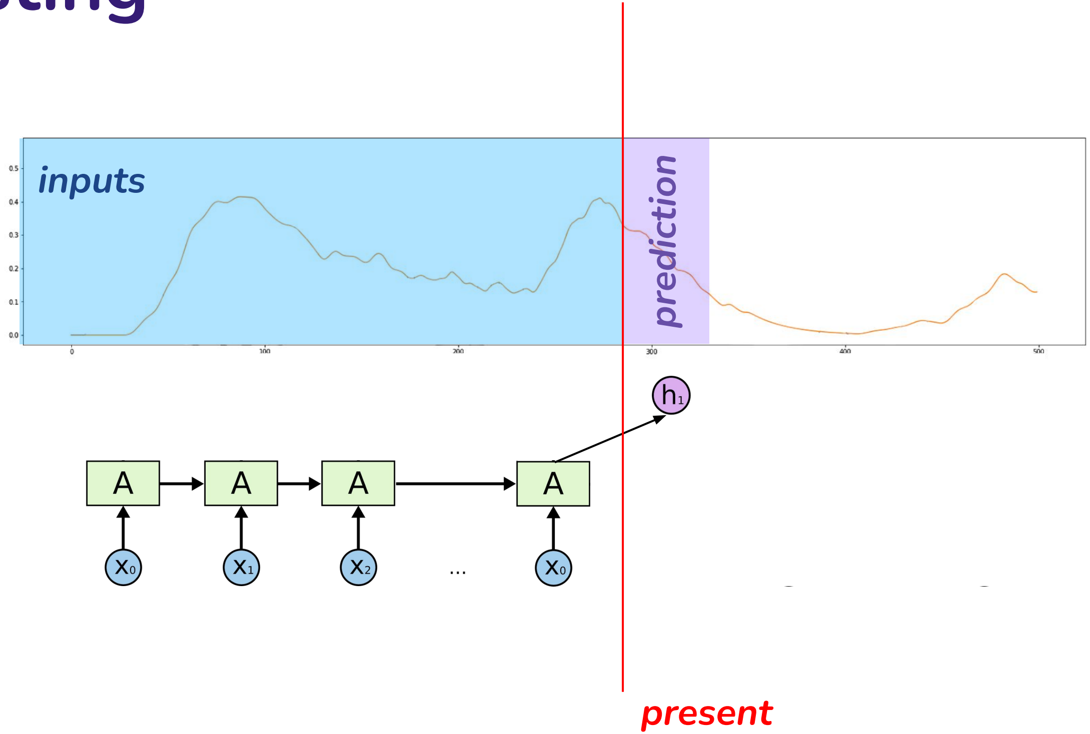
Machine Learning College

Forecasting



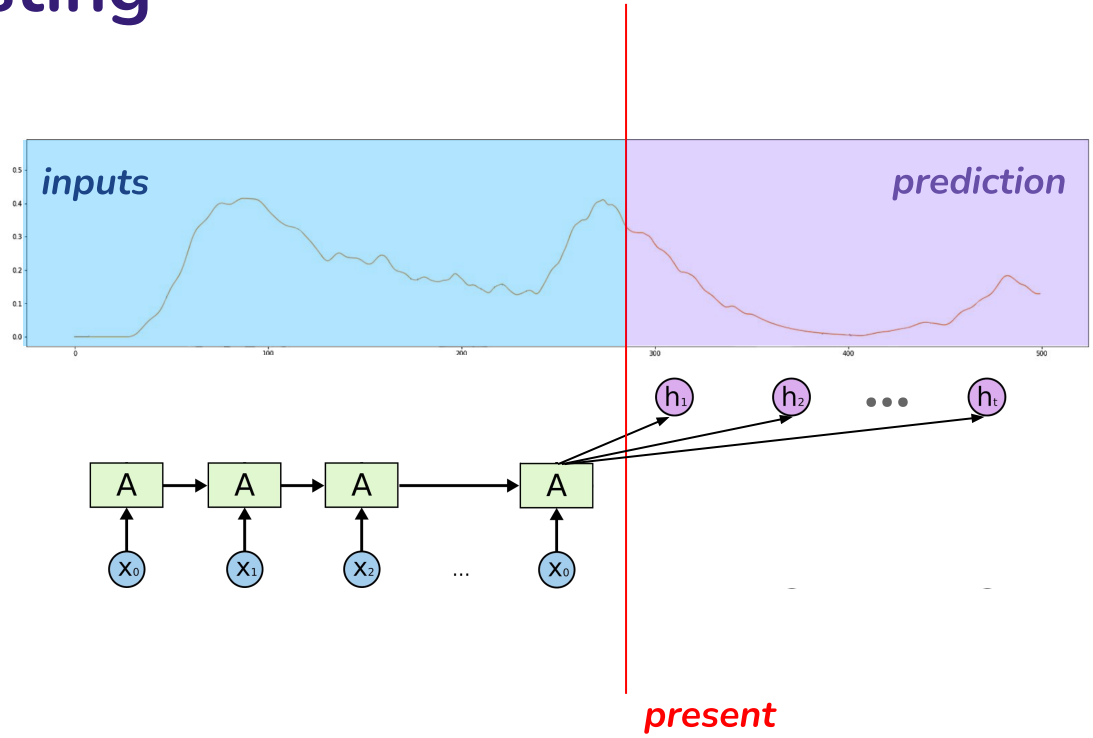
Machine Learning College

Forecasting



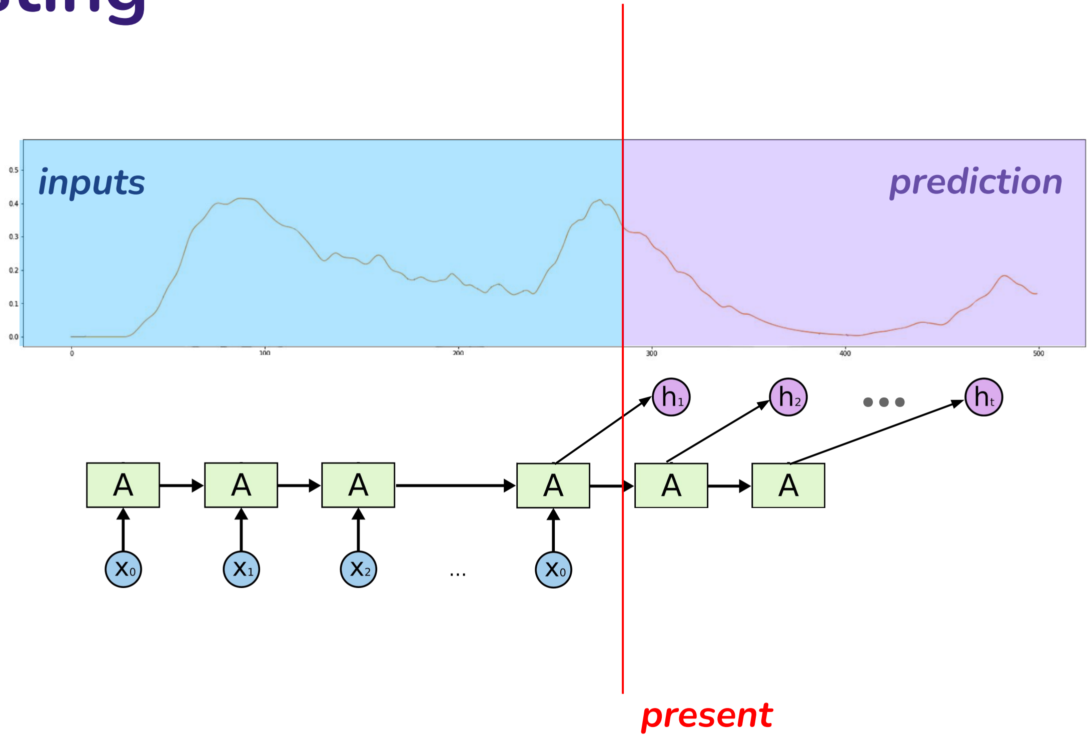
Machine Learning College

Forecasting

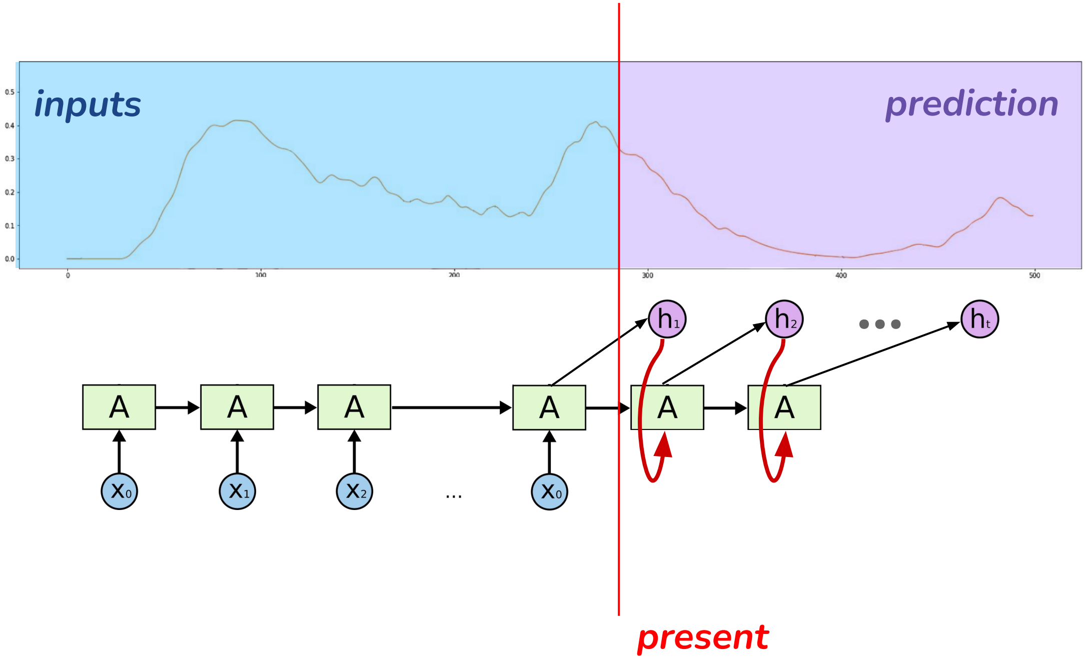


Machine Learning College

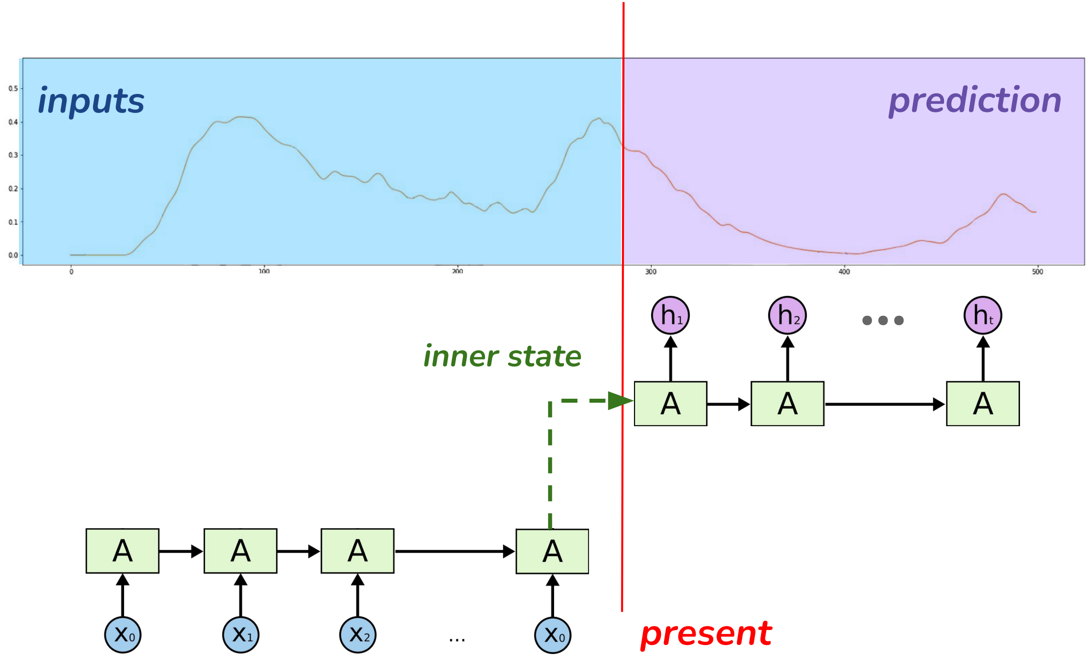
Forecasting



Forecasting

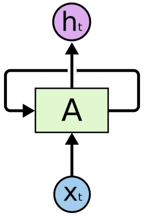


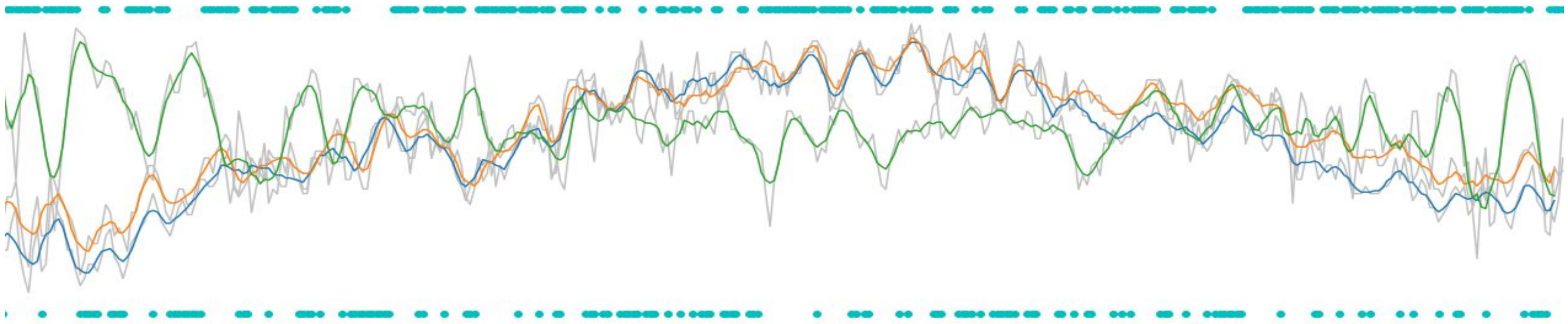
Forecasting – encoder & decoder



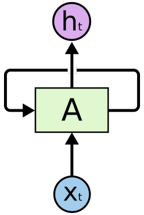
Machine Learning College

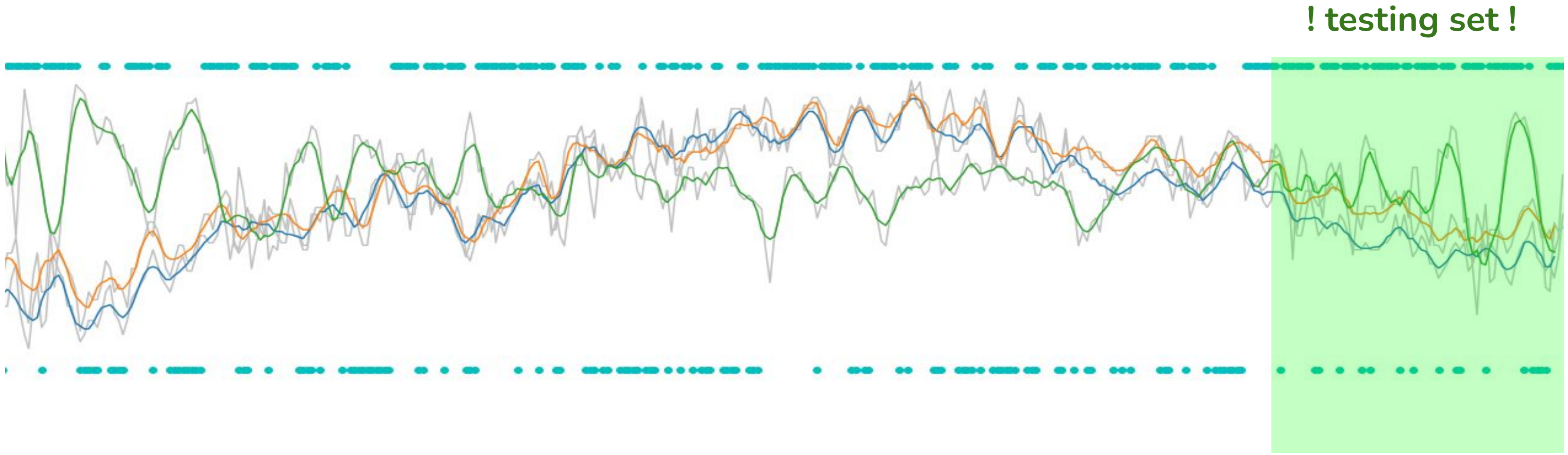
Training set construction



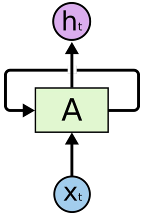


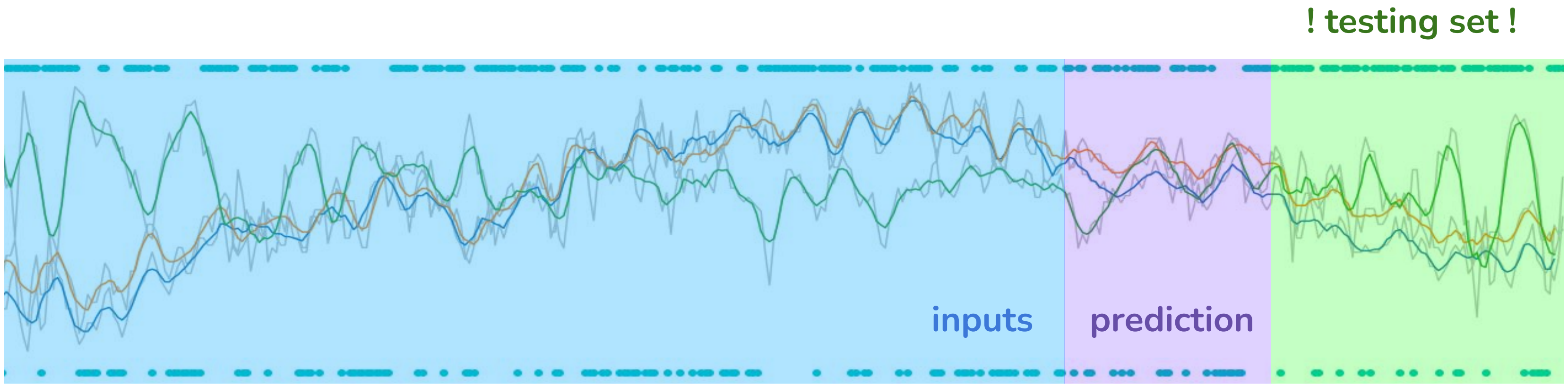
Training set construction



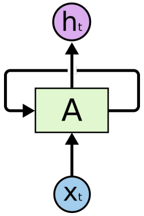


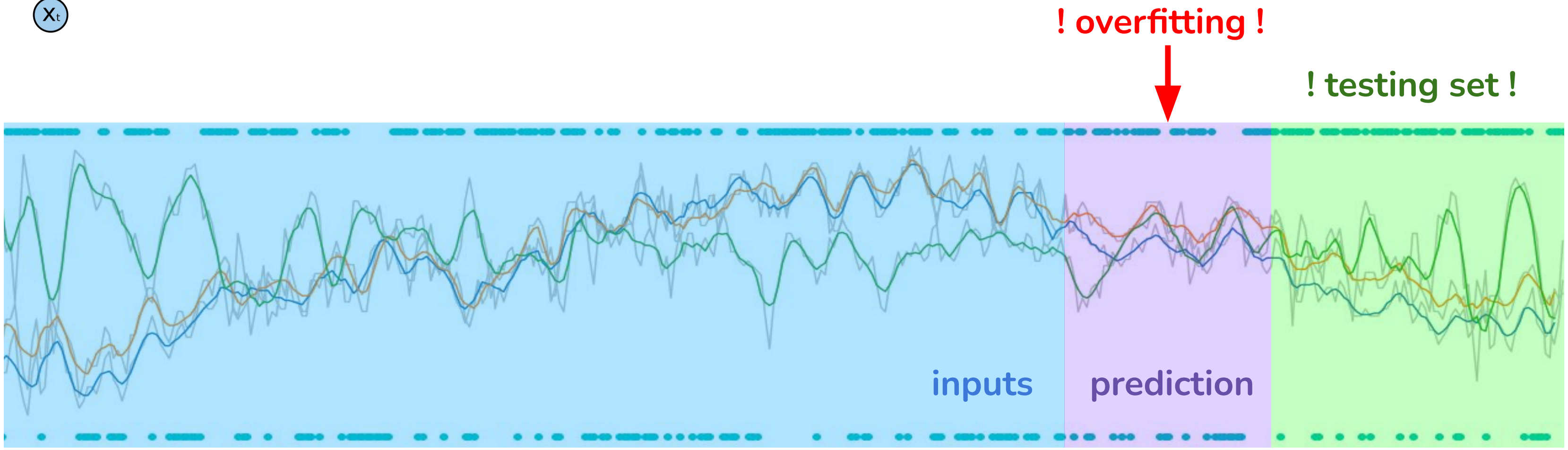
Training set construction



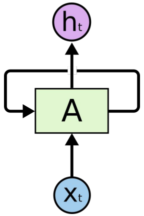


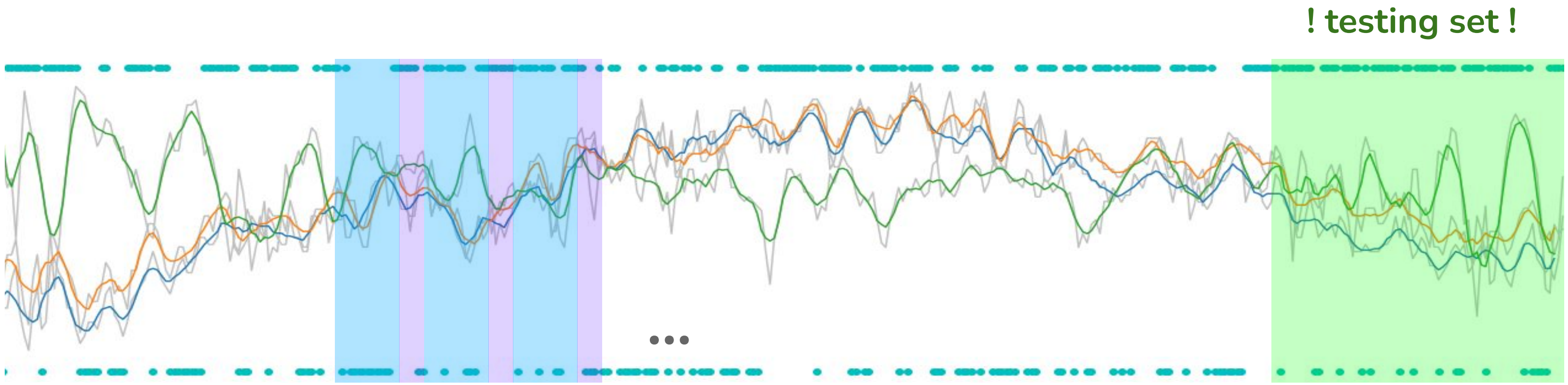
Training set construction



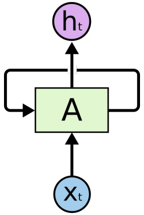


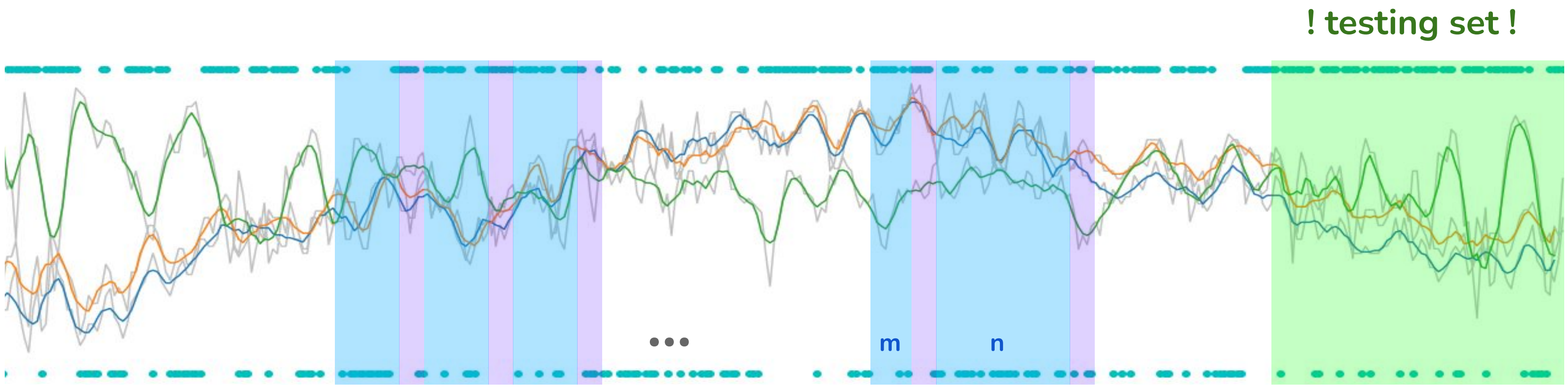
Training set construction



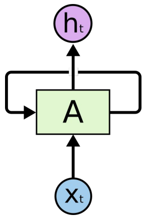


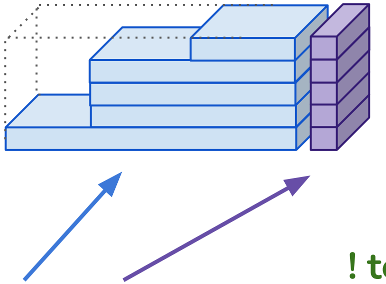
Training set construction

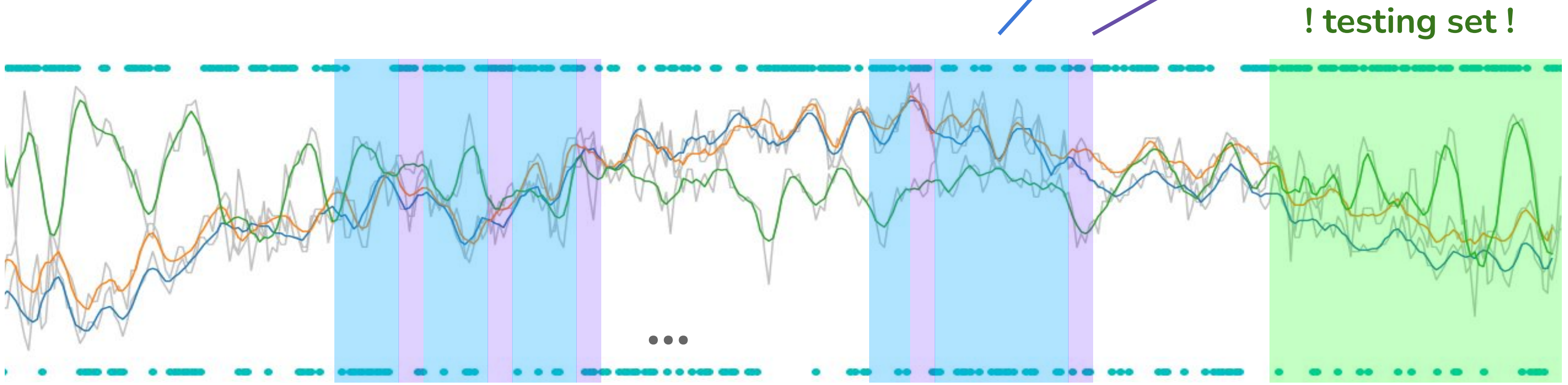




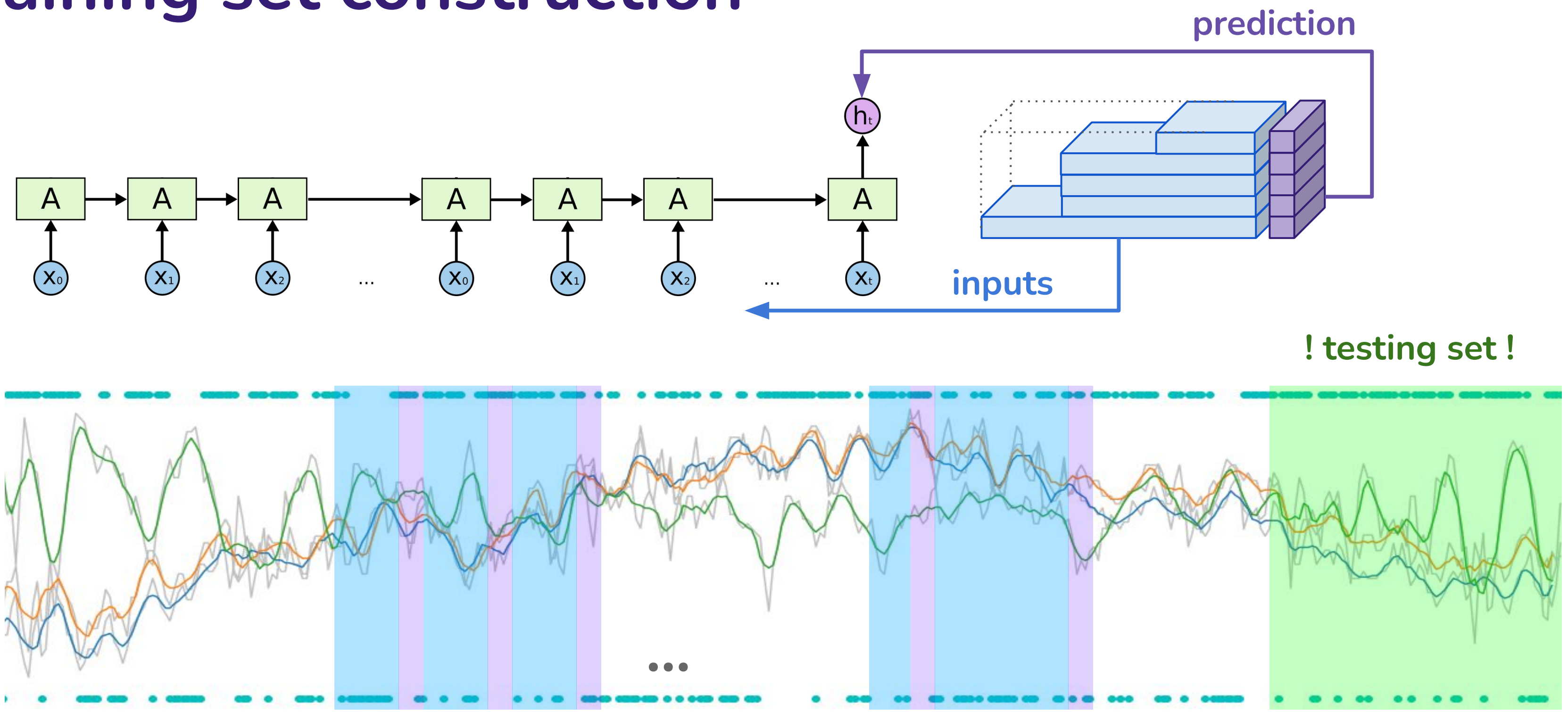
Training set construction



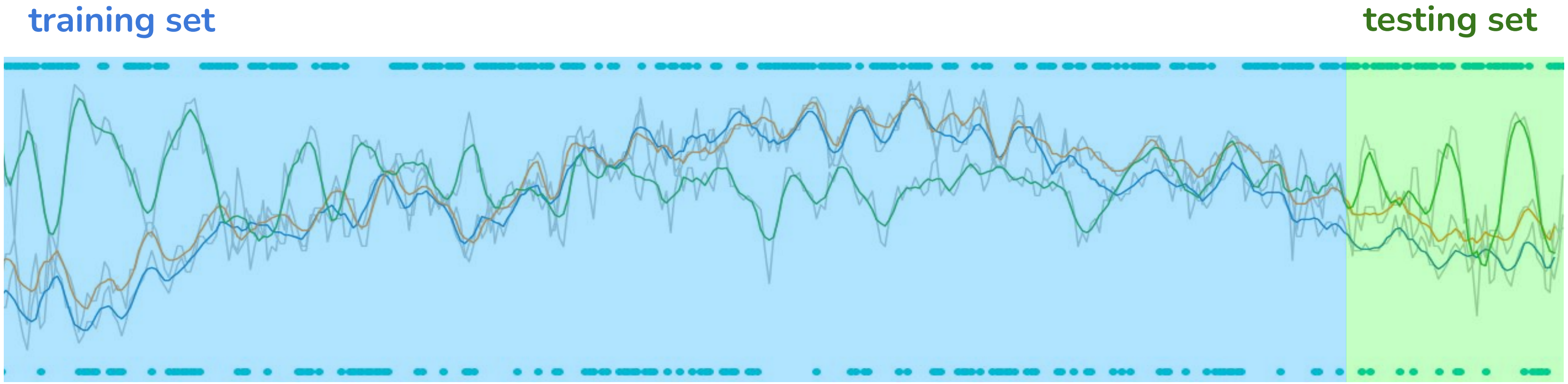




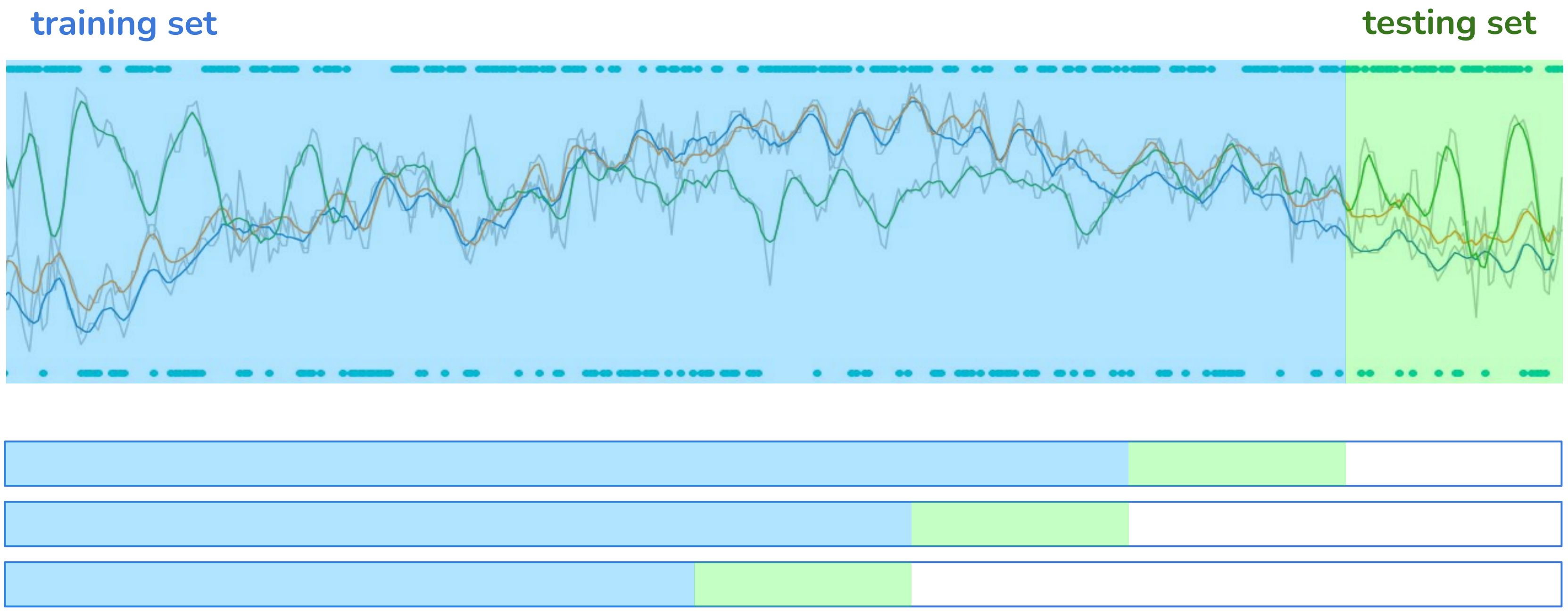
Training set construction



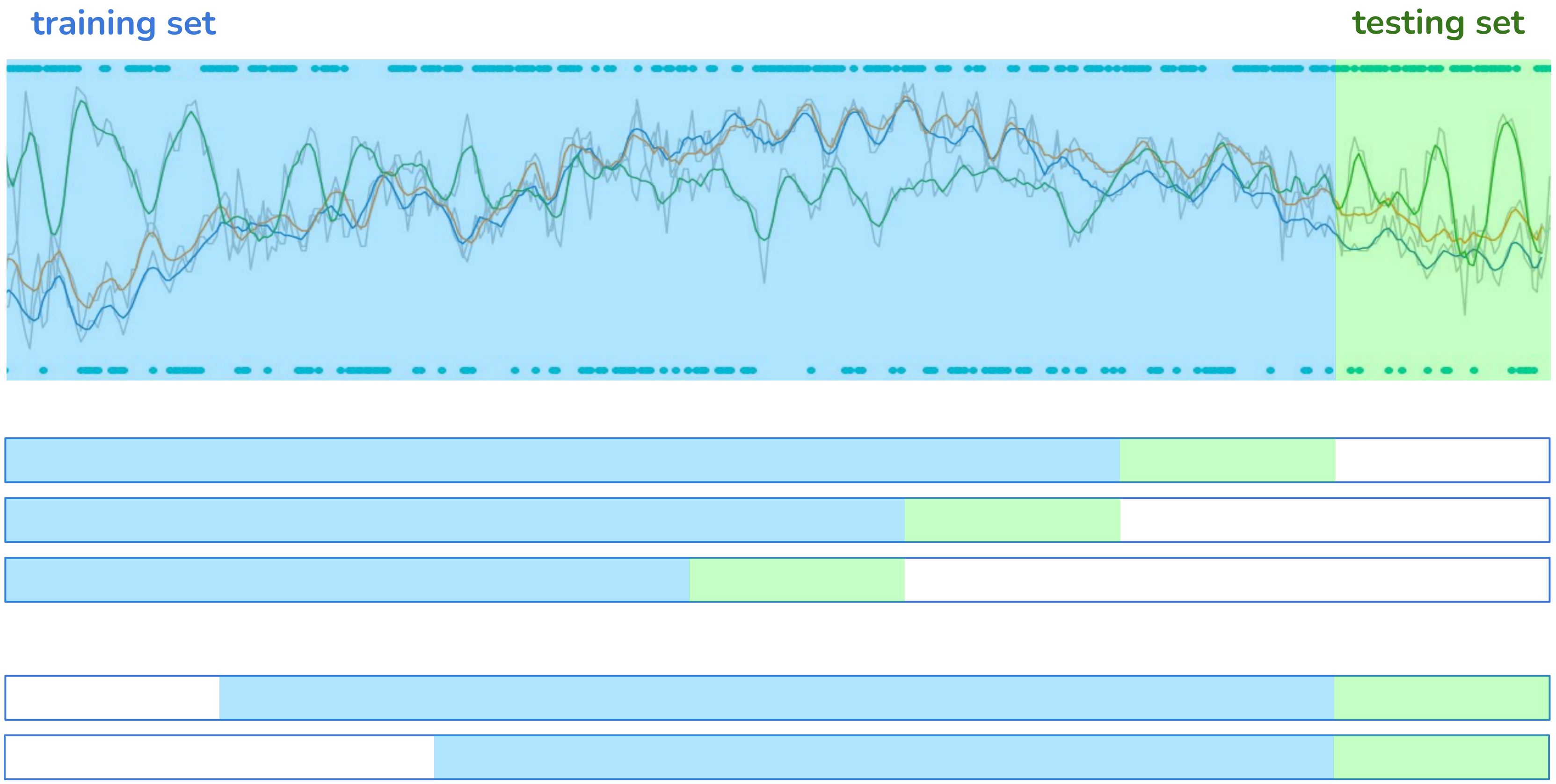
Training set construction



Training set construction



Training set construction

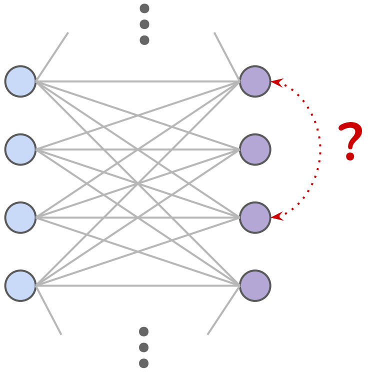


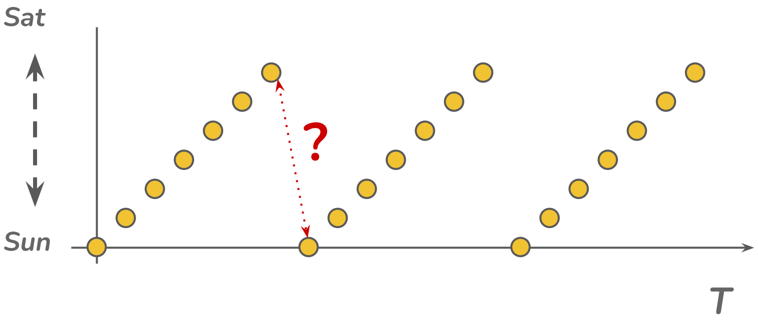
Feature Encoding – Seasonal dummy variables

- *(hour of day, day of week, ...)*
- *Numerical variables*
- *One-hot encoding*

Sun	0	0	0	0
Mon	**1**	0	0	0
Tue	0	**1**	0	0
Wed	0	0	**1**	0
Thu	0	0	0	**1**
Fri	0	0	0	0
Sat	0	0	0	0

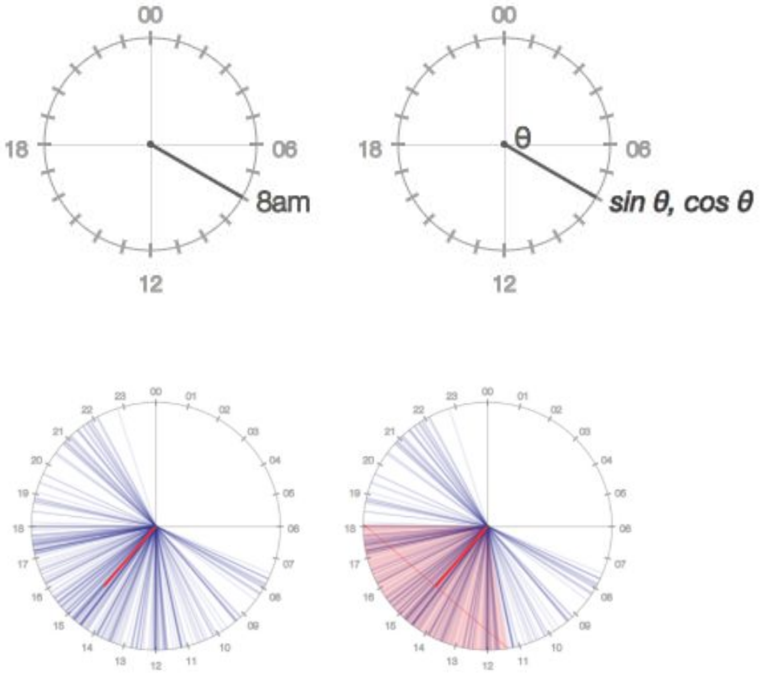
T →



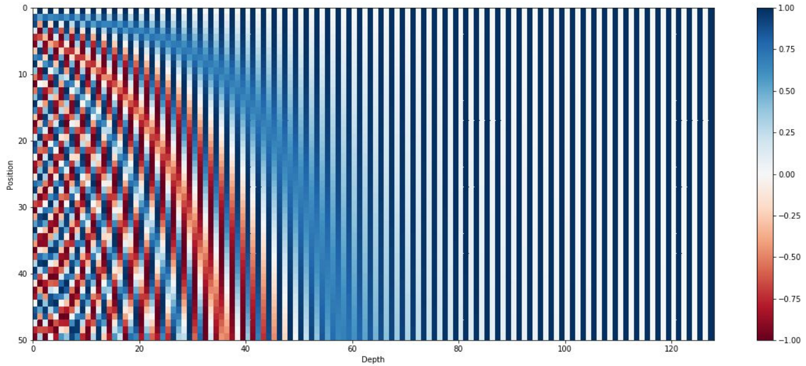


https://medium.com/life-at-hopper/ai-in-travel-part-2-representing-cyclic-and-geographic-features-4ada33dd0b22
https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Feature Encoding – Seasonal dummy variables



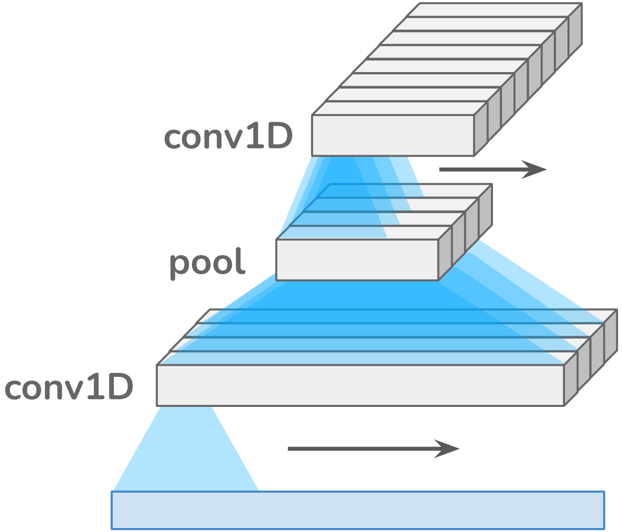
- *Circular encoding*
- *Positional embedding (transformers)*

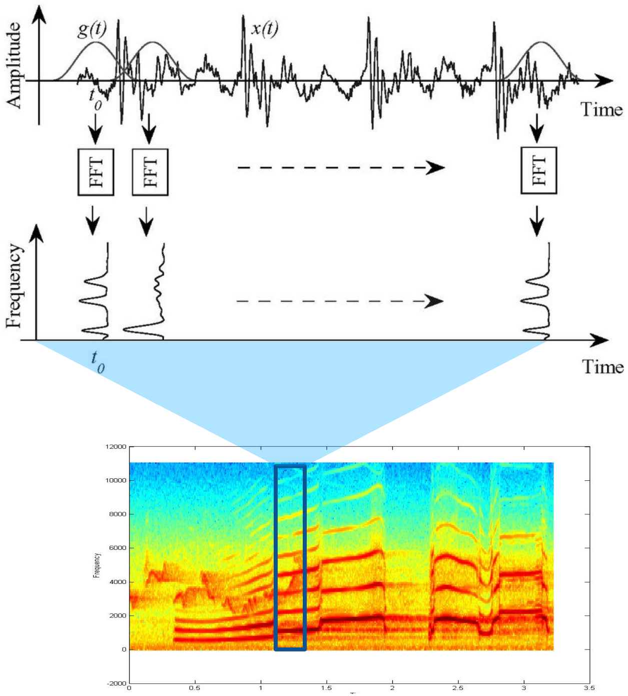


https://medium.com/life-at-hopper/ai-in-travel-part-2-representing-cyclic-and-geographic-features-4ada33dd0b22
https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

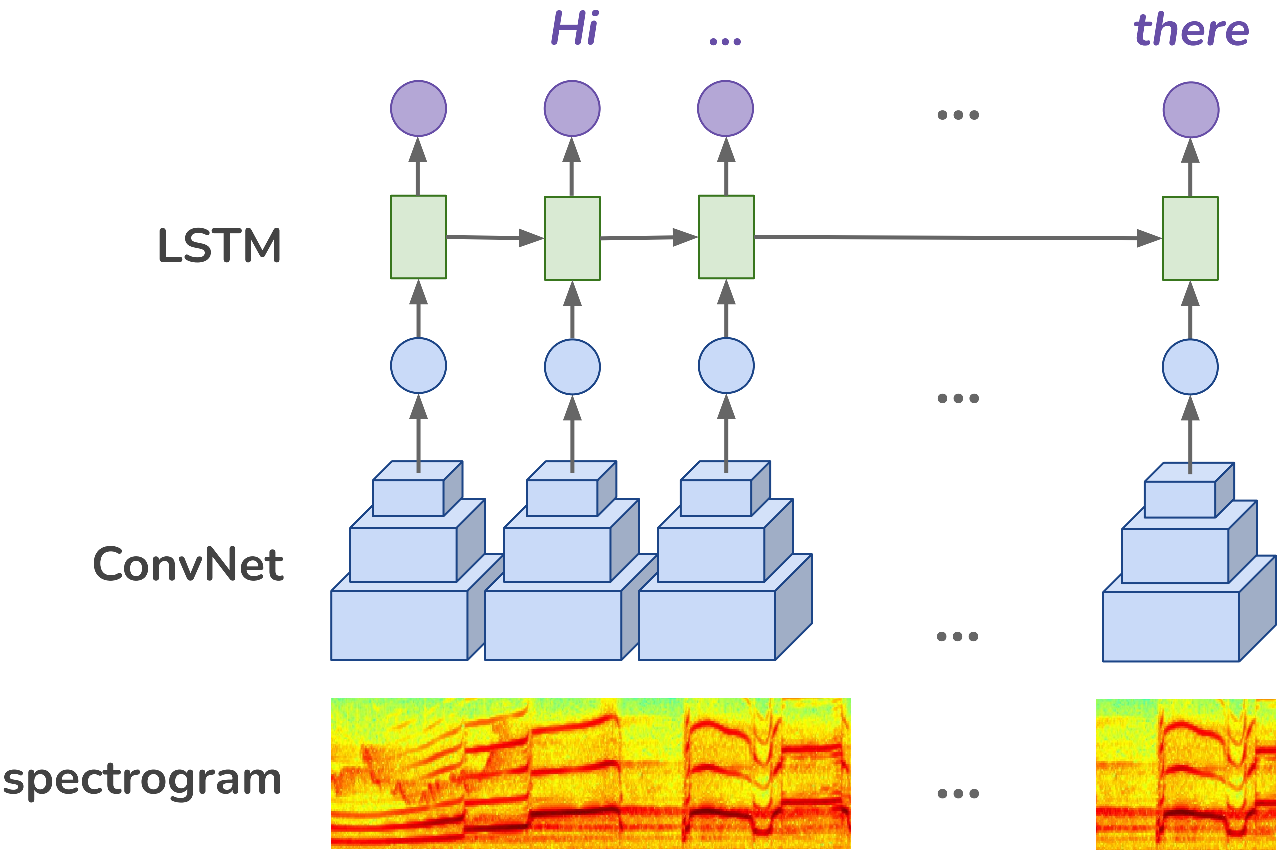
Feature Encoding II

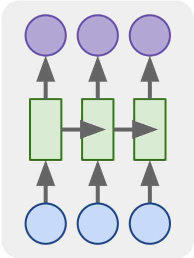
- Exchange for extra dimension
  - 1D Convolution & pooling
  - Short-Time Fourier Transform





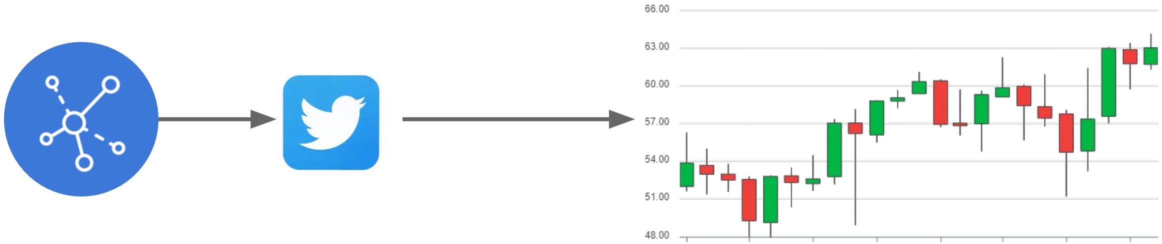
Speech recognition





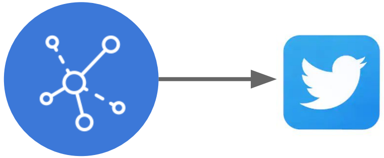
Machine Learning College

Time series prediction from textual data



Machine Learning College

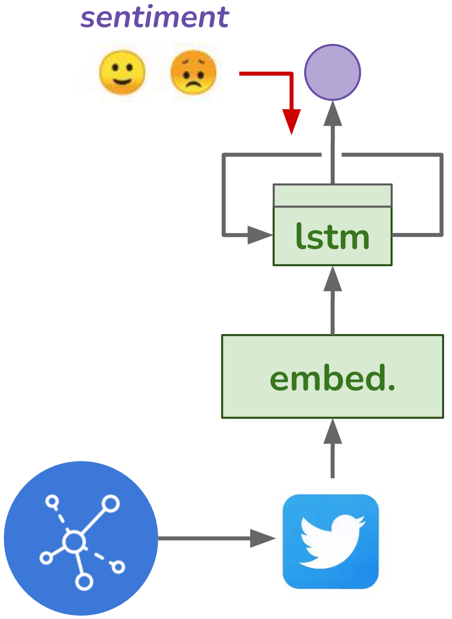
Time series prediction from textual data



Machine Learning College

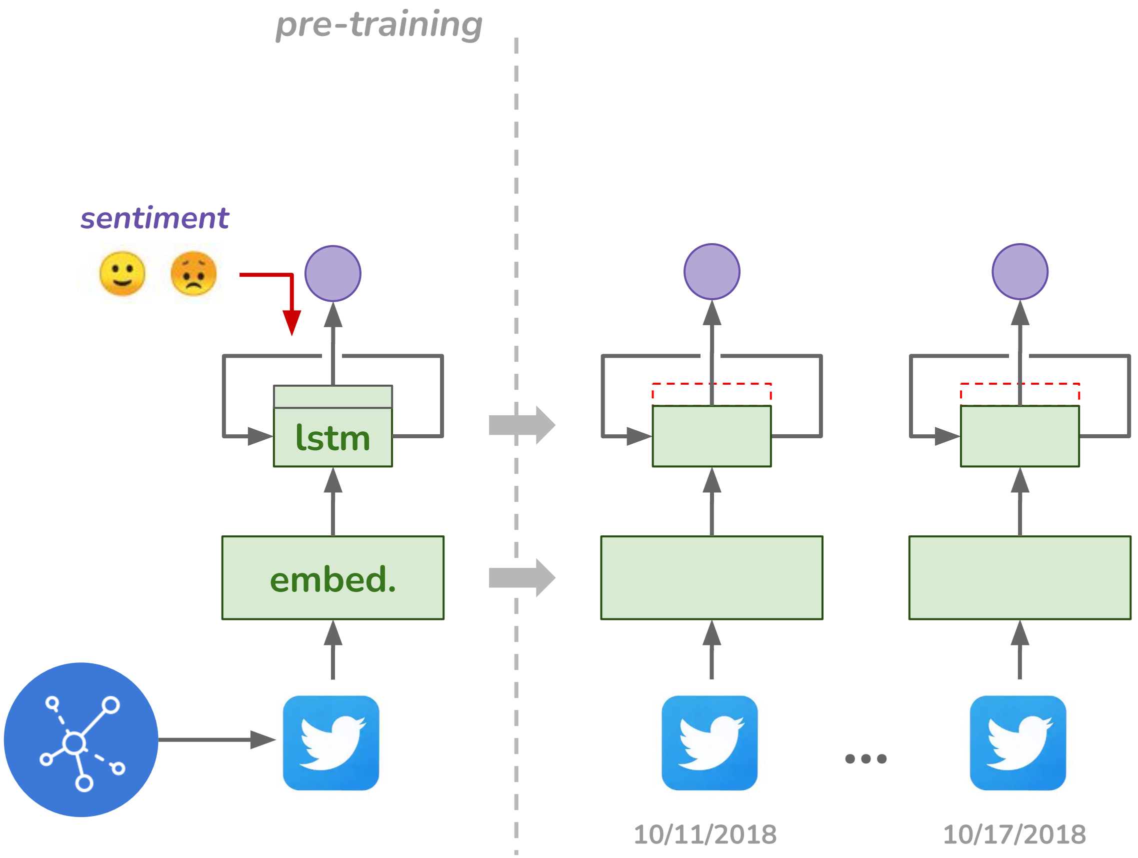
Pre-training with additional data

pre-training



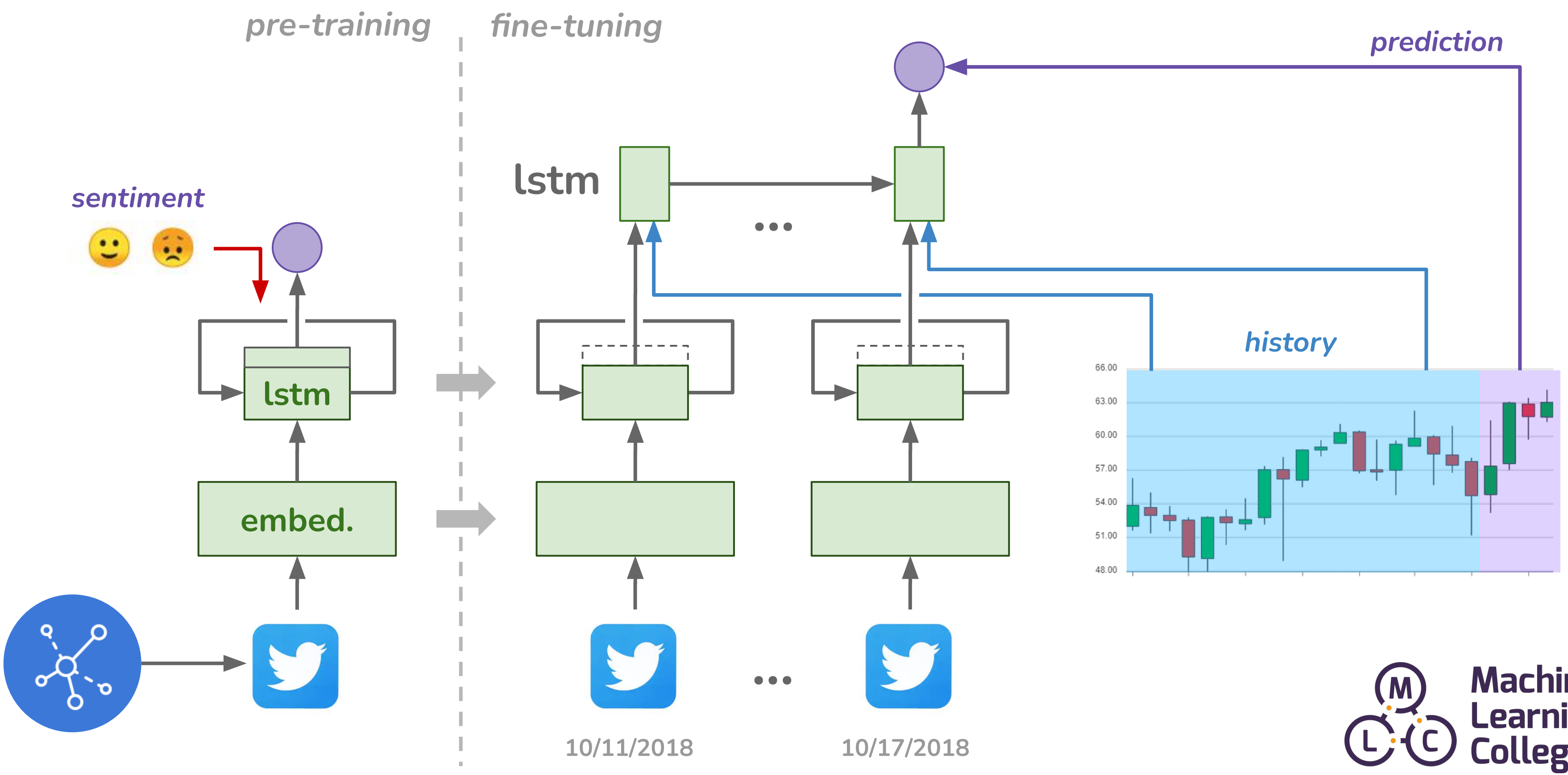
Machine Learning College

Transfering model & exposing feature layer



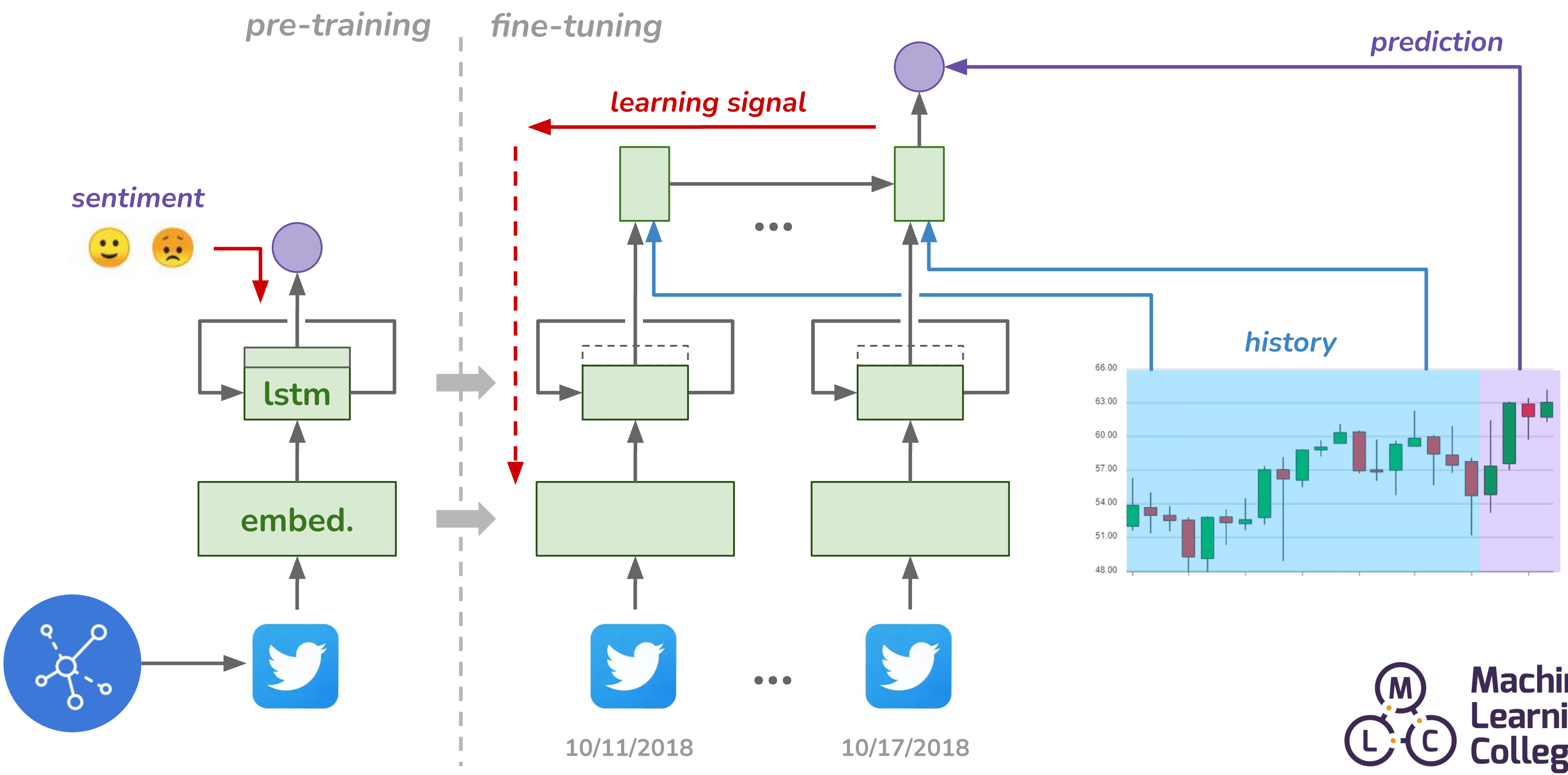
Machine Learning College

Fine-tuning with time series target data



Machine Learning College

Fine-tuning with time series target data



Machine Learning College

ML & Product Design

- ML should solve problems
  - *Easily solvable by human* ⇒ scale up & automate
  - *Not easily solvable by human* ⇒ too complex or not sensitive enough
- Automation is a spectrum
  - Collaboration instead of full automation
    - No automation
    - Scored set of possible decisions
    - Narrowed set of decision to approve
    - Veto before automatic execution
    - Full automation
- Models are imperfect
  - Right evaluation metrics
  - Expectation control / Automation bias
- End-to-end models?
  - Explainability
  - Configurability

Machine Learning College

ML Tips & Tricks

- Known your data
    - Visualize everything you can
    - Try to find patterns ⇒ *become the model yourself*
    - Look for noisy labels / missing data
    - Make sure your preprocessing is correct (especially vectorized code)
- Start with simple models
    - Build training & evaluation loop
    - Choose simple architectures first ⇒ *less room for errors*
    - Build baseline models for comparison ⇒ *even simple heuristics are useful*
- Train iteratively
    - Train without inputs ⇒ *yields another baseline model*
    - Overfit one batch ⇒ *something is off if you can't get zero loss*
    - Overfit the training set as far as you can
- Regularize
    - Early stopping ⇒ *best evaluation loss*
    - Make the model smaller ⇒ *less space for overfitting*
    - Get more training data ⇒ *more labels, data augmentation, pre-training*

http://karpathy.github.io/2019/04/25/recipe/