AI for Economics AI60003

Module 2

10th October, 2023

Analysis of Time-Series with AI/ML

Adway Mitra

How can AI/ML help in Time-series problems?

- Forecasting: Given past values, how to predict the future values?
- Classification: Can we identify whether any given time-series belongs to a particular class?
- Clustering: Can we identify which time-series are “similar” and group them together?
- Segmentation: Can we identify “change-points” in a time-series?
- Alignment: Given two time-series that are “similar” but mis-aligned, can we align them?

How can AI/ML help in Time-series problems?

- Forecasting: Given past values, how to predict the future values?

Economic application: GDP/stock price forecasting

- Classification: Can we identify whether any given time-series belongs to a particular class?

Economic application: Credit ratings to agencies/firms/economies

- Clustering: Can we identify which time-series are “similar” and group them together?

Economic application: Groups of firms/countries having similar GDP behavior

- Segmentation: Can we identify “change-points” in a time-series?

Economic application: Identify events that impact GDP/stock prices

- Alignment: Given two time-series that are “similar” but mis-aligned, can we align them?

Numerical Time Series

- Observations at any time-point may be a number of vector of numbers
- Eg. GDP time series, stock prices, prices of different commodities, …
- Machine Learning: build a mathematical model for the values as a function of time and past values
- Eg. Autoregressive Models
- Decision choices: which representation to use?

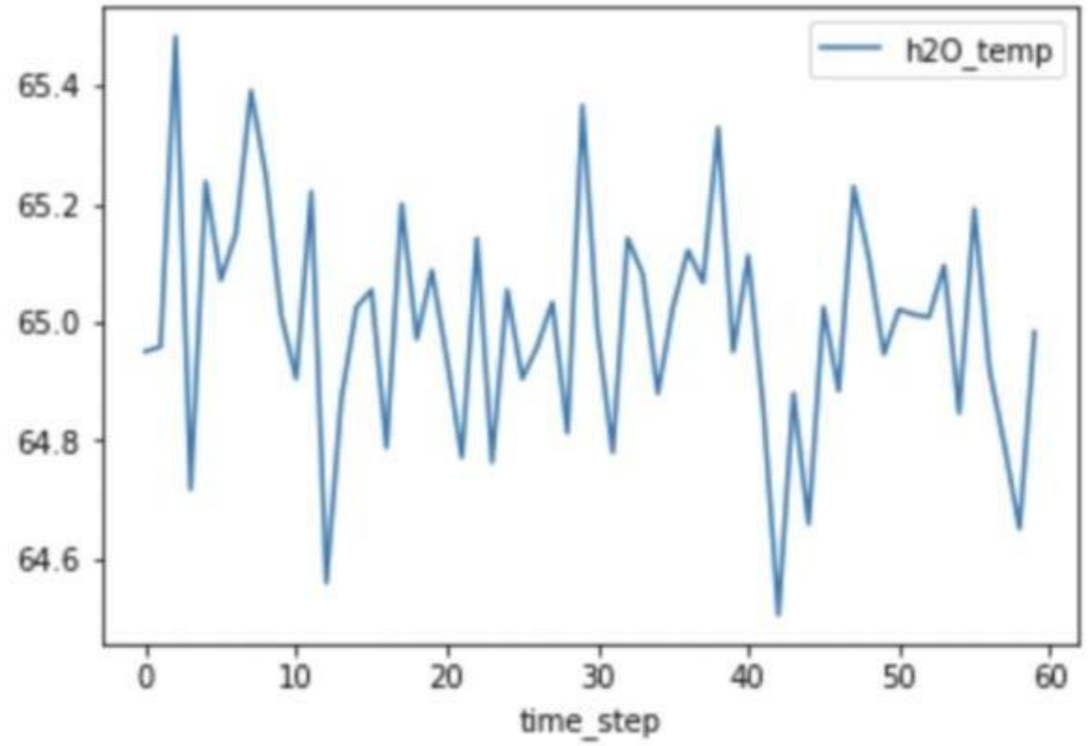
  - How many past values to consider? Which functions to use? How to combine different dimensions? How to combine external covariates?
- Learning problems: how to estimate the parameters?

Non-numeric Time Series

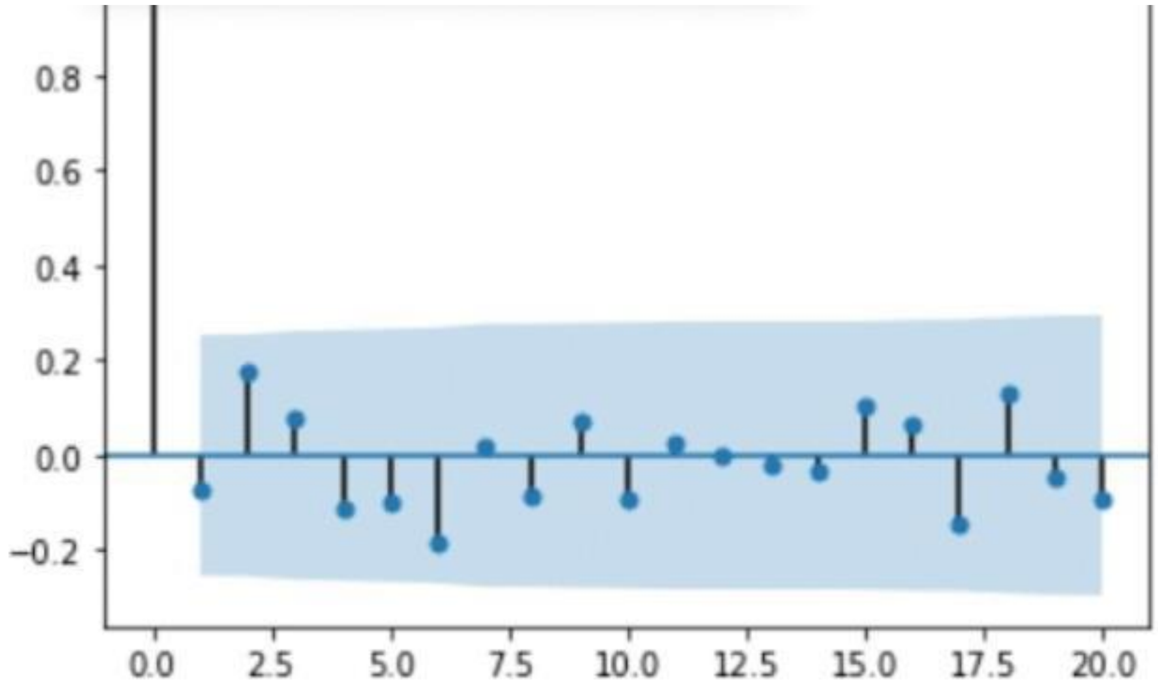
- A time-series of discrete values
- Eg. i) Most popular attributes (eg. colour) of any item (eg. cars) over time
- ii) Binary: whether the sales of an item will increase or decrease
- iii) Reviews: of a particular agency/service/personality over time
- Most mathematical models will not work
- Sequential pattern mining algorithms needed!

Coming back to numeric time-series ....

- Key property of numeric time-series : Autocorrelation!
- Correlation coefficient between RVs U,V = (Cov(U,V)) / StD(U)*StD(V)
- Are X(t) and X(t-1) correlated? How about X(t) and X(t-Δ)?



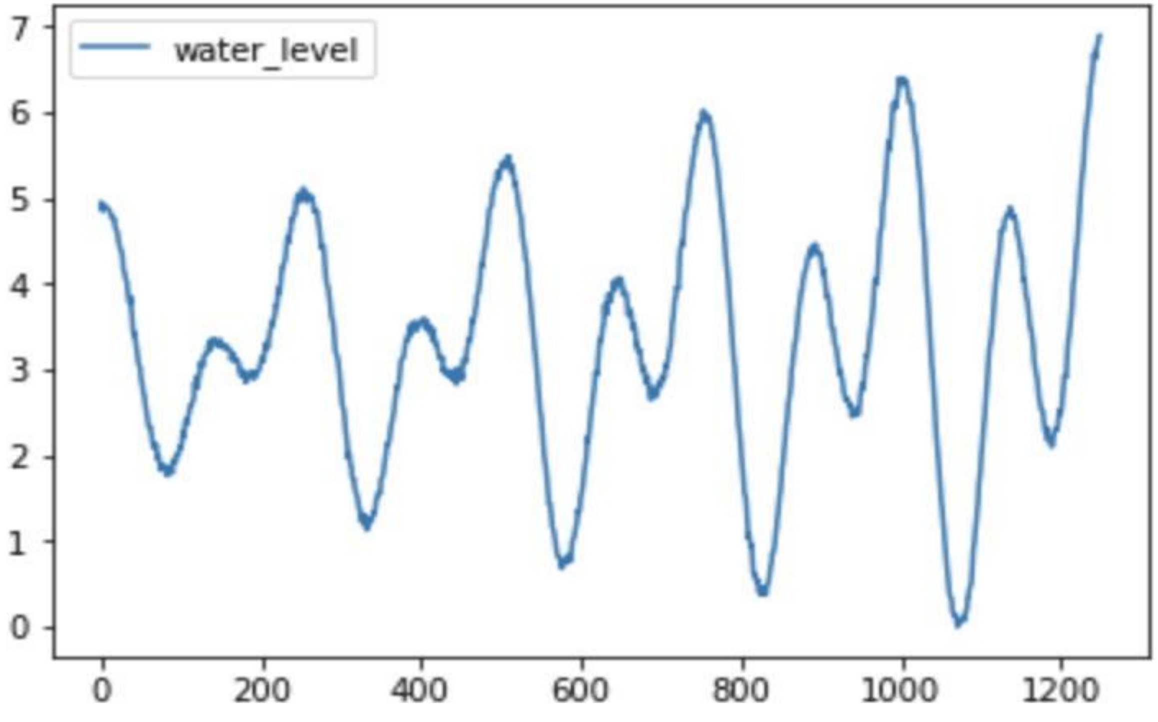
Sample time-series of random process



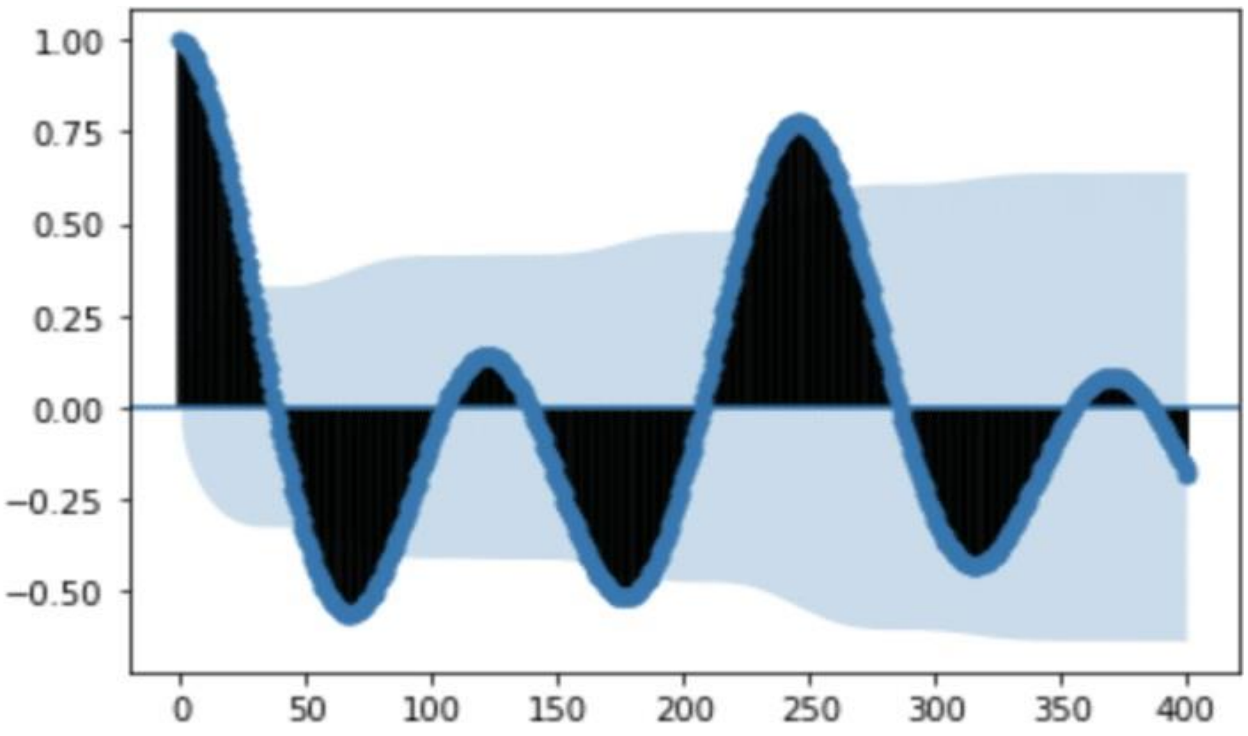
Plot of autocorrelation vs lag Δ: very low AR!

Coming back to numeric time-series ….

- Key property of numeric time-series : Autocorrelation!
- Correlation coefficient between RVs U,V = (Cov(U,V)) / StD(U)*StD(V)
- Are X(t) and X(t-1) correlated? How about X(t) and X(t-Δ)?



Sample time-series of random process



Plot of autocorrelation vs lag Δ: high AR at 250: indicates seasonality!

Autoregression of time-series

- Autocorrelation forms the basis of autoregression: using past values to predict future values!
- ARIMA (autoregressive integrated moving average)

$$X_t - \alpha_1 X_{t-1} - \cdots - \alpha_{p'} X_{t-p'} = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q},$$

Autoregression

$$\left(1 - \sum_{i=1}^{p'} \alpha_i L^i\right) X_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t$$

Where L is the lag operator, Lk(Xt) = X(t-k)

An ARIMA(p,d,q) process expresses this polynomial factorisation property with $p=p'-d$, and is given by:

$$\left(1 - \sum_{i=1}^{p} \varphi_i L^i\right) (1-L)^d X_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t$$

Assuming LHS has a factor (1-L) with multiplicity d

Machine Learning

- How to find the parameters of the model?
- Define a Loss function to compare the true values with predicted values
- Choose model parameters to minimize the Loss function!
- Need additional properties on the parameters? (eg. sparsity?)
- Need quantification of uncertainty? (eg. probability distribution?)
- Need to bring in “unobservable” covariates or causal variables?

Hidden Markov Model

Observations: sequential data Y(1), Y(2), ….. , Y(t-1), Y(t), Y(t+1), ….

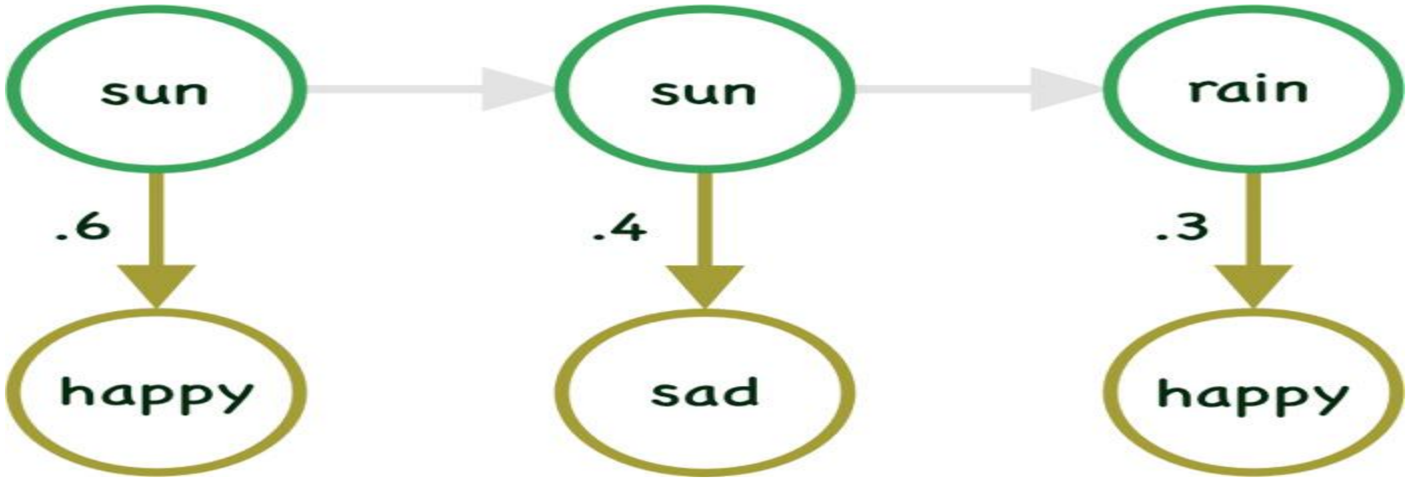
Model assumption: hidden state variable Z(1), Z(2), ….., Z(t-1), Z(t), Z(t+1), …..

Model assumption: Z is a discrete variable, taking K possible values

Model assumption: prob(Z(t)=j | Z(t-1)=i) = A(i, j) [Transition distribution]

Model assumption: prob(Y(t)=y | Z(t)=x) = f(x, y) [Emission distribution]

Model assumption: prob(Z(1)=z) = Π(z) [Initial state distribution]



Hidden Markov Model

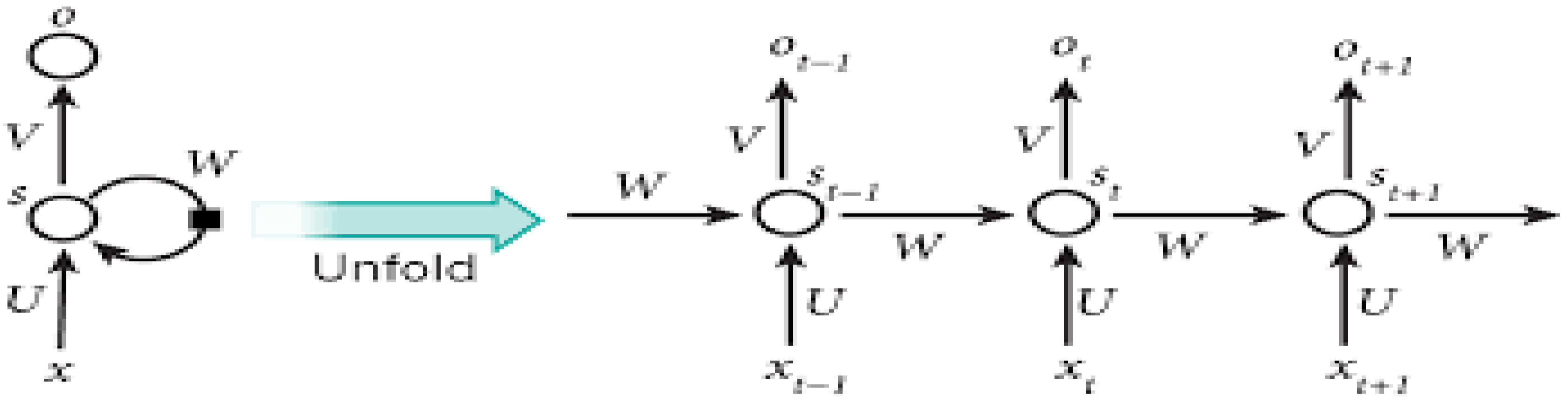
Estimate Z, given a sequence of observations Y and parameters Π, A, f
Given a set of sequences, estimate model parameters
Forecast future values of Y, given a sequence and model parameters
Sequence classification: Given parameters of several models, find which model is most appropriate for a sequences!

$$\text{Prob}(Y,Z \mid \Pi,A,f,) = p(Z(1)) * p(Z(2)|Z(1)) * p(Z(3)|Z(2)) * \ldots\ldots * p(Z(T)|Z(T-1)) *$$
$$p(X(1)|Z(1)) * p(X(2)|Z(2)) * \ldots\ldots * p(X(T)|Z(T))$$

[Marginalize over Z to get the model probability of the observations Y]

Recurrent Neural Networks

Another hidden-state approach, hidden state value continuous
Not probabilistic like HMM, transition function can be complex
Can use number of hidden state variables with instantaneous output
Hidden state dynamics: S(t) = f(W*S(t-1) + U*X(t)) where f is nonlinear
Output: O(t) = g(V*S(t)) where g is nonlinear



Functioning of RNN

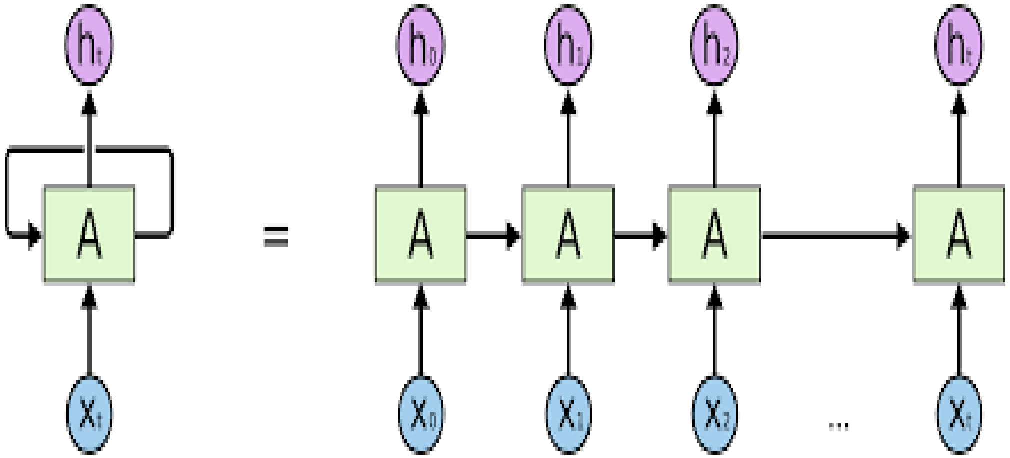
Training of RNN = estimation of parameters W and U
Trained RNN can be used to forecast future values of time-series
Stronger than HMM as not a Markovian model and can handle arbitrary functions and complex state variables
Parameters estimated using numerical methods like “Gradient Descent”
Estimation of parameters difficult if training sequences are long!
“Vanishing gradient” and “Exploding gradient” problems appear

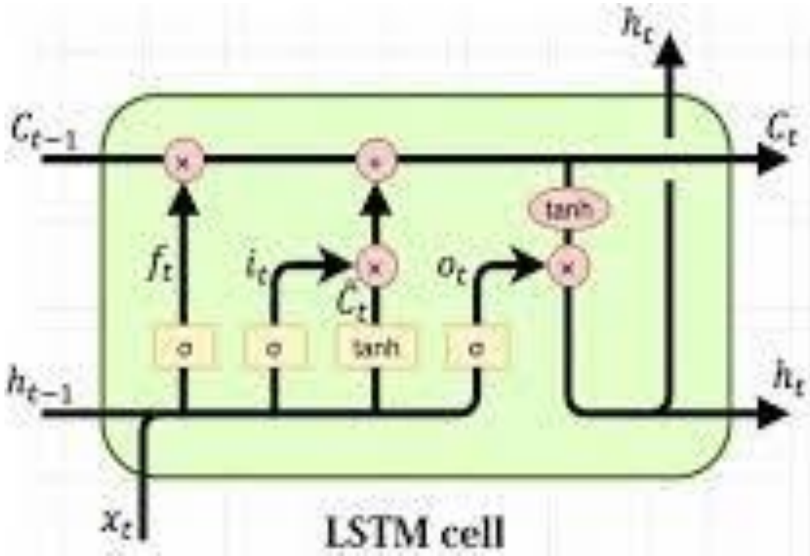
Long Short Term Memory

The “hidden state” is more complex than RNN, can retain values over long sequences

Cell state variable C(t): long-term memory, erased only if there is “forget” signal

Short-term memory h(t): updated at each step

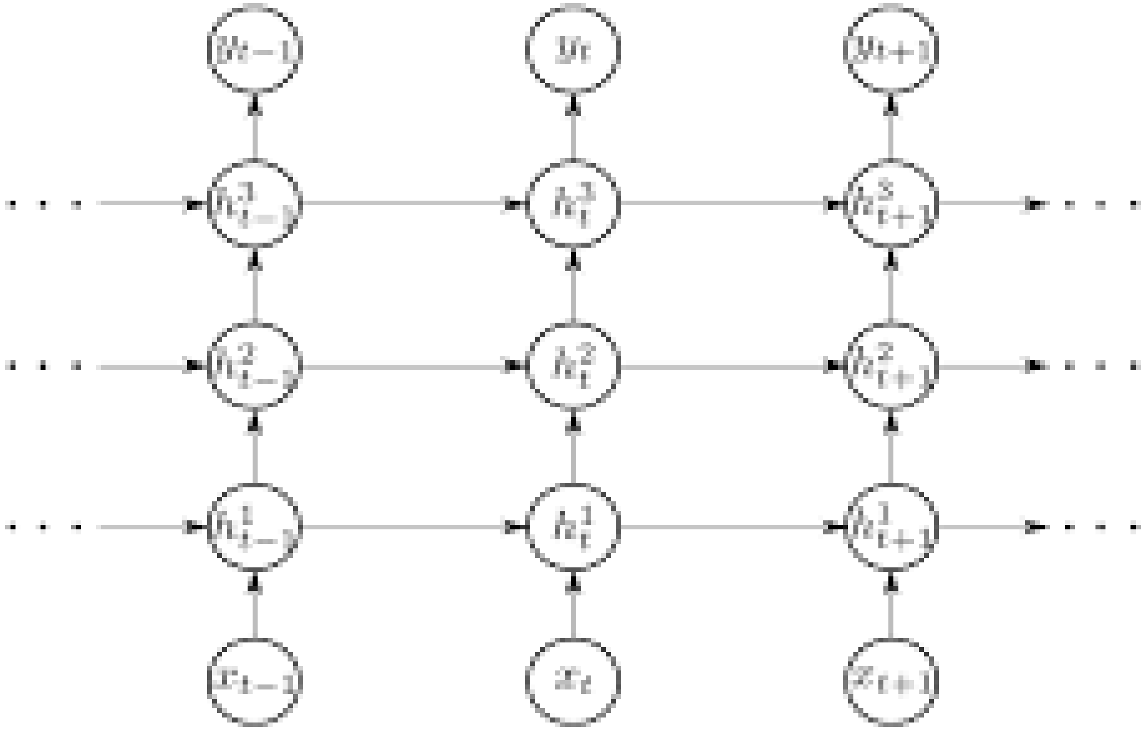


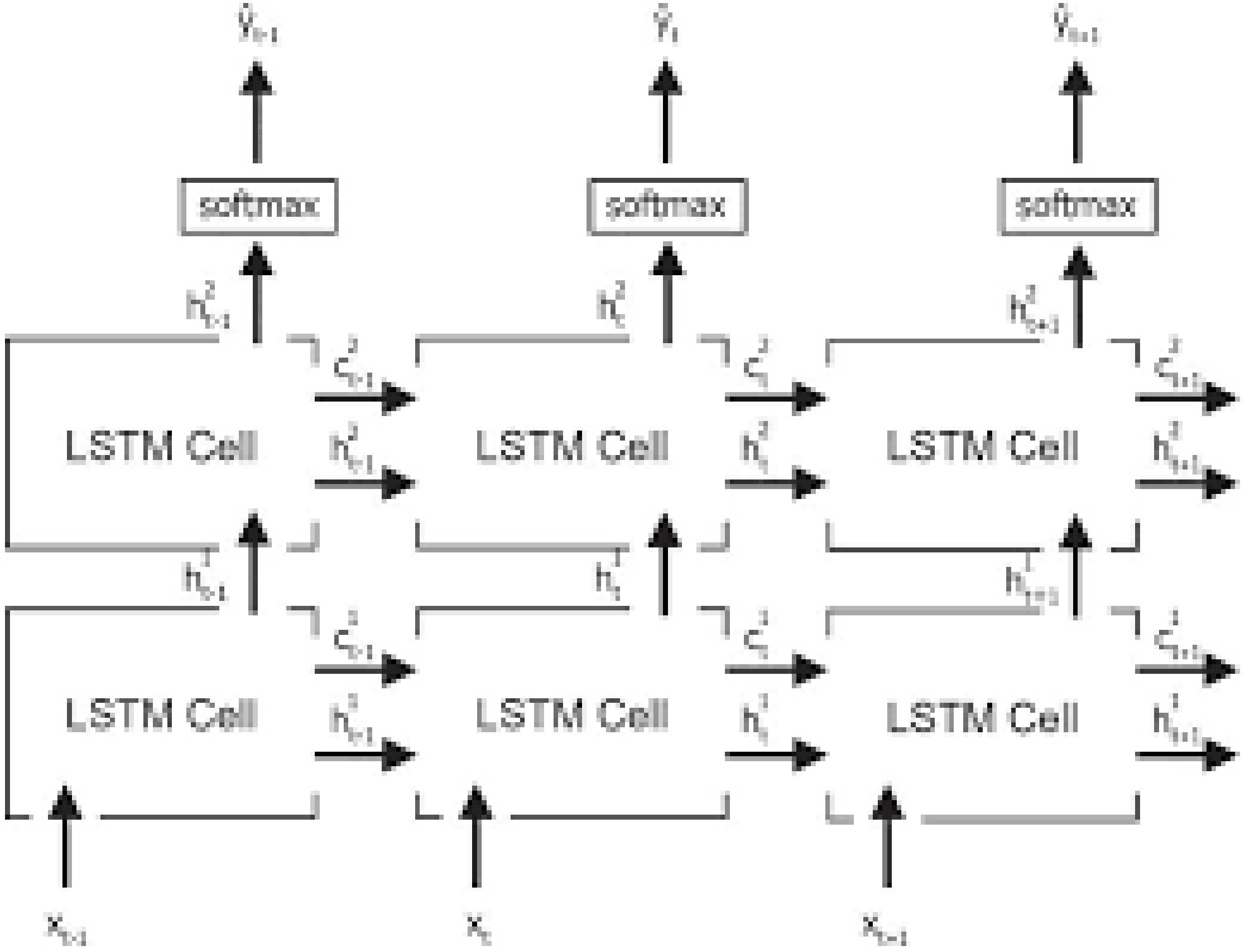


$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$h_t = \tanh(C_t) * o_t$$

Stacked RNN and LSTM

Richer representation possible with multiple hidden layers





Time Series operations

Short-term Forecasting: The dynamics of the model allows us to predict output using past inputs
Long-term Forecasting: The next step's input itself can be predictable
Time-series classification:

1) Provide the entire time-series as input
2) Hidden state forms an intermediate representation (vector/matrix/tensor)
3) Classify based on this representation using another neural network

Time-series translation/mapping: use the hidden state representation to generate a new time-series!